Universidad del País Vasco
Euskal Herriko Unibertsitatea

Doctoral Thesis

Interconnected Services for Time-Series Data Management in Smart Manufacturing Scenarios

Kevin Villalobos Rodríguez

Supervised by

Prof. Dr. José Miguel Blanco
Prof. Dr. M. Aranzazu Illarramendi

A dissertation submitted to the Department of Computer Languages and Systems of the University of the Basque Country (UPV/EHU) for the degree of Doctor of Philosophy (Ph.D.) in Informatics Engineering

Donostia-San Sebastián, June 2020

Abstract

The fourth industrial revolution has given rise to what is called Smart Manufacturing, addressing the use of modern Information Technologies to transform the acquired data across the supply chain into manufacturing intelligence, in order to achieve meaningful improvements in all aspects of manufacturing. The rise of Smart Manufacturing, together with the strategic initiatives carried out worldwide, have promoted its adoption among manufacturers who are increasingly interested in boosting data-driven applications for different purposes, such as product quality control, predictive maintenance of equipment, etc. However, the adoption of these approaches faces diverse technological challenges with regard to the data-related technologies supporting the manufacturing data life-cycle. The main contributions of this dissertation focus on two specific challenges related to the early stages of the manufacturing data life-cycle: an optimized storage of the massive amounts of data captured during the production processes and an efficient pre-processing of them. Moreover, with regard to the later stages, a third main contribution is proposed, that leverages advanced data analytics to conduct a predictive maintenance of equipment.

The first main contribution of this research work consists in the design and development of a system that facilitates the pre-processing task of the captured time-series data through an automatized approach that helps in the selection of the most adequate pre-processing techniques to apply to each data type. The proposed system is available as a visual-interactive web system that provides a wide range of pre-processing techniques for the different tasks related to time series cleaning and dimensionality reduction; as well as, it provides some recommendations on which techniques are more suitable and have more potential to work for each type of time series.

The second main contribution is the design and development of a three-level hierarchical architecture for time-series data storage on cloud environments, that helps to manage and reduce the required data storage resources (and consequently its associated costs). The proposed architecture store reduced representations of time-series data (obtained by applying time series reduction techniques) to considerably reduce the required data storage resources without limiting the use of the data for further analysis purposes.

The third main contribution resides in the design and development of an alarm prediction system that allows to anticipate the activation of different types of alarms that can be produced on a real Smart Manufacturing scenario. An early prediction of those alarms could serve for different purposes, such as the predictive maintenance of equipment or production optimization. The system applies deep learning techniques in a two-stage approach on which first, a time series forecaster predicts the future measurements of various sensors, and then, distinct classifiers determine whether the predicted measurements will trigger an alarm or not.

Finally, it is worth mentioning that the utility and applicability of the proposed contributions have been contrasted and validated in a real-world scenario representing a relevant instance of the Smart Manufacturing scenarios where these contributions are targeted at.

Resumen

Distintas iniciativas estratégicas en el ámbito de la fabricación impulsadas en todo el mundo han promovido el auge de un nuevo paradigma de fabricación denominado *Fabricación Inteligente*. Este tipo de fabricación permite a las empresas desarrollar aplicaciones basadas en los datos de fabricación, orientadas a lograr un mejor control de la calidad de los productos, un mantenimiento predictivo del equipo, etc. No obstante, la adopción de este enfoque supone diversos desafíos tecnológicos con respecto a las tecnologías relacionadas con el procesamiento y la gestión de los datos durante su ciclo de vida. Las principales contribuciones de este trabajo de investigación se centran en dos desafíos específicos relacionados con las primeras etapas del ciclo de vida de los datos: un almacenamiento optimizado de cantidades masivas de datos capturados durante los procesos de producción y un preprocesamiento eficiente de los mismos. Asimismo, con respecto a las etapas posteriores, se propone una tercera contribución que utiliza técnicas de análisis de datos avanzadas para llevar a cabo un mantenimiento predictivo del equipo de fabricación.

La primera contribución consiste en el diseño y desarrollo de un sistema que ayuda en la selección de las técnicas de pre-procesamiento más adecuadas para aplicar a diferentes tipos de series temporales. El sistema consiste en una aplicación web interactiva que proporciona una amplia gama de técnicas de pre-procesamiento para las diferentes tareas relacionadas con la limpieza y la reducción de la dimensionalidad de las series temporales; así como algunas recomendaciones sobre qué técnicas son más adecuadas y pueden funcionar mejor para cada tipo de serie temporal.

La segunda contribución consiste en el diseño y desarrollo de una arquitectura jerárquica de tres niveles para el almacenamiento de datos de series temporales en entornos de computación en la nube, que permite reducir los recursos de almacenamiento de los datos (y, en consecuencia, sus costos asociados). La arquitectura propuesta almacena representaciones reducidas de las series temporales (obtenidas mediante la aplicación de técnicas de reducción de series temporales) que permiten reducir considerablemente el espacio de almacenamiento requerido, sin limitar el uso de los datos para análisis posteriores.

La tercera contribución consiste en el diseño y desarrollo de un sistema de predicción de alarmas que permite anticipar la activación de diferentes tipos de alarmas que pueden producirse en un escenario real de Fabricación Inteligente. La predicción de esas alarmas puede servir para diferentes propósitos, como el mantenimiento predictivo de los equipos o la optimización de la producción. El sistema utiliza diferentes técnicas de *aprendizaje profundo* para construir un pronosticador que predice las mediciones futuras de varios sensores, y varios clasificadores que determinan si las mediciones predichas activarán una alarma o no.

La utilidad y la aplicabilidad de las contribuciones propuestas se han contrastado y validado en un escenario real, que representa una instancia relevante de los escenarios de Fabricación Inteligente para los que se proponen estas contribuciones.

Acknowledgements

I would like to begin this dissertation expressing my gratitude to several people who have played a very important role in the completion of this PhD thesis.

First of all, I wish to thank my supervisors, Arantza Illarramendi and José Miguel Blanco, for giving me the opportunity to carry out this research work under their supervision and for their support that has been essential to complete it. Special thanks to Arantza for encouraging me to do this PhD and for how she has been involved during the whole PhD.

Many thanks to all the colleagues that I have met during these years in the BDI research group, for their support both professionally and personally; and special thanks to Victor and Borja for their invaluable help in the development of the different artifacts supporting the contributions presented in this PhD. I would also like to thanks to the different professors that I have met in the faculty, especially to those who helped me to solve some doubts related to the different contributions of this research work.

Many thanks to the people from the companies involved in the real-world business setting with which this research work has been carried out. Specially to Fernando and Gurutz for their generosity when allowing us to access to their data and for their availability to collaborate and help us; and also to Esti for her help when accessing and understanding the data.

I would like to thanks also the colleagues I have met during my research stage in the ESAT from the KU Leuven for accepting me as a guest researcher, for their hospitality and their help during my research stage and afterwards.

Finally, my greatest gratitude goes to my friends, my family, and especially to my partner for their unconditional help, care and affection and for accompanying me during these years.

Funding

The author of this dissertation wants to thank the Basque Government for its support funding this research.

Contents

List of Figures

List of Tables

A mi familia y amigos, por vuestro apoyo

Chapter 1

Introduction

We are witnessing a digital transformation era on which ubiquitous sensors and Internet of Things (IoT) devices enable the *datafication* of virtually any aspect of digitally connected individuals and machines [Ciu18]. The captured data by these devices during the continuous monitoring of human activities, industrial processes or even smart cities leads to so large and complex datasets that it becomes difficult to process such "Big Data." In fact, the so-called Big Data and, by extension, data processing and exploitation technologies constitute one of the most relevant open research problems in the field of Information Technologies in last years, where innovative technology such as Cloud Computing has emerged to cope with these challenges and offer new ways of extracting value and knowledge from these unprecedented volumes of data. The ability to effectively manage information and extract knowledge has been seen as a key competitive advantage across different sectors [CCW16], where data processing and exploitation technologies, favored by the intensive promotion of Big Data tools and other synergic technologies such as the Internet of Things (IoT), Cloud Computing and Knowledge Discovery in Databases (KDD) has led to create a new kind of economy which drives from the data, coining the concept of data driven economy [Eur14].

Data-driven economy has been stated as one of the keys for economic development at a global scale, spurring new products and services as well as new business processes and opportunities. According to the European Commission in the report published in 2019 [IDC19] the data industry as a whole (companies whose main activity is focused in the production and delivery of data-related services or products) comprised approximately 283.000 companies in 2018 in the EU and is expected to grow up to 715.000 by 2025. Similarly, the study found that the number of data workers (workers who collect, store, manage and/or analyze, interpret and visualize data) was 7.2 million in 2018 and was estimated that under a high-growth scenario, the number of data workers in Europe will increase up to 13.1 million by 2025. Finally, the overall value of the data economy almost reached €377 billion in 2018 (nearly 2.6% of the EU GDP) and by 2025 the EU data economy is expected to increase up to €1054 billion (6.3% on the EU GDP).

One sector where data-driven economy is being implanted all over the world is the manufacturing industry, as a means to revitalize the global competitiveness of this sector, given its impact in the economy of many countries, where according to the European Commission, manufacturing accounts for 16% of Europe's GDP [Eur19b]. The instantiation of this data-driven economy in the manufacturing sector has given rise to the development of Smart Manufacturing [Niñ17], as a global-scale overarching term for different initiatives and strategies, addressing the usage of modern information technologies, to transform, the acquired data across the product life-cycle into *manufacturing intelligence* in order to achieve meaningful improvements in all aspects of manufacturing [TQLK18].

Smart Manufacturing is defined in [DEP+12] upon two main ideas: the compilation of manufacturing data (i.e., records of products with data about their history, state, characteristics, quality, etc.), and the application of manufacturing intelligence to those data, so that their exploitation allows manufacturers to manage, plan and predict, specific circumstances in order to optimize their production [Niñ17]. This opens the door to important business opportunities for manufacturing companies either to apply this approach internally or to *servitize* their business [NBI15], in order to help other manufacturing firms to shift their production towards Smart Manufacturing-oriented approaches.

In general, the deployment of Smart Manufacturing approaches demands the introduction of data-related Information Technologies and digital platforms supporting it. Moreover, the design and implementation of such technologies and platforms faces diverse research and innovation challenges including, among others, the following: improved methods of gathering valuable machine data, and data integration across different sources of heterogeneous nature; automated data quality monitoring algorithms ensuring the integrity and quality of the captured and communicated data; data architectures matching industrial needs; implementation of advanced data analytics technologies and methods, such as artificial intelligence techniques, that adapt reliably according to changes in industrial processes, and are capable to predict all kinds of behaviour that would enable predictive maintenance of the equipment, forecasting of product quality, anticipating faults, etc. [TWW17], [EFF16].

The wide spectrum of these technological challenges and their complexity, presents a major threat for the core competencies of specialized manufacturers, which often lack the capabilities needed for the development of the required information technologies to shift their businesses towards Smart Manufacturing [TWW17]. Therefore, they demand the support of specialized information technology suppliers [Eur19a] whose business is focused on supplying *Smart Services* [KRH+14] and products for companies adopting Smart Manufacturing approaches. Such is the case of the alliance of companies that make up the real advanced manufacturing scenario that gives the context for this research work (described in detail in Section 3.1), on which the overarching goal is to provide contributions that helps to develop innovative data-driven smart services for the development of Smart Manufacturing approaches.

1.1 Scope of this Research Work

This research work has been conducted in a real-world scenario, conformed by an Information Technologies Services Provider (ITS Provider) supplying smart services to diverse Smart Manufacturing scenarios, that granted the access to an ongoing *smartization project* fruit of the strategic partnership established with a Capital Equipment Manufacturer (CEM) deploying a data-driven servitiazation strategy in an extrusion based manufacturing sector distributed worldwide. Within this context, different opportunities arise for relevant contributions aligned with the achievement of the goals established for Smart Manufacturing approaches and the challenges related to them. This work is focused on three specific challenges; two of them related to the requirements of an ITS Provider with regard to the early stages of the data life-cycle, and a third one related to advanced analytics demands that arise from the specific needs of an advanced manufacturing environment. The three challenges on which this research work has been focused are the following:

1. The automatization of the pre-processing tasks to clean and reduce the dimensionality of heterogeneous industrial time-series data captured by sensors of different nature through the recommendation of the most suitable techniques to apply for each type of time series.

2. The design of a more efficient architecture for the storage of industrial time-series data, that reduces the associated costs of the cloud infrastructure required for the storage of the massive-scale amounts of data coming from large-scale sensor networks deployed on different manufacturing plants that compose a Smart Manufacturing scenario.

3. Devising a flexible and extensible approach to conduct a predictive maintenance of the equipment of a manufacturing plant through the early prediction of the activation of different alarms.

The aforementioned challenges outlines the research scope of this work aiming to build purposeful solutions that; on the one hand, are based on the needs and requirements of a real Smart Manufacturing context that provides the ground for a field validation of the proposed solutions in a real-world setting, ensuring that they are useful for the practitioner audience and their environment; and on the other hand, are supported by the identified synergies and opportunities for innovative contributions to the state of the art in the related research areas.

1.2 Method for this Research Work

The research scope outlined by the challenges described in the previous section points at an important characteristic of this work; instead of being driven by a specific research and knowledge area, it is driven by a wider analysis focus around the requirements related to a real Smart Manufacturing scenario composed by an ITS Provider and a CEM aiming to shift

their business towards servitization approaches. This scenario opens the door to opportunities for very interesting contributions, due to the applicability of scientific approaches to real-world problems; which in turn, result arduous, due to the intrinsic complexity of a real manufacturing scenario.

In fact, when it comes to working with time-series data (which is the most common data type in Smart Manufacturing scenarios), it has been observed on a recent report on the *First and Second Interdisciplinary Time Series Analysis Workshop* (ITISA) [PB19] that there exist a slight disconnect between the needs of scientists and practitioners that process and analyze time series, and Computer Science (CS) researchers that work on time series. This disconnection is mainly due to the fact that the problems that CS researchers have been studying are for the most part simplified, clean, and sanitized versions of the real problems and analysis workflows that practitioners have to address in the real world.

The accomplishment of the goals established by the research scope of this work implies a research work that analyzes: on the one hand, the manufacturing context that builds-up the real-world scenario of this research work; and on the other hand, the related work from different research and knowledge areas, in order to identify synergies with relevant related work and discover limitations that could serve as opportunities that would lead to interesting contributions. The method followed to achieve the established goals is based on two main methodological approaches: *Design Science Research* and *Case Study Research*.

Case Study Research [TSM08] enables researchers to learn by studying the innovations implemented by professionals and to capture and formalize their knowledge. A case study provides two main benefits for this research work: on the one hand, it allows to capture a more detailed characterization of targeted Smart Manufacturing scenarios, through the analysis of a relevant instance of those scenarios and their specific needs and requirements. On the other hand, it provides the ground for testing the proposed solutions in a real-world setting; which allows to assess the contributions of design science research when applied to real Smart Manufacturing scenarios.

In order to conduct the case study, this research work has been carried out in collaboration with an ITS Provider that supplies its services to various Smart Manufacturing scenarios. This collaboration granted the access an ongoing smartization project fruit of the strategic partnership established between the ITS Provider and a CEM deploying a data-driven servitiazation strategy in an extrusion based manufacturing sector distributed worldwide. This allows to interact with relevant stakeholders in the involved companies for the identification of problems, challenges and opportunities that arise in the services offered by the ITS Provider. Moreover, it also allows to access to the data coming from the monitored facilities; and a better understanding of these data behaviour and properties, provided by the domain experts from the CEM.

Design Science Research [PTRC07] provides a methodology for research in information systems with the aim of building purposeful design artifacts that on the one hand, are based on the needs and requirements of a Smart

Manufacturing context; and on the other hand, are supported by the identified synergies and opportunities for innovative contributions to the state of the art in the related research areas. These foundations grant rigor and relevance to the solutions proposed as contributions of this research work, at the same time that they ensure that are valid research contributions for the academic audience and useful contributions for the practitioner audience and their environment.

The case study research allowed the observation and analysis of the main characteristics of a real scenario and the requirements for an effective solution to the challenges that arise in it. In addition, it allowed the identification of the relevant research and knowledge areas to examine and integrate in this research work (see Section 3.2): techniques and strategies for time-series data pre-processing (including time series cleaning and dimensionality reduction), time series management systems and architectures for the management of time-series data, Knowledge Discovery and Data Mining methods for advanced data analytics, and predictive maintenance of equipment strategies in industrial environments. Thus, the contributions presented in the following subsection are sustained by the examination and identification of synergies with relevant proposals in these areas, as well as the discovery of opportunities to overcome their limitations to address practical aspects of a real-world scenario.

1.3 Main Contributions of this Research Work

The first main contribution of this research work consist in the design and development of a *system that efficiently guides a data engineer in the task of pre-processing raw time-series data* coming from industrial sensors in data-enabled Smart Manufacturing (Industry 4.0) scenarios. The proposed system is available as a visual-interactive Web application that provides various functionalities for time-series data cleaning and dimensionality reduction. On the one hand, it offers a wide-spectrum of available techniques for detecting and handling outliers, removing noise and imputing missing values in time series. Moreover, the system provides some recommendations for the selection of the appropriate techniques and parameter values required by them. On the other hand, it allows obtaining an adequate reduced syntactic representation of raw time-series data, while preserving their main characteristics. Dealing with those reduced representations, data storage and transmission costs can be decreased, as well as some analysis tasks associated with time-series processing can be optimized (e.g., time series similarity search, time series clustering, and time series data mining). Those reduced representations are obtained by applying the time series reduction techniques suggested by a machine learning-based model that given a time series, recommends the most appropriate reduction techniques.

The second main contribution is the design and development of *a three-level hierarchical architecture for an efficient storage of time-series data* on cloud environments, that helps to manage and reduce the considerable costs associated with the storage of the captured data in Smart Manufacturing scenarios. The architecture follows a multi-temperature data management paradigm on which the temperature tiers *hot*, *warm* and *cold* are considered, and thereby, it is materialized as a three-level hierarchical architecture. In the first level of the architecture *(Hot Storage)*, the most recent raw time-series data (which are usually the most relevant data to retrieve) are stored on Solid-State Drives (SSDs), for a short period of time, providing fast access for real-time applications. In the second level *(Warm Storage)*, recent raw time-series data are stored on magnetic Hard Disk Drives (HDDs), for a medium period of time. In the third level *(Cold Storage)*, as data get older and are more rarely accessed, a reduced representation of the data are stored in HDDs, for a longer period of time, allowing to keep more data with the same storage resources (i.e., widen the time-window of the stored time-series data). The main novelty of the proposed architecture relies on the nature of the third level, where a reduced representation of the time series is obtained by using different types of time series dimensionality reduction techniques, recommended by the system developed in the first main contribution.

The third main contribution of this research work resides in the design and development of *a flexible alarm prediction system following a forecaster-analyzer approach.* This system allows to anticipate the activation of different types of alarms that can be produced on a real Smart Manufacturing scenario and thus, warn the operators in the plants about situations that could hamper the machines operation or stop the production process. The system follows a two-stage *forecaster-analyzer* approach on which, first, a forecaster predicts the future measurements of different types of time-series data captured by the sensors implanted on a real extruder machine; and then, distinct analyzers (one for each type of alarm to predict) determine if the predicted measurements will trigger an alarm or not. An early prediction of those alarms can bring several benefits to manufacturing companies, such as predictive maintenance of the equipment, or production optimization.

1.4 Dissertation Outline

Chapter 2 provides, a detailed background of the context of Smart Manufacturing together with the main key enabling technologies supporting it and various public and private initiatives worldwide promoting its development and adoption. This chapter also details the role played by data-driven and servitization approaches in Smart Manufacturing, together with the business context of a relevant agent that arise within the rise of Smart Manufacturing.

Chapter 3 details the real-world Smart Manufacturing context within this research work has been carried out, and the three main agents involved in it. The chapter also presents some of the requirements and challenges that arise from this context. Moreover, it provides an overview of the relevant research and knowledge areas examined and integrated for building the contributions of this research work, proposed as effective solutions for these challenges.

Chapter 4 presents the first contribution of this research work, i.e., *a system that efficiently guides a data engineer in the task of pre-processing raw time-series data* coming from industrial sensors. The chapter starts describing some of the problems that arise in these scenarios for the processing of massive amounts of data coming from multiple heterogeneous sensors and some of the challenges that the data engineers in charge of preprocessing those data face. Next, an overview of the considered time-series data pre-processing techniques for cleaning those data is provided, along with some recommendations on which are the most appropriate ones for the considered time series. Afterwards, the steps followed for building the machine learning-based model that recommends the most suitable time-series dimensionality reduction techniques are described; as well as some performance results are presented, together with an analysis of the built model. Then, the built system is presented along with an usage demonstration; and finally, the conclusions of the realized work are presented.

Chapter 5 presents the second contribution of this research work, i.e., *a three-level hierarchical architecture for an efficient storage of time-series data* on cloud environments. This chapter begins describing the problems related to the management of the captured time-series data in Smart Manufacturing scenarios. Then, an analysis of the state of the art regarding Time Series Management Systems is provided. Afterwards, an overview of the proposed architecture is presented; followed by some results of the performed tests using four different types of Database Management Systems under two different main dimensions: used storage space (and an estimation of the associated costs), and performance (in terms of query answering time). To end the chapter, a discussion of different aspects of the realized work is presented together with the extracted conclusions.

Chapter 6 presents the third contribution of this research work, i.e., the design and development of *a flexible alarm prediction system following a forecaster-analyzer approach*. As a motivation for this contribution, this chapter begins describing alarm prediction systems and one of the open research problems regarding the early prediction of those alarms. After an analysis of the state of the art in alarm prediction systems, the chapter continues describing the approach followed in this research work to design and build the system and a detailed description of the main components involved in it (i.e., the forecaster and the analyzers). Then, how these components are combined for predicting alarms is presented, and lastly, the conclusions of the realized work are presented.

Finally, Chapter 7 presents the global conclusions extracted after conducting this research work and some of the opportunities that arise for further research work in line with the proposed contributions.

Chapter 2

Smart Manufacturing Context and Antecedents

The fourth industrial revolution has given rise to what is called Smart Manufacturing, addressing the use of modern Information Technologies (IT) to transform, the acquired data across the product life-cycle into manufacturing intelligence in order to achieve meaningful improvements in all aspects of manufacturing. The rise of Smart Manufacturing together with the strategic initiatives carried out in different countries have promoted its adoption among manufacturers who are increasingly interested in providing not only equipment, but also value-added, data-enabled services to their customers, and therefore they are developing data-driven servitization strategies. However, the adoption of these servitization strategies faces diverse technological challenges with regard to the required data-related IT. Given the wide spectrum of technological challenges and their complexity, the adoption of these technologies by manufacturing companies aiming to shift their businesses towards Smart Manufacturing approaches, demands the support of technology suppliers specialized in Smart Manufacturing oriented IT-services (ITS Providers). This context facilitates establishing strategic partnerships between ITS Providers and manufacturing companies, aiming at developing the required smart services that allow these manufacturers to leverage the potential of Smart Manufacturing to transform their businesses or their production processes.

This research work and its contributions, are focused on a real Smart Manufacturing context given by an ITS Provider whose business model is based on supplying the required IT support and data-related smart services for manufacturing companies, including Capital Equipment Manufacturers (CEMs) developing a data-driven servitization approach. Such a business context emerges as a consequence of the evolution of the main key enabling technologies supporting Smart Manufacturing and various public and private initiatives worldwide promoting the development of the concept of Smart Manufacturing and its adoption among manufacturers. This chapter presents first, the technological background supporting Smart Manufacturing; and then, the context of Smart Manufacturing, together with the two main approaches that have led to the adoption of Smart Manufacturing among manufacturers; and the business context of a key agent in the adoption of these approaches, the ITS Providers.

2.1 Technological Background

The services deployed by ITS Providers supplying the required IT support for manufacturing companies are based on three of the main key enabling technologies supporting Smart Manufacturing: *Big Data* [GBB18], the *Internet of Things* (IoT) [Eva11] and *Cloud Computing* [ZZCW10]. They deploy *Industrial IoT (IIoT)* devices that connect to the low-level IT infrastructure operating in manufacturing plants to capture massive amounts of data generated by industrial sensors regarding different magnitudes or indicators of interest. These captured data (time series generated by the continuous operation of the manufacturing process or equipment being analyzed) are usually transmitted to a Cloud Computing environment, where diverse processing functionalities on the data are supported. The massive data gathered can serve to extract Knowledge from the data *(Knowledge Discovery)* and perform advanced *Data Analytics* for different purposes, such as product quality or process efficiency control, fault diagnosis, predictive maintenance of equipment, etc. This will depend on the specific manufacturing business sector where the company operates and on the specificities of the servitization strategy and data-enabled services it wants to provide. In the following subsections, details about these enabling technologies are given.

2.1.1 Industrial Internet of Things

Since the development of the first prototype of the Internet (for fault-tolerant communications via interconnected computer networks) in the ARPANET project funded by the US Department of Defense in 1960, the world has witnessed an exponential growth in the developments of content materials in the Internet. In the 1990s, Internet began to provide more and more services to particular and business users; and since the 2000s social networks have been facilitating the interconnectivity among billions of people. Most recently, the interconnectivity has been extended not only to people but also to everyday objects and thus, there has been a shift from the internet of people to the Internet of Things (IoT) [YKBT19].

The origins of the concept of IoT traces back to 1999, when Kevin Ashton first uses it at the MIT Auto-ID Laboratory [A$^+$09], for referring to an "Internet" composed of a large numbers of interconnected physical devices or "Things," such as sensors, actuators, smart applications, computing devices, machines, people, objects, etc. Thanks to rapid advances in underlying technologies, IoT has brought tremendous opportunities for a large number of novel applications that promise to improve the quality of people lives. Consequently, in recent years, IoT has gained much attention from researchers and practitioners from different application domains, and the number of "smart" devices connected to the Internet is growing exponentially. In fact, according to Cisco's Annual Internet Report [Cis20], the number of connected devices will grow from 18.4 billion in 2018 to 29.3 billion by 2023 (see Figure 2.1).

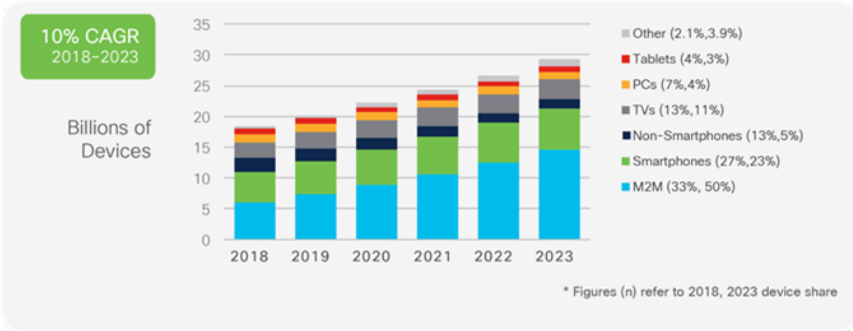


FIGURE 2.1: Global device and connection growth (extracted from [Cis20])

With the rapid advances in underlying technologies, devices have been equipped with different type of sensors and actuators and are able to connect and communicate over the Internet. This way, the IoT connects real-world objects and embeds the extracted knowledge from them in a whole system to smartly process the object specific information and take useful autonomous decisions [KKZK12]. The basic simplified workflow of IoT can be divided in three main tasks: *All-out Perception*, *Reliable Transmission* and *Intelligent Operation*. *All-out Perception* refers to the process of sensing objects or "Things" and collect information from them; *Reliable Transmission* refers to have "Things" connected and enabling them to exchange reliable information through communication networks; and *Intelligent operation* refers to the analysis of the data through various intelligent computing technologies to extract knowledge from it and realize a smart control and an intelligent decision-making [YYY+11].

The aforementioned workflow and the entailing tasks have been materialized in a three-layered model, coining one of the most popular architectures for IoT (see Figure 2.2 (a)), on which each task has its corresponding layer in the architecture:

- *The perception layer* (a.k.a *device layer*) is the physical layer, which has objects or sensors for sensing and gathering information about the environment. This layer is in charge of the identification of the objects or sensors and the collection of their information.

- *The network layer* (a.k.a *transmission layer*) is responsible of connecting sensors, servers and other network devices and transferring data among them.

- *The application layer* delivers the specific application services to the end-users. The applications implemented by IoT can be multiple, across different domains of application: Smart Manufacturing, health, smart city, intelligent transportation, etc.

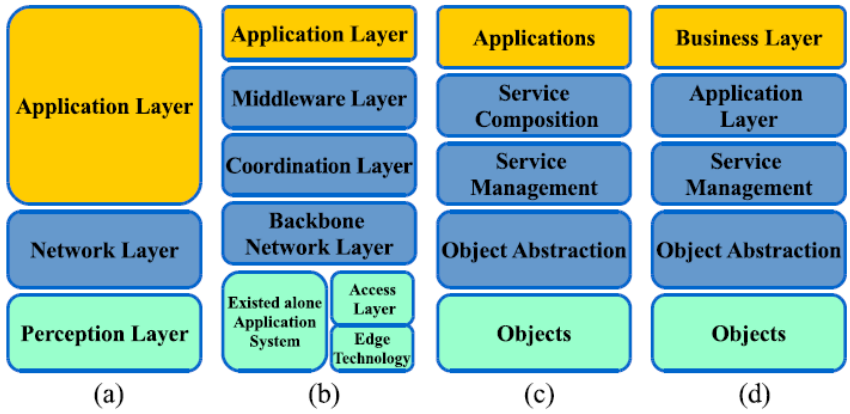


FIGURE 2.2: IoT architectures: (a) Three-layer. (b) Middle-ware based. (c) SOA based. (d) Five-layer. (extracted from [AFGM+15])

However, although that three-layer architecture has been extensively studied and used by different researchers and practitioners [WLL+10], [YYY+11], [SS17], it was not sufficient for further research which often focuses on finer aspects of the Internet of Things. Therefore, more layered architectures have been proposed in the literature [SS17], where the ever increasing number of proposed architectures has not yet converged to a reference model [KPC14]. Figure 2.2 shows some other architectures extracted from the literature, among which the five-layer model (Figure 2.2 (d)) has been one of the most popular IoT architectures in recent years. The five-layer architecture which additionally includes the *processing* and *business* layers, keeps the role of the *perception* and *application* layers from the three-level architecture, including the following roles for the remaining layers:

- *The transport layer* is in charge of transferring the sensor data from the perception layer to the processing layer and vice versa.
- *The processing layer* (a.k.a middle-ware layer) is responsible of storing, analyzing, and processing huge amounts of data that comes from the transport layer. It can manage and provide a diverse set of services to the lower layers. It performs information processing and ubiquitous computation and takes automatic decision based on the results.
- *The business layer* is responsible for the management of overall IoT system including the applications and services. It builds business models, graphs, flowcharts, etc. based on the data received from the *Application* layer.

The different layers of these architectures, encompasses several technologies that integrate these *"Things"* into a global network of Internet of Things. A review of them can be found in [AFGM+15] and [CH18].

Nevertheless, there exist some technologies whose development has supposed a major breakthrough for the IoT and thus, they are presented as key enabling technologies for the IoT in different works [GBMP13]. These key enabling technologies include among others the following:

- *RFID:* Radio Frequency Identification has been a major breakthrough in the embedded communication paradigm which enables the design of microchips for wireless data communication. RFID is one of the key technologies for making the objects uniquely identifiable. Its reduced size and cost makes it easily integrable into any object. The basic components of RFID technology include: RFID transponders (tags), RFID transceivers (readers), and data processing IT infrastructure.

- *WSN:* Recent technological advances in wireless communications and low power integrated circuits have made available, low-power and low-cost miniature devices, for efficient remote sensing applications. This fact has improved the viability of utilizing a sensor network consisting of a large number of intelligent sensors, enabling the collection, processing, analysis and dissemination of valuable information, gathered in a variety of environments. The components that make up each WSN sensor include: a radio transceiver (i.e., a transmitter for data transmission and a receiver to receive control signals); a processing unit (i.e., a microcontroller providing an embedded computing system); a sensing unit (i.e., an analog circuit for signal sensing and processing); and a power source.

- *Addressing schemes:* The ability to uniquely identify "Things" is critical for the success of IoT. In this regard, the Uniform Resource Name (URN) system is considered fundamental for the development of IoT. URN will not only allow to uniquely identify billions of devices; but also to control remote devices through the Internet. This way the entire network forms a web of connectivity from users (high-level) to sensors (low-level) that is addressable (through URN), accessible (through URL) and controllable (through URC).

The application of the IoT technologies to the industrial world has led to coin the concept of Industrial Internet of Things, defined in [KRZ+20], as *"the network of intelligent and highly connected industrial components that are deployed to achieve high production rate with reduced operational costs through real-time monitoring, efficient management and controlling of industrial processes, assets and operational time."* In IIoT, the term *"Things"* usually referees to different assets and people from a manufacturing environment, including among others: materials, sensors, machines, actuators, controllers, material handling equipment, humans, operators, robots, etc. Although both concepts are closely related, IIoT shifts the service model of what is usually addressed as IoT (a.k.a. customer IoT) from a human-oriented service model to a machine-oriented service model where new challenges appear due to the specific requisites and demands of

the industrial world [SSH+18]. Table 2.1 provides a comparison between both concepts.

TABLE 2.1: Comparison Between Consumer IoT and IIoT (extracted from [SSH+18] and [KRZ+20])

	Consumer IoT	**Industrial IoT**
Impact	Revolution	Evolution
Service Model	Human-centered	Machine-oriented
Current Status	New devices and standards	Existing devices and standards
Connectivity	Ad-hoc (infrastructure is not tolerated)	Structured (centralized network management)
Nodes	Mobile	Fixed
Criticality	Not stringent (excluding medical applications)	Mission critical (timing, reliability, security, privacity)
Data Volume	Medium to High	High to Very High
Area of Focus	General Applications	Industrial Applications
Focus Development	Smart Devices	Industrial Systems
Security and Risk Measures	Utility-centric	Advanced and Robust
Interoperability	Autonomous	CPS-Integrated
Scalability	Low-scale Network	Large-scale Networks
Precision and Accuracy	Critically Monitored	Synchronized with milliseconds
Programmablity	Easy Off-site programming	Remote on-site programming
Output	Convenience and Utilization	Operational Efficiency
Resilience	Not Required	Required High Fault Tolerance
Maintenance	Consumer Preferred	Scheduled and Planned

While the most general requirements and technologies supporting IoT and IIoT are similar (e.g., support of an Internet ecosystem using low-cost, resource-constrained devices and network scalability), many requirements and technologies are specific to each domain. In fact, IIoT can be seen as an application of IoT which requires higher levels of safety, security and reliable communication without the disruption of real-time industrial operations due to mission-critical industrial environments [KRZ+20]. The specific requirements and challenges that arise from such a strict scenario have leveraged to the development of specific technologies supporting IIoT; a review of them can be seen in [SSH+18] and [XYGG18]. Moreover, as well as the industrial environment requires specific technologies to develop the IoT over the industry, IoT architectures also have been adapted to this environment in order to meet its particular necessities. For example, in [XYGG18], the three-layered architecture for IoT mentioned before is adapted for IIoT and in [KRZ+20], a general architecture for developing IIoT systems (see Figure 2.3) is presented.

The adoption of those IIoT technologies in the industry allows to connect all the industrial assets sensing and monitoring the manufacturing processes to collect and analyze a large amount of data, that can be used, monetized and improve the overall performance of the systems for providing new types of services. However, given the huge amount of heterogeneous data produced by IIoT, traditional information technologies fall short for their management, and therefore the integration of IIoT technologies with other synergic technologies like, Big Data and Cloud Computing, is required for their exploitation.

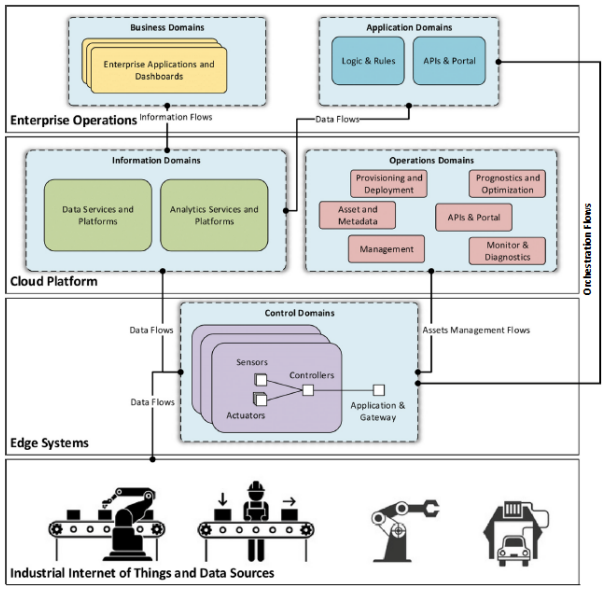


FIGURE 2.3: A General architecture for IIoT systems (extracted from [KRZ+20])

2.1.2 Big Data

Over the last decade the world has witnessed an explosive increase of global data size, that is expected to keep increasing over the years. Such is the pace of data growth, that according to the analysis published in 2012 by the International Data Corporation (IDC) [IDC12], the *"digital universe"* (defined as a measure of all the digital data created, replicated and consumed in a single year) was expected to grow from 2005 to 2020 by a factor of 300, from 130 exabytes to 40.000 exabytes; and it was expected to double its size every two years until 2020. These estimations were coherent with a more recent analysis conducted by the IDC in 2018 [RGR18], manifesting, that the *"digital universe"* (also referred as *"Global Datasphere"*) is expected to grow from 33 Zettabytes in 2018 to 175 Zettabytes by 2025 (see Figure 2.4).

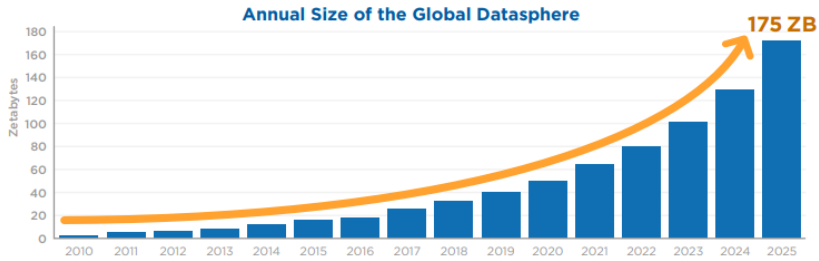


FIGURE 2.4: Expected growth of the Digital Universe by 2025 (extracted from [RGR18])

As data volumes increased, the concept of Big Data for referring to the large and diverse sets of information growing at ever-increasing rates was gaining more and more popularity; and since the early 2000s the Volume (the generation and collection of masses of data), Velocity (rapidly collected and analyzed data), and Variety (different types of data from several data sources) in data (a model later known as "3V" [Lan18]) began to be analyzed as key aspects in any strategy for an efficient data management. However, it has been along the 2010s decade when the importance of Big Data has been generally recognized and, finally, this concept was consolidated. In fact, in May 2011, McKinsey & Company, a global consulting agency announced Big Data as *"the next frontier for innovation, competition, and productivity"* [MCB+11].

At present, although the importance of Big Data has been generally recognized, no consensus has yet been reached on its definition. In general, Big Data refers to *"huge datasets that could not be perceived, acquired, managed, and processed by traditional IT and software/hardware tools within a tolerable time"* [CML14]. The mainstreaming of this concept as a technological trend has massively contributed to its popularization among researchers, practitioners and users; and thus, it constitutes one of the most relevant global trends in IT in last years.

The demand of the technologies supporting Big Data had its origin in the application of data analytics by the big technological companies aiming to boost their business by exploiting their data banks. The large volumes of data to be analyzed by these companies made unfeasible in practice their management and processing via traditional techniques. Therefore, new technologies were required to handle the huge volume and heterogeneity of Big Data.

One of the pioneers in facing this kind of problems was Google, to process their PageRank algorithm [PBMW99] efficiently when applied to massive volumes of data [LRU19]. Instead of using the existing strategies and solutions for parallel processing of big volumes of data, by using high-performance machines with a large amount of processing cores, Google opted for an alternative approach based on an efficient division of the processing of large volumes of data across a set of distributed machines (nodes in a cluster). This approach was built upon two main elements: a distributed file system (the Google File System [GGL03]) for large-scale data storage management in a partitioned and replicated way across the set of distributed machines; and a software solution providing efficient implementations for the most complex tasks to be executed by those distributed applications dealing with the processing of large-scale data stored in such a system. This software and, by extension the programming model enabled by it, received the name of MapReduce [DG08], marking the main milestone in the origin of Big Data technologies [Niñ17].

The strategy followed by Google served as inspiration for other projects aiming to solve similar problems. Following the aforementioned approach, Yahoo build the open-source system called Apache Hadoop [Apa19] to process efficiently the enormous volumes of data required by their search engine, in a distributed way. In a similar manner to Google's solution

Apache Hadoop was built upon two main elements that can be seen in Figure 2.5: a Distributed File System (HDFS) [SKRC10] and the Hadoop Map Reduce framework that was developed upon the HDFS. The open-source nature of Apache Hadoop encouraged the development of additional tools around this platform which boosted its utility and the subsequent emergence of alternative platforms such as Apache Spark [ZCF+10].

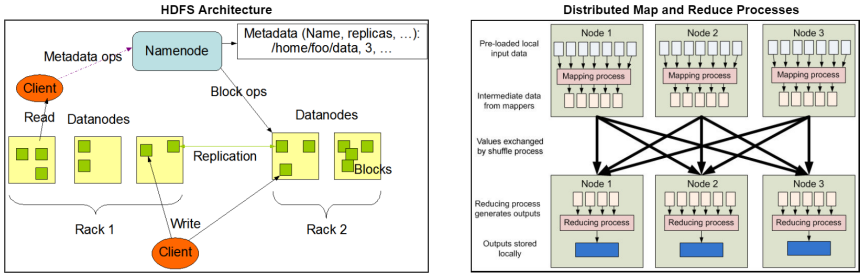


FIGURE 2.5: HDFS and MapReduce programming model (extracted from [PBN12])

The availability of open-source solutions, together with the progressive development of other synergic technologies supporting Big Data, has facilitated the increasing adoption of Big Data technologies. On the one hand, the progressive development of Cloud Computing technologies (see Section 2.1.3), facilitated the access to large clusters of machines in a dynamic-renting mode that enabled, Big Data systems developers to be equipped with the required infrastructure to store and process large-scale volumes of data in a more affordable way. On the other hand, the proliferation of IoT devices (see Section 2.1.1) capturing and sharing vast amounts of data over the Internet, opened the possibility for many different application fields to centralize their data in Cloud Computing systems for further exploitation purposes.

In addition to the key enabling technologies supporting Big Data, the adoption of Big Data technologies was also facilitated by various conceptual proposals guiding the design of Big Data systems and their integration within the existing technologies. Among these proposals it is worth highlighting the importance of two of them: the notion of Data Lake [O'L14], describing an approach for the centralized storage of the heterogeneous data (structured, semi-structured or unstructured) coming from diverse sources; and the Lambda Architecture [MW15], a design pattern for Big Data systems aiming at reducing their complexity and increasing fault tolerance.

Given that usually Big Data systems should support diverse analytical exploitation functionalities for different purposes, and considering that those functionalities are typically not characterized in detail beforehand, the captured data from heterogeneous sources should be stored without having applied any transformation or processing to make those data fit any particular structure (i.e., in their raw format). This approach would allow to accumulate the data in a Data Lake on which different data views

could be generated *a posteriori* (by applying the appropriate transformations to the data) to meet the specific necessities required by different data analytics purposes. On top of the accumulated data, the Lambda architecture (outlined in Figure 2.6) defines different layers to organize the system and its components that provide different data-driven services: a batch layer that pre-calculates the required operations on the master dataset (i.e., the Data Lake) to generate elaborated and transformed data views for different exploitation purposes; a service layer that provides efficient access to those batch views, for resolving queries with low response times; and a speed layer that resolves the incremental processing of real-time data (and the queries requiring them) as long as they have not been stored yet in the master dataset from which batch views are generated [Niñ17].

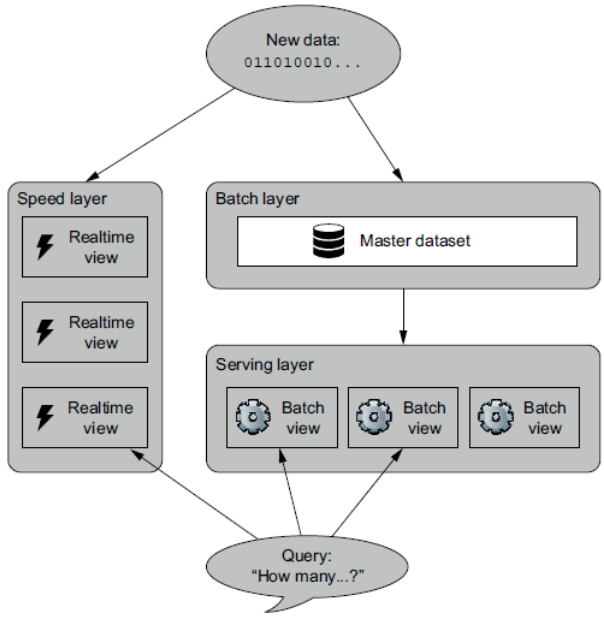


FIGURE 2.6: Lambda Architecture diagram (extracted from [MW15])

Once the basic Big Data technologies were developed, their progressive adoption and development, in parallel with the evolution of the sinergic technologies have led to an ecosystem of numerous tools that have been increasingly adopted in diverse industries. In fact, Big Data constitutes one of the key enabling technologies in Smart Manufacturing, where manufacturing companies aim at leveraging the potential of Big Data Analytics [NLMBC17] to explode the massive amounts of data generated by thousands of IIoT devices, implanted throughout their production processes, over different nodes deployed in Cloud Computing systems.

2.1.3 Cloud Computing

With the rapid development of processing and storage technologies and the success of the Internet, computing resources have become cheaper, more powerful and more ubiquitously available than ever before. This technological trend has enabled the realization of a new computing model called Cloud Computing, in which resources are provided as general utilities that can be leased and released by users through the Internet in an on-demand fashion [ZCB10].

The main idea behind Cloud Computing was not a new one. In fact, it was in the 1960s when John McCarthy envisioned that computing facilities will be provided to the general public like a utility; an idea later reinforced by Douglas Parkhill, which predicts that the computer industry would come to resemble a public utility *"in which many remotely located users are connected via communication links to a central computing facility"* [Par66]. Since then, Cloud Computing has evolved from previous computing paradigms since the first attempts at virtualizing mainframe operating systems by IBM in the late 1960s; to the late 1990s, when the pioneer Application Service Providers (ASPs) like Salesforce start offering enterprise applications via websites, and the early 2000s, when Amazon Web Services provide a suite of cloud-based services, including storage and computation [Pal10]. However, it was after Google's CEO Eric Schmidt used the word "cloud" to describe the business model of providing services across the Internet in 2006, when the term really started to gain popularity [ZCB10].

The popularization of this term led to different perceptions of what Cloud Computing was, mainly due to the fact that unlike other technical terms, it is not a new technology, but rather a new operation model that brings together a set of existing technologies to run business in a different way. Indeed, most of the technologies used by Cloud Computing, such as virtualization and utility-based pricing, were not new. Instead, Cloud Computing leverages these existing technologies to meet the technological and economic requirements of today's demand for information technology. However, several efforts have been made to provide and standardize the definition of this concept, among which one of the most notable is the definition of Cloud Computing given by The National Institute of Standards and Technology (NIST) that covers, the essential aspects of Cloud Computing [MG11]: *"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."*

Cloud Computing leverages some existing technologies such as virtualization and automatic resource provisioning, to achieve the goal of providing, as a utility, distributed computing resources. Thus, Cloud Computing has been often compared to synergic technologies with which shares certain aspects [ZZCW10]:

- *Grid Computing:* A distributed computing paradigm on which networked resources are coordinated to achieve a common computational objective. In a similar manner, Cloud Computing employs distributed resources to achieve application-level objectives. However, Cloud Computing takes one step further by leveraging virtualization technologies to realize resource sharing and dynamic provisioning at multiple levels (hardware and application platform).

- *Utility Computing:* Cloud Computing can be perceived as a realization of utility computing; a model of providing resources on-demand and charging customers based on usage rather than a flat rate. It adopts a utility-based pricing scheme entirely for economic reasons: with on-demand resource provisioning and utility based pricing, service providers can truly maximize resource utilization and minimize their operating costs.

- *Virtualization:* A technology that allows the creation of a virtual version of some technological resource through software. In Cloud Computing Virtualization provides the capability of pooling computing resources from clusters of servers and dynamically assigning or reassigning virtual resources to applications on-demand.

- *Autonomic Computing:* With the goal of overcoming the management complexity of today's computer systems, Automatic Computing aims at building self-management computing systems that are capable of adapting to unpredictable changes without human intervention. In this sense, Cloud Computing exhibits certain autonomic features, such as automatic resource provisioning; however, its objective is to lower the resource cost, rather than reducing system complexity.

The rapid advances in the synergic technologies supporting Cloud Computing have leveraged to an increasing adoption of the Cloud Computing model that has revolutionized the way on which companies providing Information Technology Services (ITS Providers) design their supply of IT services an business applications. These companies have evolved towards a service-driven business model on which not only the means of providing IT services has changed, but also the nature of the provided services has also suffered a significant transformation [Dha12]. In such a business model, hardware and platform-level resources are provided as services on an on-demand basis that can be grouped into three main categories:

- *Infrastructure as a Service (IaaS):* IaaS refers to on-demand provisioning of infrastructural resources, such as storage space, computing power, networks and other fundamental computing resources. IaaS Providers supply those resources to customers who want to deploy and run different applications on that infrastructure. In order to integrate/decompose physical resources in an ad-hoc manner to meet growing or shrinking resource demand from cloud consumers, *virtualization* is extensively used in IaaS cloud. The basic strategy of

virtualization is to set up independent Virtual Machines (VM) that are isolated from both the underlying hardware and other VMs.

- *Platform as a Service (PaaS):* PaaS (a.k.a *cloudware*) refers to providing platform layer resources, including operating system support and software development frameworks. PaaS providers offer to their customers a development platform supporting the full "Software Lifecycle" which not only cover the essential technical resources; but also, some essential application services that allows cloud consumers to develop and run cloud services and applications directly on the PaaS cloud.

- *Software as a Service (SaaS):* SaaS refers to providing on demand applications over the Internet. Saas providers provide fully developed, purpose-specific solutions to end users. SaaS is aligned with the Application Service Provider (ASP) model; the hosted application management model of SaaS providers is similar to the ASPs model, where the provider develops and hosts the software which is it delivered to end users over the Internet.

Those categories fits to the services offered by the different layers on which generally speaking, the architecture of a Cloud Computing environment can be divided: the hardware/data center layer, the infrastructure layer, the platform layer and the application layer. Each layer is loosely coupled with the layers above and below, allowing each layer to evolve separately giving more modularity to the architecture. This modularity, allows cloud computing to support a wide range of application requirements while reducing management and maintenance overhead. Figure 2.7 shows a general layered architecture for Cloud Computing environments, and next, the layers involved in it are described:

- *The hardware layer* is responsible of managing the physical resources of the cloud, including among others, physical servers, routers, switches, power and cooling systems. In practice, this layer is typically implemented in data centers containing thousands of servers that are usually organized in racks and interconnected through switches, routers or other fabrics.

- *The infrastructure layer* (a.k.a the *virtualization layer*) creates a pool of storage and computing resources by partitioning the physical resources using virtualization technologies. These technologies enable some of the key features of Cloud Computing, such as dynamic resource assignment.

- *The platform layer* consists of operating systems and application frameworks that are built on top of the infrastructure layer. The purpose of this layer is to minimize the burden of deploying applications directly into VM containers.

- *The application layer* consists of the actual cloud applications developed at the highest level of the hierarchy. In contrast to traditional

applications, cloud applications can leverage the automatic-scaling feature to achieve better performance, availability and lower operating cost.

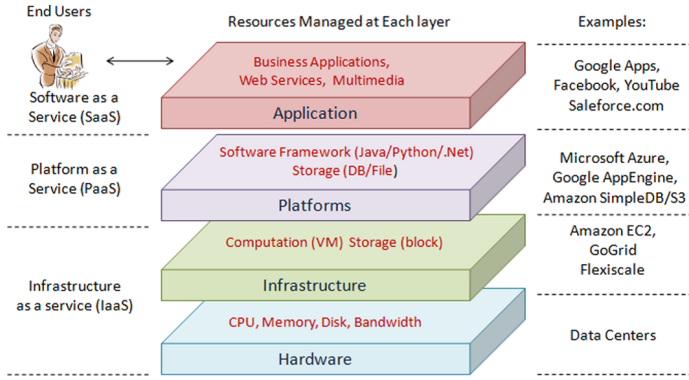


FIGURE 2.7: Cloud Computing architecture (extracted from [ZCB10])

Despite the increasing popularity among companies of this service-driven business model, there are many issues to consider when moving an enterprise application to the cloud environment. For example, some service providers could be more interested in lowering operation cost, while others may prefer high reliability and security. Therefore, different types of cloud deployment have been defined in the Cloud Computing community [ZCB10], [DWC10]:

- *Public Clouds:* In this type of cloud, which is the dominant form of the current Cloud Computing deployment model, service providers offer their resources as services to the general public. The cloud service provider has full ownership of the public cloud with its own policy, value, and profit, cost, and charge model. This allows risk free rapid developments by the public cloud customers without requiring any capital investment in infrastructure (which is assumed by the provider) at the cost of losing control over the data, network and security settings, which could hampers their effectiveness in some business scenarios.

- *Private Clouds:* This type of clouds are designed to be used exclusively by a single organization. The cloud infrastructure is managed by the organization or by external providers. In contrast to public clouds, a private cloud offers higher degree of control over the resources and their performance, reliability and security; with the main drawback that it requires organizations face up-front capital costs. Additionally, several organizations could jointly construct and share cloud infrastructure as well as policies, requirements, values, and concerns in a *Community cloud*.

- *Hybrid Clouds:* In order to address the limitations of public and private clouds, a hybrid cloud is presented as a combination of them,

on which part of the service infrastructure runs in private clouds and the remaining part runs in public clouds. In the same way as public clouds, hybrid clouds facilitate on-demand service expansion and contraction; while they provide tight control and security over application data, characteristic of private clouds.

- *Virtual Private Clouds (VPC):* This type of cloud is essentially a platform running on top of public clouds. A VPC allows service providers to design their own topology and security settings (such as firewall rules) over public clouds, by using Virtual Private Networks (VPN) technologies. It can be seen as a more holistic alternative than public and private clouds since in addition to virtualizing servers and applications, it also virtualizes the underlying communication network.

For most service providers, selecting the right cloud model is dependent on the business scenario. However, regardless of the business scenario and the adopted cloud type, Cloud Computing is changing the business model of industries and enterprises towards a new paradigm in which virtualized resources are provided as a service over the Internet in a dynamically scalable manner. As a consequence, Cloud Computing has an enormous impact in the global economy [BCW+11]; for example the Worldwide Public Cloud Revenue was 227.8 billion in 2019 and according to the forecast made by Gartner, is expected to grow up to 354.6 billion of U.S dollars by 2022 (see Figure 2.8).

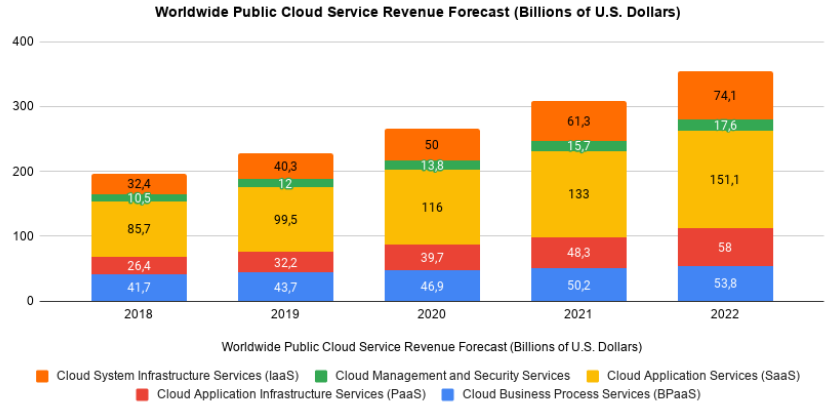


FIGURE 2.8: Worldwide Public Cloud Service revenue (generated from the data extracted from [Gar19])

Lastly, with regard to the industrial context, Cloud Computing is emerging as one of the key enablers; it allows to process and manage big volumes of manufacturing data captured by IIoT devices to extract knowledge from them and improve the overall manufacturing process. Cloud Computing can revolutionize the traditional manufacturing business model, helping it to coordinate product innovation with market strategy,

and creating interconnected smart factories that encourage effective collaboration. The adoption of Cloud Computing technologies in the manufacturing sector has given rise to coin the concept of Cloud Manufacturing, the manufacturing version of Cloud Computing [Xu12].

2.1.4 Knowledge Discovery and Data Analytics

Traditional methods for turning data into knowledge consist on manual analysis and interpretation of them. Regardless the field or the application domain, in this classical approach, the data analysis relies fundamentally on one or more analysts becoming intimately familiar with the data and serving as an interface between the data and the users and products. This form of analyzing a dataset results slow, expensive, and highly subjective. Moreover, with the dramatically growth of data volumes, this approach become completely impractical in many domains and therefore, as some authors already pointed out, more automatic approaches were required. For example, in 1996 Fayyad already stated that *"There is an urgent need for a new generation of computational theories and tools to assist humans in extracting useful information (knowledge) from the rapidly growing volumes of digital data"* [FPSS96].

The need to automatize and scale up human analysis capabilities to handling the large volumes of data was both economic and scientific. From the business perspective, data analytics could allow to gain competitive advantage, increase efficiency, and provide more valuable services to customers. On the other hand, from the scientific point of view, the captured data about the environment could serve as the basic evidences to build theories and models for diverse analysis purposes in different application domains. Those analysis would be focused on extracting knowledge from data in the form of patterns, models or trends that provided useful information about the outcome of potential future actions.

In order to refer to this type of analysis, at the end of the 80s the Data Mining term was coined. This term origins from the analogy with mining techniques, where a valuable material (in this case, knowledge) is extracted from mining deposits (data repositories in this case). Historically, the notion of finding useful patterns in data has been given a variety of names, including data mining, knowledge extraction, information discovery, etc. However, the term data mining was one of the most known and extended term to refer to this type of analysis [HPK11]. This term has been mostly used by statisticians, data analysts, and the Management Information Systems (MIS) communities and it has also gained popularity in the databases field.

Around the same time, the expression Knowledge Discovery in Databases was coined at the first KDD workshop in 1989 [PS90] to emphasize that knowledge is the end product of a data-driven discovery. Although in many occasions both terms were used interchangeably, KDD refers to the overall process of discovering useful knowledge from data, whereas data mining constitutes a particular step in this process in which different algorithms are applied to extract patterns from the data. This

workshop, later led to the *First International Conference on Knowledge Discovery and Data Mining* (KDDM) in 1995 [FU95]. These KDDM related workshops and conferences gave rise to one of the most popular definition of KDD as *"the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data"* and its foundational schema (outlined in Figure 2.9), by the academicians organizing them.

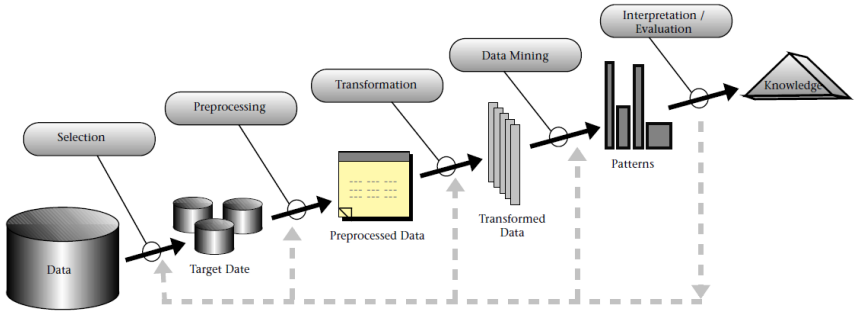


FIGURE 2.9: An overview of the steps that compose the KDD Process (extracted from [FPSS96])

The KDD process involves numerous steps with many decisions made by the user in an interactive and iterative way. Table 2.2 summarizes some of the basic steps of the KDD process. After that, various additional KDD process models and methodologies have been proposed [KM06]; among which, the CRoss-Industry Standard Process for Data Mining (CRISP-DM) reference model [She00] (see Figure 2.10) stands out given its acceptance among KDDM practitioners. CRISP-DM is the most widely used methodology for developing data mining projects. Indeed, it is cited as the most often used methodology [PS14] and it is considered the *de facto* standard to manage data mining projects [MMF10].

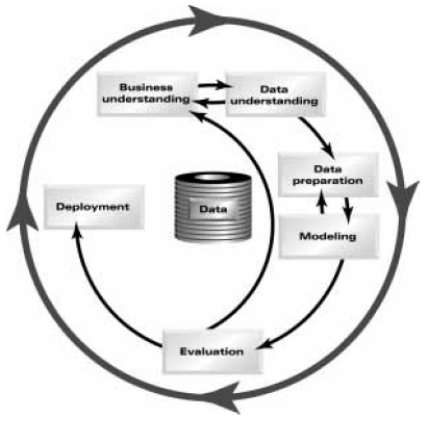


FIGURE 2.10: Phases of the CRISP-DM reference model (extracted from [She00])

TABLE 2.2: Steps involved in the KDD Process (extracted from [FPSS96] and [Fay96])

Step	Description
1. Understanding application domain	Developing an understanding of the application domain and the relevant prior knowledge and identifying the goal of the KDD process.
2. Creating a target dataset	Selecting a dataset, or focusing on a subset of variables or data samples, on which knowledge discovery is to be performed.
3. Data cleaning and pre-processing	Basic operations such as noise and outliers treatment, collecting the necessary information to model or account for noise, handling missing data, accounting for time-sequence information, known changes, data normalization, etc.
4. Data reduction and transformation	Finding useful features to represent the data depending on the goal of the task by using dimensionality reduction or transformation methods.
5. Choosing the data mining task.	This involves deciding whether the goal of the KDD process is; classification, regression, clustering, summarization, or dependency modeling.
6. Exploratory analysis and data mining algorithm selection	Choosing the data mining algorithm(s) and selecting method(s) to be used for searching for data patterns. Deciding which models and parameters might be appropriate and matching a particular data mining method with the overall criteria of the KDD process (e.g., prioritizing the model's over predictive capabilities).
7. Data mining	Searching for patterns of interest in a particular representational form or a set of such representations, including classification rules or trees, regression, and clustering.
8. Interpreting mined patterns	Interpreting mined patterns, possibly returning to any of steps 1 through 7 for further iteration. This step can also involve visualization of the extracted patterns and models or visualization of the data given the extracted models.
9. Acting on the discovered knowledge	Using the knowledge directly, incorporating the knowledge into another system for further action, or simply documenting it and reporting it to interested parties.

Since then, KDD has evolved, and continues evolving from the intersection of different research fields (see Figure 2.11), such as machine learning, pattern recognition, databases, statistics, artificial intelligence, knowledge acquisition for expert systems, data visualization, and high-performance computing into a process on which the unifying goal is extracting high-level knowledge from low-level data in the context of large datasets. Moreover, some of these fields provide the data mining methods that are used in the data mining step of the KDD process. The continuous evolution of KDD and recent advances in the field have led to its adoption in different application domains to boost multi-purpose, data-driven applications. Among the different application domains such as health, finances, transportation, etc., one of the most interesting application domains to boost data-driven applications is the manufacturing industry, due to the massive volumes of data generated by IIoT devices monitoring the whole manufacturing process.

In the manufacturing context, many industrial processes are now automatized and computerized, capturing vast amounts of data from the manufacturing process. The collected data from the manufacturing chain contains valuable information and knowledge that could be integrated into manufacturing systems to help decision making and enhance productivity. However, the massive volumes of manufacturing data, which contain large numbers of records, with many attributes that need to be concurrently explored to extract knowledge and useful information, make manual analysis unfeasible. All these factors, make crucial the use of more intelligent and automatized approaches, such as the data analysis methodologies and tools from KDD and data mining [CHT09]. Moreover, the advances in IoT, Cloud Computing, Big Data, and KDDM-related technologies (such

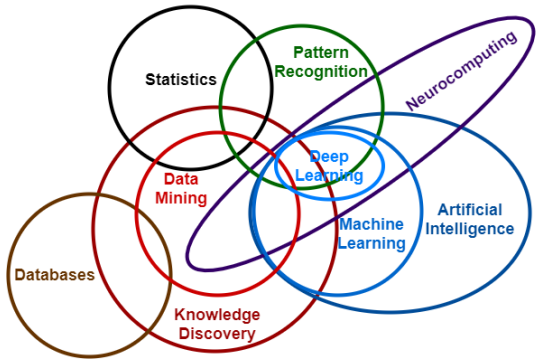


FIGURE 2.11: Representation of the intersection between KDD-related fields (adapted from [SR18] and [AM19])

as artificial intelligence and machine learning) have profoundly impacted manufacturing; driven the development of Smart Manufacturing [TQLK18] which aims to transform the collected data across the product life-cycle into manufacturing intelligence in order to achieve meaningful improvements in all aspects of manufacturing.

2.2 Smart Manufacturing Context

The adoption of knowledge discovery and data mining technologies in the industry is not particularly recent [HSSK05]. However, the global increasing interest in technologies that (a) handle large volumes of data, known as Big Data; (b) provide, as utility, distributed computing resources, known as Cloud Computing; and (c) make the Internet of Things possible; has given rise to the resurgence of data analytics and its mainstreaming among researchers and practitioners from many application domains including the manufacturing industry. Moreover, the resurgence of data analytics has motivated a new wave of interest in boosting data analytics applications as a driver for a new revolution of the industrial sector and a cornerstone of new strategies for the competitiveness of the manufacturing industry around the world. In particular, it has led to the emergence of Smart Manufacturing and the adoption of manufacturing servitization strategies through the creation of data-driven smart services.

This section details: first, the emergence of Smart Manufacturing together with various public and private initiatives and policies promoting its adoption among manufacturers, given rise to the popularization of Smart manufacturing; then, the role that data play in Smart Manufacturing as a key enabler; and finally, how manufacturing companies are shifting their business model towards servitization, in order to provide data-driven smart services, together with one of the key agents enabling this transformation.

2.2.1 The Emergence of Smart Manufacturing

Historically the world has witnessed three industrial revolutions that have led to paradigm changes in the domain of manufacturing [TWW17]: mechanization through water and steam power in the first one, massive production in assembly lines in the second one, and automation using electronics and information technologies in the third one (see Figure 2.12). Nevertheless, over the past years industries together with researchers and public and private policies and initiatives worldwide have increasingly advocated an upcoming fourth industrial revolution, which is expected to be capable of driving fundamental changes in the industry sector in a similar way to the other industrial revolutions.

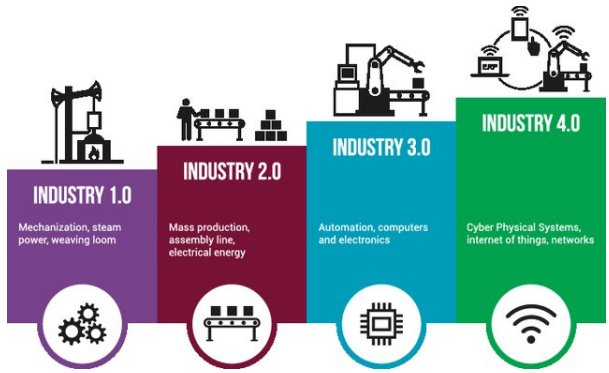


FIGURE 2.12: Highlights of industrial revolutions (extracted from [KHDK18])

The manufacturing of the modern era dates back to the middle of the last century [Kus18]. Since then, the progressive evolution of computer and machine-building technology has led to the automation of manufacturing. Depending on the scope and degree of automation of a manufacturing plant, different terms have been used to describe automated manufacturing since 1980s, including among others, computer-integrated manufacturing and intelligent manufacturing [Kus18]. The last term was coined around 1990 and marked with the establishment of the *Journal of Intelligent Manufacturing* [Kus90] and the publication of the book *Intelligent Manufacturing Systems* [Kus].

Around the same time, Japan was embarked on the research in intelligent manufacturing that led to the establishment of the Intelligent Manufacturing System (IMS) Programme [Par98] in 1995. However, it realised that the industry of a single country could not reshape manufacturing and that international cooperation was needed, and therefore, Japan, United States, Korea and European countries initiated collaborative efforts on the future of manufacturing. In the United States, much of the IMS activities were developed within the Next Generation Manufacturing Systems (NGMS) Programme [Kus18]; and in Europe, the research efforts in intelligent manufacturing established by the European Union expanded the IMS Programme [Gro95].

In addition to the aforementioned program, many strategies and initiatives have been developed by both the public and private sectors under different names such as Industry 4.0, Industrial Internet of Things, Advanced Manufacturing, etc. Figure 2.13 shows some of the main initiatives worldwide for the fourth industrial revolution, and next some of the most relevant ones are overviewed:

- *Advanced Manufacturing (US):* The President's Council of Advisors on Science and Technology (PCAST) presented in June 2011 the *"Ensuring American Leadership in Advanced Manufacturing"* report [Pre11], recommending the launching of an innovation policy in order to ensure the strategic development of the manufacturing industry. This policy was built on the concept of an *Advanced Manufacturing* on which new emergent technologies are used to transform the whole manufacturing process. The recommendations collected in this report led to the establishment of the Interagency working group on Advanced Manufacturing (IAM) who developed the report *"A National Strategic Plan for Advanced Manufacturing"* [Nat12]. The efforts derived from those actions led to the National Network for Manufacturing Innovation (NNMI) Program and to the Advanced Manufacturing Partnership (AMP), to revitalize US manufacturing through an Industry-Academia-Government partnership, which released the *"Accelerating U.S. Advanced Manufacturing"* report. Moreover the inter-agency Advanced Manufacturing National Program Office (AMNPO) [ANM] publishes the annual report on manufacturing USA and the annual strategic plan.

- *Industrial Internet (US):* Around the same time, the concept of *Industrial Internet* started being promoted and developed by major US corporations, leaded by General Electrics, who published the report *"Industrial Internet: Pushing the Boundaries of Minds and Machines"* [EA12b]. This report gave rise to the concept of Industrial Internet as the strategic use of new emergent technologies related to data analysis and connectivity, and their application to the industrial equipment. The work carried out under the development of this strategy led to the foundation of the Industrial Internet Consortium (IIC) that has been collecting its efforts in different contributions such as technical reports, publications, reference architectures and frameworks, etc. [IIC].

- *Industrie 4.0 (Germany):* Germany's Industrie 4.0 (Industry 4.0) has been one of the most popular initiatives launched in Europe. The notion of Industrie 4.0 was coined in 2011 as a strategic initiative promoted by German public and private agents, to reinforce German manufacturing competitiveness through a progressive adoption of new emerging technologies. As a result the Industrie 4.0 Working Group was created to develop the main lines of that strategy (compiled in their final report [KWH13]) and the Plattform Industrie 4.0 (Industry 4.0 Platform) was constituted by various German

industrial associations for its further development. Among the produced outputs by this platform, the Reference Architectural Model Industrie 4.0 (RAMI 4.0) [Pla16] stands out.

- *Made in China 2025:* In 2015, the State Council launched "Made in China (MIC) 2025" [WMZ+16], an initiative that the Chinese government has set out to follow to boost and restructure its industry, so that it moves from an era of quantity to a new era of quality and efficiency in production [GPC]. MIC 2025 departs from the 2006 initiative "Strategic Emerging Industries" (SEI) focused on the adoption of advanced technologies to ensure the strategic position of emerging industries. The "SEI Catalogue" of technologies includes, among others; artificial intelligence, cyber-security services, integrated circuits and network equipment and software. However, MIC 2025 is broader in scope, focused on *"the entire manufacturing process rather than only technical innovations, promoting traditional industries and services and introducing specific measures for innovation, quality, intelligent manufacturing, and green production"* [ISD].

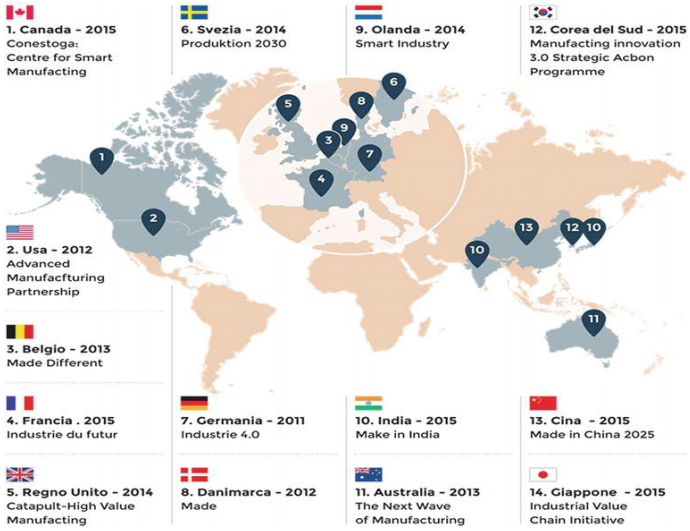


FIGURE 2.13: Main initiatives for the fourth industrial revolution worldwide (extracted from [PFCZ18])

The different initiatives present slight differences among them, for example, the Industrie 4.0 initiative put its strategic focus on "integrating information, communication and manufacturing technologies in smart, self-organizing factories," while US's focus (and increasingly also China's) is on "smart products, large Internet-based platform ecosystems and the new data-driven business models that are based on them" [Niñ17]. However, notwithstanding the names discrepancy, and superficial differences, they all share the same idea that matches the view of the National Institute of Standards and Technology (NIST) which defines Smart Manufacturing

as a *"fully integrated, collaborative manufacturing system that respond in real time to meet changing demands and conditions in the factory, in the supply network and in customer needs,"* [NIS17].

With regard to the concept of Smart Manufacturing, it was a few years later, in April 2008, when the Smart Process Manufacturing (SPM) workshop, comprised of academic researchers and industrial practitioners in process systems engineering, was held. As a result of the work carried out by those attending this workshop, the technical report named *"Smart Process Manufacturing: an Operations and Technology Roadmap"* was published in 2009, where the concept of Smart Manufacturing was firstly analyzed in detail. In September 2010 the work carried out in this workshop was extended with a new workshop named *"Implementing 21st Century Smart Manufacturing workshop"* with the objective of identifying and prioritizing the required actions to overcome some of the challenges that Smart Manufacturing presents. The information generated during the workshop was collected in a technical report [Sma11] with the same name. Moreover, this workshop also lead to the constitution of the *Smart Manufacturing Leadership Coalition* (SMLC) by some of the attendances.

The aforementioned technical report provides an overview of Smart Manufacturing. It also details the goals and the vision for the ideal Smart Manufacturing company, based on a data capture that would allow virtual tracking of all aspects of a manufacturing environment. The integration and exploitation of the captured data throughout the entire product life-cycle would lead to innovative and intelligent manufacturing environments where the manufacturing process are flexible and can react quickly to specific circumstances; optimizing overall performance and efficiency while reducing costs. Moreover, the actions needed to overcome the identified challenges in Smart Manufacturing are identified and prioritized, conforming an action plan organized into different categories, among which the creation of *"affordable industrial data collection and managements systems"* stands out. In fact, the processing of large-scale volumes of data generated during the entire product life-cycle and their processing into useful information has been identified as the key of Smart Manufacturing approaches [TQ19].

However, despite this evidence, some of the work carried out in the aforementioned workshops and reports were materialized as a journal paper [DEP+12] on which the authors reflects one of the worries that the SMLC had already strongly underscored, about *"the premise that manufacturing continues to be data rich and knowledge poor, and as a result, operates with constricted decision processes, even in operations using sophisticated modeling and control technologies."* Moreover, authors state that although some processes were being automatized and optimized through the application of information technologies, there was a lack of holistic Smart Manufacturing systems that integrate manufacturing intelligence in real-time across an entire production process.

Therefore, to help distinguishing Smart Manufacturing from the prior years of implementing information, modeling, control and optimization

technologies, advanced robotics and automation systems, etc., Smart Manufacturing is defined in [DEP+12] upon two main ideas: the compilation of manufacturing data (i.e., records of products with data about their history, state, characteristics, quality, etc.), and the application of manufacturing intelligence to those data, so that their exploitation allows manufacturers to manage, plan and predict, specific circumstances in order to optimize their production [Niñ17]. In simpler terms, Smart Manufacturing is considered in [OLBO15a] as *"the pursuit of data-driven manufacturing, where real-time data from sensors in the factory can be analysed to inform decision-making."*

2.2.2 Data-Driven Smart Manufacturing

In the last years, data have become a key enabler for improving manufacturing competitiveness worldwide, and companies have begun to recognize the strategic importance of data [TQLK18]. As a consequence, the volume of data generated by manufacturing systems is experiencing an explosive growth. Such is the growth pace that in 2015 the manufacturing industry was already generating more than 1000 EB of data annually and it is expected to increase by 20 times in the next 10 years [YK15]. However, as the authors of [OLBO15a] already envisioned, the value of big volumes of data is not just based on their accumulation, but rather on the information and knowledge that can be extracted from them [TQLK18]. Therefore, although, data-driven manufacturing can be considered as a pre-requisite for Smart Manufacturing; is the automatized computational processing and analysis of the manufacturing data which will lead to more informed decisions and a more intelligent manufacturing.

Since its inception, information technologies have driven manufacturing towards informatization. From the development of the first numerical controller milling machine in 1950, to the development of the first integrated circuits in 1960 and the adoption of Internet related technologies in 1980; the progressive adoption of information technologies in manufacturing have leveraged the advancement of computer hardware and software and its integration in manufacturing. This results in the development of many new manufacturing technologies, such as: Computer Integrated Manufacturing (CIM), Computer Aided Manufacturing (CAM), Manufacturing Execution System (MES), Enterprise Resource Planning (ERP), Supervisory Control And Data Acquisition (SCADA) software, Programmable Logic Controllers (PLCs) and Networked Manufacturing (NM), etc. Moreover, in the last years, the raise of the key enabling technologies supporting Smart Manufacturing, such as IoT, Cloud Computing, Big Data and Data Analytics; and their integration in the manufacturing environment has led to new paradigms specific for manufacturing, such as IIoT [KRZ+20], Cloud Manufacturing [Xu12], cyber-physical manufacturing systems, etc.

As a consequence of the progressive adoption of all the aforementioned information technologies (and some others) in manufacturing environments, not only the volumes of manufacturing data are increasing at an unprecedented pace; but also they are becoming increasingly richer.

Figure 2.14 shows the evolution of the manufacturing data, where the authors distinguish four data ages in manufacturing which are preceded by the different industrial revolutions. The volume of manufacturing data, together with their variety and complexity increases as time goes by through the different data ages, at the same time that they become increasingly relevant in the industrial environment for Smart Manufacturing approaches.

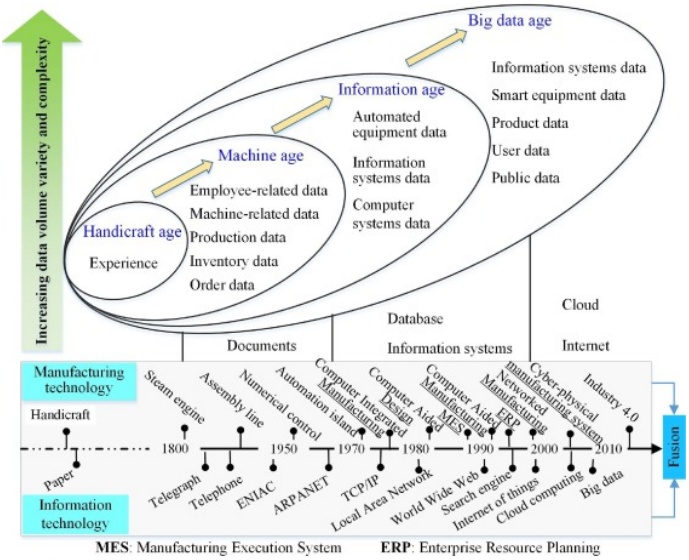


FIGURE 2.14: Manufacturing data evolution (extracted from [TQLK18])

Consequently data have been stated as a key enabler for Smart Manufacturing [TQLK18]. However, as stated before, the value of the data is not just based on their accumulation, but rather on the useful information and knowledge that can be extracted from it. In fact, as Kusiak says [Kus17]: *"Most companies do not know what to do with the data they have, let alone how to interpret them to improve their processes and products."* In its raw format the massive volumes of data generated during the manufacturing process cannot be ingested neither understood by humans. Therefore, generally, in order to extract useful information from the data, they must pass through multiple steps of a process known as the *"Lifecycle of manufacturing data"* that comprises the steps outlined in Figure 2.15, which are described next:

- *Data sources:* Manufacturing data proceed from different sources including among others IIoT devices, machines, products, operators, controllers, information systems, etc. The volume of data captured from these sources during the manufacturing process and product life-cycle is increasing at an unprecedented rate.

- *Data collection:* Data coming from different sources may be collected in distinct manners depending on the nature of the source. For example, IIoT devices allow to capture the data sensed from equipment

and products through smart sensors; RFID enables the automatic identification, management and tracking of a large number of industrial assets (e.g., products, goods, materials, etc.); the emerging mobile internet allows to capture user data from different sources (e.g., smart phones, pc, etc.); Software Development Kits (SDKs) and Application Programming Interfaces (APIs) allow to capture data from manufacturing databases and information systems, such as SCADA systems; web crawling techniques allow collecting useful information from web sites in an automatic and efficient manner, etc.

- *Data storage:* The massive volume of heterogeneous data (structured, semi-structured and unstructured) collected from manufacturing processes must be securely stored and effectively integrated. Such data type hampers the ability of traditional data management systems (usually more focused for structured data management) to handle the velocity and volume of manufacturing data. Therefore, new data management systems have been developed to deal with data with such characteristics. Moreover the deployment of those new data management systems through Cloud Computing allows to achieve a scalable and highly cost effective data storage and management in a flexible fashion manner.

- *Data processing:* In order to extract useful information and knowledge from the large volumes of manufacturing data, a series of operations must be conducted (typical of a Knowledge Discovery in Databases process). This process implies a pre-processing of the data on which data must be first cleaned (missing values imputation, handling noise and outliers, removing duplicates, etc.), and then transformed (data reduction) into meaningful and simplified datasets. Once the data have been pre-processed, they are ready to be exploited by using different data mining and analysis techniques, to extract knowledge and useful information from them.

- *Data visualization:* The extracted information by applying data mining and analysis techniques must be then communicated to the end users. For communicating the information different data visualization techniques, such as charts, diagrams, graphs and virtual or augmented reality are used with the objective of making the results of the data processing step more user-friendly (more accessible, clear and easier to understand).

- *Data transmission:* As a result of the recent advances in IoT, Internet and communication networks, distributed manufacturing data sources can be integrated almost anytime and anywhere. Consequently, data is flowing continuously among the different sources (CPSs, IIoT devices, operators etc.) and data transmission plays an important role in sustaining interactions and communications among distributed data sources.

- *Data applications:* In data-driven manufacturing, data are present in almost all the aspects of the production process and thus, the applications are numerous. A review of them can be found in [TQLK18], on where the authors, highlight, among others, the control and improvement of product quality, fault diagnosis and predictive maintenance of the equipment, etc.

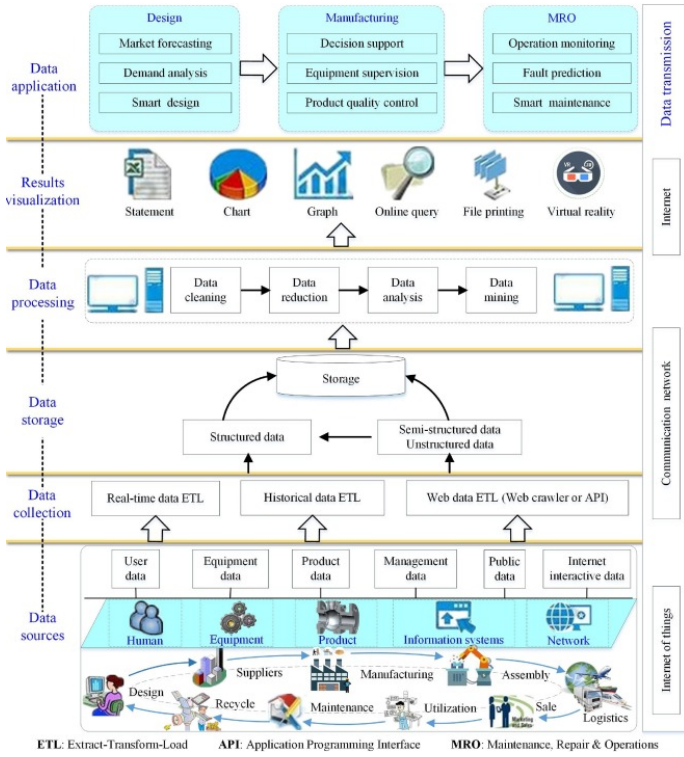


FIGURE 2.15: Manufacturing data life-cycle (extracted from [TQLK18])

During the whole life-cycle of data, manufacturing enterprises collect, store, process and analyze data for different purposes by means of Big Data and other synergic technologies such as IIoT, Cloud Computing, Artificial Intelligence, etc. Taking full advance of manufacturing data, opens the door to what is known as data-driven Smart Manufacturing, which allows to provide a full range of services to manufacturing enterprises. Those services refine the manufacturing process, shifting primary processes to smart processes that improve the flexibility and the smart level of the whole manufacturing process [TQLK18].

2.2.3 Servitization in Smart Manufacturing

The progressive adoption of the new information technologies in manufacturing environments has led to a digitalization era on which almost any aspect of the production process is digitalized, generating huge volumes of

manufacturing data. The exploitation of those manufacturing data, has paved the way to a data-driven Smart Manufacturing on which enterprises can offer a full range of highly IT-based services, also known as "Smart Services" [KRH+14]. As a consequence, many companies have shifted their business model from a product-centric model to a product-service model on which rather than products alone, companies also integrate value-added services in their products.

This strategic transformation towards a service-oriented approach on which services are also provided together with products is called "servitization." The origins of the term servitization are usually traced back to the article published in 1988 in [VR88], used to refer a trend wherein companies are offering a set of *"packages or bundles of customer-focused combinations of goods, services, support, self-service, and knowledge,"* among which services are acquiring more and more relevance [VR88].

Since its origins, the concept of servitization gained the attention from the manufacturing industry. However, it has been in recent years when there has been a notably growth in the interest among manufacturing companies in the idea of adapting their business models towards servitization [NBI15], mainly due to two reasons: firstly, in a globalization era the need for differentiation has increased dramatically [NBI15] among companies, which are seeing servitization as a key strategy to differentiate themselves from their competitors in a global market [Niñ17]; and secondly, attaching value-added services to their products allows companies to increase the value provided to their customers which helps to reinforce the relationships with their customers, hence locking out competitors [AAAS15], [Niñ17]. As a consequence, more and more companies have already expanded the scope of their business model with services, to enhance market competitiveness and gain more revenues [TQ19].

Driven by the new information technologies and data related technologies, this new business model makes use of the growing volume of data that is being captured from manufacturing processes to provide Smart Services such as preventive maintenance of the equipment, advanced diagnostics applications, product quality control, etc. [BMN15]. The so called Smart services, which are defined in [CMJB18] as services *"that reacts on collected and analyzed data based on networked, intelligent technical systems and platforms,"* have been seen an important means to enhance Smart Manufacturing [TQ19]. Indeed with the aim of becoming the leader country in Europe in terms of digital growth, after its first strategic initiative Industrie 4.0 (see Section 2.2.1), Germany started promoting the second strategic initiative, entitled "Smart Service World," which is focused on the value chains that incorporate the smart products made by the Industrie 4.0 once they have left the factory [KRH+14].

However, the provision of smart services involves important challenges for companies such as Capital Equipment Manufacturers (CEMs) which business core (i.e., the production of equipment to be used in other manufacturing processes) often lacks the technological skills and specialized teams to develop and deploy high IT-based solutions, required for adopting Smart Manufacturing approaches or for transforming their business

model via data-driven servitization [Niñ17]. This introduces an additional barrier for the CEMs aiming to design, develop and launching to the market these Smart Services, as they need to integrate new capabilities related to the key enabling technologies supporting them.

In order to face these challenges, CEMs (and other manufacturers) often need to establish strategic partnerships with technological partners who are specialized in providing the involved key information technologies. In fact, as stated in [CMJB18] *"In contrast to the technology of Industry 4.0 which can exist in just one specific sector, Smart Services require cross-functional areas."* This way, the expertise of these partners in information technologies can be combined with the manufacturers knowledge from their specific sector in order to design the aforementioned smart services. In such a business context emerges the role of ITS Providers, whose business model is focused on providing the key enabling technologies supporting Smart Manufacturing approaches.

2.2.4 Business Context of Information Technologies Service Providers in Smart Manufacturing

Information Technologies Service (ITS) Providers are those companies whose business model is focused on supplying information technologies as services (ITaaS) and enterprise software for other companies [Niñ17]. In the particular context of the industry, ITS Providers are specialized on supplying the key enabling technologies supporting the adoption of Smart Manufacturing approaches among manufacturers and helping them shifting their business model towards data-driven servitization approaches. Consequently, this kind of specialized ITS Providers play a prominent role in Smart Manufacturing scenarios.

The role of ITS Providers emerges in response to the difficulties that manufacturing companies, especially manufacturing Small and Medium Enterprises (SMEs) were facing when attempting to smartize their manufacturing business, mainly due to the lack of technological skills and specialized teams to develop and deploy high IT-based solutions, required for adopting Smart Manufacturing approaches or for transforming their business model via data-driven servitization. Indeed, the smartization of a manufacturing company involves the adoption of multiple artifacts and software products that arise from the integration of the key enabling technologies supporting it, which are out of the scope of the business core of manufacturing companies (barring big manufacturing companies).

The services supplied by ITS Providers cover a wide spectrum of information technologies from the different key enabling technologies involved in the Smart Manufacturing approaches. However, the adoption of such approaches are driven by the datification of almost all the aspects of a manufacturing process and thus, data-related technologies acquire more relevance in this context (which is why usually these providers are also known as Infrastructure for Big Data Services (IBDS) Providers [Niñ17]). Therefore, the provided services are usually focused (but not limited) in

the technologies involved in the manufacturing data life-cycle, which include, among others: the deployment of sensors devices and actuators and their integration into purposeful solutions to capture and export data from manufacturing facilities (with regard to IIoT technologies); technologies related with the storage, management and exploitation of the captured data and their processing centralization across massive cloud-based distributed computing infrastructures (related to Big Data and Cloud Computing technologies); and extracting valuable and useful information and knowledge from the exploited data (regarding data analytics and KDD-related technologies).

Through those services, ITS Providers allow CEMs and other manufacturers (especially manufacturing SMEs) to overcome the limitations they were being encountered when shifting their business towards servitization approaches for providing data-driven smart services. This opens the door to establish strategic alliances between manufacturers and ITS Providers that benefit both parties. On the one hand, manufacturers receive the specialized know-how on the involved technologies and for the effective design and deployment of their data-driven servitization strategies. On the other hand, the ITS Provider gains access to multiple manufacturing sectors and the specialized know-how on each of their business context, which favours the design of more general purpose solutions based on the required data-related technologies with cross-sector applicability (instead of solutions targeted for a specific sector). This way, the built solutions can be deployed in different manufacturing sectors leading to a high replicability potential on the deployment of their IT solutions for a higher monetization of them.

The business model of an ITS Provider follows a *utility computing* based model on which hardware and software resources are provided through cloud computing technologies on-demand basis that can be grouped into three main categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-service (SaaS). However, in the particular context of Smart Manufacturing, ITS Providers offering services targeted to SMEs, usually adopt the PaaS model as the core of their business model for building purposeful platform-level solutions that integrate IIoT, Big Data, Cloud Computing and Data Analytics technologies, to support data-related exploitation services (e.g., data gathering from different sources and integration).

Those solutions are usually deployed over the infrastructure provided by another ITS Provider, specialized in supplying cloud-based infrastructure resources (usually storage and computing resources), in a IaaS model. Moreover the platform solution is usually deployed in specific manufacturing sectors, where the partnerships established with the manufacturers allow them to co-design sector-specific solutions that generally are provided in a SaaS model on the top of the deployed platform solutions. At the same time, on the top of those sector-specific solutions, diverse smart services are provided (by other manufacturer such as CEMs) for different manufacturing sectors. Finally, it is worth mentioning that although this kind of ITS Providers usually adopt the PaaS model as the core of their

business model, they could also provide their own hardware and infrastructure resources in a IaaS model (e.g., IIoT devices to capture and process the data) and software in a SaaS model (e.g., Smart Services directly applied by manufacturers).

Chapter 3

Real World Context

In the context of Smart Manufacturing, the strategic partnerships established between manufacturers and specialized ITS Providers supplying services targeted at supporting the data-driven servitization approaches among those manufacturers, have given rise to the real-word Smart Manufacturing context of this research work. The access to this context was facilitated by an ITS Provider with which the research group on which this research work has been carried out has been collaborating in the period between 2014 and 2020. Moreover, the ITS Provider also facilitated the collaboration with two particular CEMs with which it has established partnerships to supply their services and provide their solutions. Both CEMs granted the access (through the ITS Provider) to ongoing collaboration projects as part of their servitization strategies, which provided:

- A better understanding of the business setting and a detailed characterization of the main agents involved in it, which enables the extraction of valuable insights from the business setting and the limitation of the research context in terms of the Smart Manufacturing scenarios towards which the contributions proposed in this research work are targeted.
- Access to raw data captured from real operating factories, together with different materials providing: on the one hand, a detailed characterization of the machinery of these factories and the different sensors implanted in it; and on the other hand, a detailed description of the manufacturing process, in order to facilitate the understanding of captured data and their properties.

Within this context, the research group on which this work has been carried out, has launched a project with the aim of developing useful contributions that integrate the research work in related research fields, into the real world. Ensuring this way, that the proposed contributions are interesting research innovations for the related fields, but also useful contributions for real-world scenarios.

The rest of the chapter is organized as follows: firstly, a characterization of the Smart Manufacturing scenarios where the contributions of this research work are targeted is presented (for simplicity, this characterization is presented in terms of the scenario of a particular CEM with which

the research group has collaborated during most of this research work); and secondly, an overview of the research fields on which this work aims to provide interesting contributions, together with details of the different technologies from these fields supporting the proposed solutions.

3.1 Real-World Business Setting

This research work is driven by the challenges and requirements that arise from a real-world scenario aiming to adopt Smart Manufacturing approaches. However, it is worth mentioning that while this scenario serves as a case study, providing the ground for the field validation for the contributions of this work, the scope of the proposed solutions is not focused only to this particular scenario. Therefore, although this real-world scenario serves to characterize the Smart Manufacturing context to which the contributions of this research work are targeted at, it does not limit the scope of the proposed solutions, which could also be applied on other scenarios facing similar challenges or with similar requisites. Indeed, although in this section a real-world scenario conformed by an ITS Provider and a particular CEM is presented, this is not the only scenario to which this research work has had access. For example, the first contribution of this research work has been carried out within another instance of the characterized scenarios, conformed by the ITS Provider and another CEM.

The real-world scenario arise from a collaborative project on which three different types of industrial agents are involved: an ITS Provider, a CEM and various manufacturers. As part of its data-driven servitization strategy, the CEM has established a strategic partnership with an ITS Provider with the aim to offer not only equipment, but also value-added data-enabled services to their customers (other manufacturers). In the following subsections, the different agents involved in this business setting are presented, together with a characterization of a particular instance of each of them in the real-world business setting. In addition, some of the challenges that arise from such a business context are presented along with the proposed solutions in this research work, aiming to provide useful and innovative contributions to help facing them.

3.1.1 ITS Provider

The ITS Provider[1] with which this research work has been carried out is an Information Technologies SME whose business model is based in the deployment of IT-related services *(ITaaS)* related to the key enabling technologies supporting Smart Manufacturing: Internet of Things, Big Data, Cloud Computing and Data Analytics. They deploy IIoT devices that are connected to the IT infrastructure of the manufacturing plants. Those devices allow to capture huge volumes of data generated during the continuous operation of the manufacturing process or equipment to be analyzed. The captured data (mostly time-series data captured by industrial sensors

[1]ITS Provider with which this research work has been carried out: `https://www.savvydatasystems.com/es/inicio`

monitoring different magnitudes or indicators of interest) are automatically transmitted to a Cloud Computing environment, on which the ITS Provider deploys its platform solution for different exploitation functionalities (e.g., predictive maintenance of equipment) on those massive volumes of data [Niñ17].

The services supplied by this ITS Provider are mainly targeted at CEMs supplying their equipment to other manufacturing companies (usually SMEs) worldwide, and therefore, their equipment is deployed in different manufacturing plants distributed all over the world. The solutions implemented by this ITS Provider allow these CEMs to shift their business towards data-driven servitization approaches, focused on the exploitation of the data produced during the continuous operation of the manufacturing process of these equipment. Moreover, this ITS Provider also deploys its solutions directly, as smart services, in manufacturing companies (usually SMEs) aiming to adopt Smart Manufacturing approaches. Figure 3.1 outlines the business model of this ITS Provider.

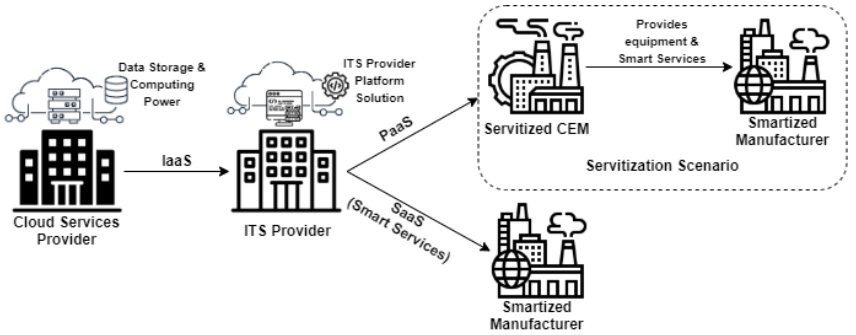


FIGURE 3.1: Business model of an ITS Provider

Thereby, this ITS Provider supplies services to different customers (mainly CEMs) from different sectors. Depending on the specific manufacturing business sector where these data-driven services are provided, they may be focused to different purposes, such as predictive maintenance of equipment, product quality or process efficiency control, fault diagnosis, etc. These customers, own a total of 85 manufacturing facilities distributed worldwide on which more than 600 machines and capital goods are managed, which generate more than 400 GB of data every week [Sys20]. The global scale of these customers and the massive amounts of data generated by all the manufacturing processes distributed worldwide to be exploited with different purposes, characterize the data-driven services offered by this ITS Provider.

In order to provide these services, the ITS Provider implements a platform solution that integrates the required technologies supporting the management of the data during the manufacturing data life-cycle (data capture, storage, processing, visualization and diverse analytical exploitations). Given the characteristics of the services to provide (heterogeneous services for different customers with different purposes, and driven by

massive amounts of data coming from different manufacturing plants distributed worldwide), this platform is implemented in a *PaaS* model on the top of a generic cloud computing infrastructure supplied by a Cloud Services Provider *(IaaS)*. This architectural approach allows to scale in a flexible way the required data storage and processing (computing power) resources while optimizing the use of dedicated resources for an efficient management of the costs. Moreover, it allows minimizing the investments required for building and deploying such a platform.

3.1.2 Capital Equipment Manufacturer

Among the different customers with which the ITS Provider has established strategic partnerships to supply their services and provide their solutions, the ITS Provider has facilitated the collaboration with a particular CEM aiming to shift its business model towards servitization and to provide smart services to its customers. This CEM is a manufacturing SME whose business model is based on selling equipment for the extrusion processes-based manufacturing sector. The manufacturing sector for which this CEM provides equipment is mainly based on the production of plastic containers (e.g., bottles) based on an extrusion-blowing process. They provide different equipment (e.g., extruder machines) to manufacturing companies distributed all over the world which may own different facilities. Moreover they also deploy their own equipment in their own plastic bottles production plant, so that they also provide packaging containers to different sectors (e.g., food sector). Figure 3.2 outlines the business model of this CEM.

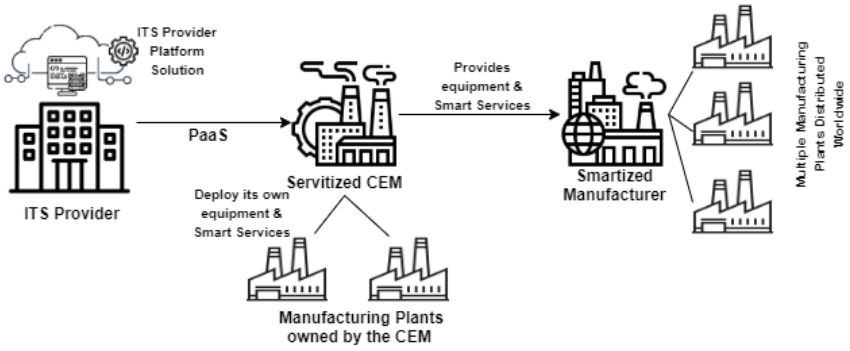


FIGURE 3.2: Business model of the CEM

The core of the production process of this manufacturing sector is based on an extrusion-blowing process known as *plastic extrusion*. Plastic extrusion is a mechanical manufacturing process in which raw plastic is melted and formed into a continuous profile. The extrusion process starts by feeding plastic material (usually in the form of small plastic particles called pellets) from a hopper into the barrel of the extruder. The material is gradually melted by the mechanical energy produced by turning screws and by heaters arranged along the barrel. The molten polymer is then pushed

into a die, which forms the polymer into a parison (i.e., a tube-like piece of plastic with a hole in one end through which compressed air can pass), which is then clamped into a mold and air is blown into it. The pressurized air then pushes the plastic out to match the mold. Once the plastic has cooled and hardened, the excess is trimmed, and finally, the mold opens up and the product is ejected. Figure 3.3 outlines an extrusion-blowing process.

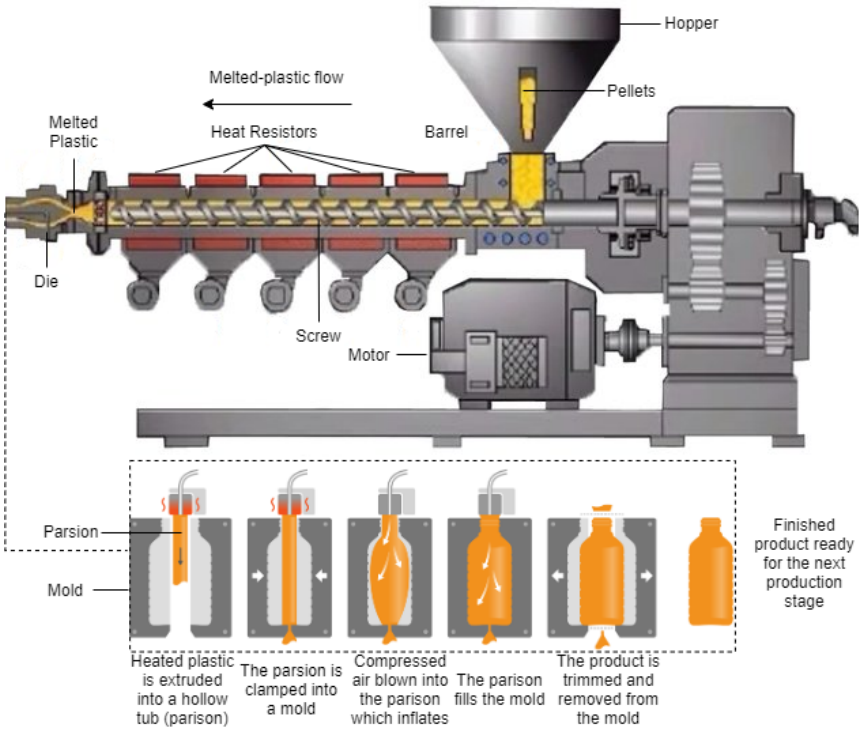


FIGURE 3.3: Extrusion-blowing process for the production of plastic bottles (adapted from [GSRP+18])

One of the key aspects of this manufacturing process is that there are several key parameters and variables controlling it. However, currently there not exist an automated system that relates all the relevant variables in the production process that helps to run the process with an optimal performance. In fact, those parameters and variables are usually managed by skilled operators that manage and adjust them based on their own criteria and experience, but without any scientific basis or analytical formula that helps them. This semi-artisan way of operating introduces several inefficiencies in the production process that sometimes entails extra costs in the final product due to energy waste, impurities in the product, poor quality of it, not an optimal use of the raw materials from which the final product is produced, etc. Moreover, taking into account that it is a continuous process, any optimization, however small, that can performed in

such a system, it would be transformed into great benefits at the end of the year.

In order to optimize the production process on which its equipment its implanted and plays a crucial role, this CEM, aims to offer smart services to their customers to transform their business towards Smart Manufacturing approaches and thus, increase the value of their production systems with an optimized performance. As part of its data-driven servitization strategy, this CEM has installed various sensors and data capture and collection devices (IIoT devices) on the equipment that it provides. Those devices capture data from a variety of equipment, setting parameters and physical magnitudes (temperatures, pressures, etc.) related to the raw materials, production processes and industrial equipment from the production plants. Moreover, through the collaboration with the ITS Provider, it has implemented a specific infrastructure to ensure the capture and management of the raw data that can be extracted from the IIoT devices implanted in a manufacturing plant.

This infrastructure consists among others of a *"collector"* (i.e., a device installed in the production plant that is in charge of centralizing the captured raw data by the different IIoT devices); the hardware necessary to capture the raw data from the different phases of the production process; networks, hardware and software required to interconnect all data sources and the centralized machine; software for storing and integrating raw data; and specific purpose devices to automate the connection and data exchange between the plant machines and the cloud storage resources. Figure 3.4 provides an overview of the architecture of this infrastructure.

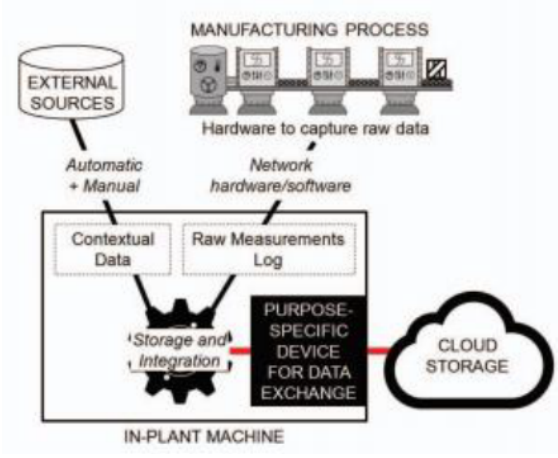


Figure 3.4: Infrastructure architecture implemented by the CEM (extracted from [NBI15])

3.1.3 Smart Factory

Once the infrastructure has been deployed and the availability of the raw data is guaranteed, the objectives of the project resulting from the collaboration of the CEM and the ITS Provider are organized according to two key processes: transformation and the integration of the raw data into a master (pre-processed) dataset, and then, offering different Smart Services, built on the top of the master dataset, that allow to optimize the production process. Depending on the manufacturing sector, there may be several aspects that influence the optimization of the production process. Therefore, the purpose of these services may vary according to the manufacturing-sector (e.g., equipment maintenance, fault diagnosis, product quality control, production optimization, etc.).

Concerning the particular context of manufacturing based on a blown-extrusion process, it is characterized by being a sector with a high product quality (i.e., the ratio of defective bottles produced is really low) and good use of raw materials (e.g., the material that is wasted in the production process is fed back as new raw material to the hopper). Therefore, although it would be also desirable the optimization of some aspects, such as product quality or making the most of raw materials, these aspects do not suppose a major concern for the manufacturers of this sector. On the other hand, the production based on a blown-extrusion process is a continuous process in which the uptime of the machine plays a crucial role, since long periods of downtime considerably decrease production. Therefore, one of the objectives of these manufacturers is to maximize the uptime of the machines. In this regard, equipment maintenance and diagnosis plays a crucial role in promoting machine uptime, as well as it can have a potential impact on the operating costs, on which some estimates claim that equipment maintenance can exceed 30% of total operating costs, or between 60% and 75% of equipment life-cycle cost [OLBO15a]. Therefore, among all the Smart Services to be built for this particular context, those related with the equipment maintenance and diagnosis stands out.

3.1.4 Real-world Challenges that have Motivated the Contributions of this Research Work

The adoption of Smart Manufacturing approaches poses a major challenge for manufacturers with regard to the required Information Technologies, which demands the support of technology suppliers specialized in Smart Manufacturing oriented IT-Services. However, these specialized technology suppliers also face important challenges when deploying their solutions. In this regard, the collaboration with an ongoing project by the three industrial agents mentioned before, has facilitated a first-hand identification and understanding of some of these challenges.

One of the major challenges that arise in such a business setting is related to the data pre-processing process that must be performed in order to extract useful information from the massive amounts of data coming from heterogeneous IIoT devices implanted in the multiple manufacturing plants to which the ITS Provider supplies services. In fact, the pre-processing of

massive amounts of raw data represents one of the major challenges in the field of Big Data [OLBO15b], mainly due to the multiple existing alternatives to treat specific problems, the diversity of treatments that each data may require, and the lack of structured methodologies and automatic approaches to determine which techniques to apply in each situation. In this regard, the first contribution of this research work aims to facilitate the pre-processing task of those data through an automatized approach that helps in the selection of the most adequate pre-processing (cleaning and reduction) techniques to apply to each data type.

The storage of those massive amounts of data also poses a major challenge for the ITS Provider when deploying their cloud-based platform. This platform is implemented in a *PaaS* model on the top of a generic cloud computing infrastructure supplied by a Cloud Services Provider *(IaaS)*. This architectural approach allows to scale in a flexible way the required data storage and processing (computing power) resources while optimizing the use of dedicated resources for an efficient management of the costs. However, despite the optimization of resources and their associated costs, that this on-demand business model provides, the required resources (data storage, transmission, processing, etc.) to handle all the data generated by all the connected facilities owned by all of its customers, still represents an important internal cost for the ITS Provider, that could hamper the sustainability of such a business model. Is in this regard, where the second contribution of this research work aims to help, alleviating those cost through a data storage architecture that allows to reduce the required data storage resources.

Those data are captured and stored with the aim of extracting useful information from them that allow offering to manufacturers different data-driven Smart Services. Among the Smart Services that the CEM aims to offer to their customers, the predictive maintenance of equipment stands out due to the importance of the role that it plays in promoting machines and equipment uptime [OLBO15a]. In order to employ a more efficient equipment maintenance strategy, the CEM has already implanted an alarm system to control the production process, and warn the operators in the plant about situations that could hamper the machine operation or incur in stops in the production process. Overall, those alarm systems play a prominent role in maintaining plant safety and operation efficiency of modern industrial plants, by keeping the processes with normal operating ranges [WYCS16]. However, sometimes, in those systems the activation of the alarms is so close to the issue that there is no action-margin for the operators to manage the situation [LQPH13], and therefore, new approaches are required [WYCS16]. Is in this regard, where the third contribution of this research work aims to help to conduct a predictive maintenance of equipment, through an early prediction of those alarms, that allows to reconfigure, in time, the settings of the machine in order to avoid stops in the production process or hampering the machine.

The aforementioned challenges have served as the motivation for the proposal of the contributions of this research work that, on the one hand, aims to build purposeful solutions, aligned with the achievement of the

goals established for the collaborative project among the three industrial agents; and on the other hand, are supported by the identified opportunities for innovative contributions for the research areas related to the different steps that compound the *manufacturing data life-cycle* (see Figure 2.15). The first two contributions are related to the requirements of an ITS provider with regard to the early stages of the data life-cycle (data storage and pre-processing), while the third one is related to advanced analytics demands (predictive maintenance of equipment) that arise from the smart services that a CEM aims to provide to its customers in the later stages.

These contributions would be integrated in the form of different smart services in the platform solution of the ITS Provider. The integration of these contributions in the form of services would lead to more modular solutions on which the different services could be interconnected and interact with each other. Indeed, some of the time series dimensionality reduction techniques that the pre-processing service built in the first contribution recommends, are those used to reduce data storage resources in the architecture proposed for the storage service in the second contribution; whereas some of the time series cleaning techniques recommended by the same pre-processing service, are the techniques that allow to prepare the time series for the models built in the alarm prediction system in the third contribution. The solutions proposed in these contributions are supported by different technologies from different knowledge areas and research fields which are presented in the following section.

3.2 Technological Context

Since the third industrial revolution, Information Technologies have driven manufacturing towards informatization. The progressive adoption of these technologies in manufacturing has led to factories where manufacturing processes are controlled by hardware and software fully integrated into manufacturing systems. However, it has been during the last years when the recent advances in cutting-edge technologies, such as Internet of Things, Big Data, Cloud Computing and Data Analytics, have given rise to what is conceived as the smart factory; an informatizated factory driven by data, in which all of its processes are connected and interact with each other, in order to yield positive impacts in all aspects of manufacturing.

As a result, in recent years there has been a notably growth in the interest among manufacturing companies in the idea of adapting their business models towards data-driven Smart Manufacturing approaches. However, the adoption of the data related technologies that support Smart Manufacturing still faces significant challenges, which opens the door to a brand of opportunities for interesting contributions by researchers and practitioners from different fields. Consequently, during the last years, many recent advances in the key enabling technologies supporting Smart Manufacturing and the research fields related to them have been presented. In this section a brief overview of some research fields related with the proposed contributions in this research work, and some of the most relevant technologies

associated to them are presented. First, the required steps for the pre-processing of the captured data from real-world operating manufacturing plants and some of the different techniques entailed in this process; then, some existing alternatives for storing and managing massive amounts of industrial data; and lastly, different techniques from the data analytics related areas that are used in manufacturing intelligence approaches.

3.2.1 Pre-processing of Manufacturing Data

Data pre-processing [GLH15] is one of the major steps comprehended in the Knowledge Discovery in Databases (KDD) process [RGKG+17]. This step focuses on one of the most significant challenges of the KDD process, which consists on adapting the data to the requirements of the data mining algorithms. Consequently, the data pre-processing step usually takes a considerable amount of whole KDD process time (greater than 50% of the total effort) [GLH15]. The ultimate goal of this step is to obtain a correct dataset, on which different data mining algorithms can be applied.

Data collection is usually a poorly controlled process which results in raw data with many imperfections, such as inconsistencies, errors, missing values, noise, impossible data combinations, etc. As a consequence, the resulting data (i.e., bad quality data or dirty data [KCH+03]) are not suitable to start the data analysis process. Indeed, some of the data analysis techniques and data mining algorithms cannot be directly applied over raw data that has not been pre-processed (e.g., for being unable to deal with missing values, being sensitive to noise, etc.). Moreover, analyzing data that has not been carefully selected and prepared for such tasks, not only introduces difficulties in the knowledge discovery process (due to irrelevant and redundant information or to the presence of noise and unreliable data), but also can produce misleading results. Therefore, data must be correctly pre-processed to ensure data quality for accurate data analysis [KCH+03].

Data quality has been defined in [GLH15] upon three main elements: accuracy, completeness and consistency. With the aim of transforming raw data into quality data for accurate data analysis, data pre-processing involves different tasks that could be grouped in: data preparation tasks; compounded by integration, cleaning, normalization and transformation of data; and data reduction tasks; such as feature or instance selection, discretization, etc. [GLH15]. These pre-processing tasks are chained to achieve a quality dataset, which can be considered appropriate and useful for further data mining and knowledge discovery algorithms. Figure 3.5 outlines the main pre-processing tasks which are presented next:

- *Data Integration:* The data integration process consists on gathering all the data coming from different and variate sources into a single dataset. Data integration should be properly performed in order to avoid the appearance of inconsistencies and redundancies, which decreases the accuracy and the speed of the data mining algorithms.

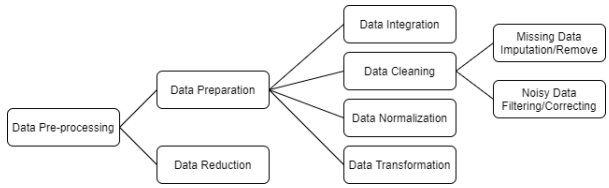


FIGURE 3.5: Main data pre-processing tasks

- *Data Cleaning:* Dirty data may include missing data or wrong data which hampers the knowledge discovery process over those data. Therefore, before applying data analytics or data mining techniques, data bust be cleaned in order to remove or repair dirty data. There are multiple sources of dirty data: data entry errors, data update errors, data transmission errors, data processing errors, etc. However, in general dirty data is presented in two forms: wrong (noisy) data, that must be filtered or corrected; and missing data, that must be ignored or filled in.

- *Data Normalization:* Adjust data attributes measured with different scales and units of measurement to a common scale in order to give every attribute the same weight. Some data mining algorithms such as Artificial Neural Networks or distance-based methods work much better with normalized data.

- *Data Transformation:* Conversion of the data, so that some data mining process can be applied to them or that this process is more efficient. Data transformation usually applies different mathematical formulas to the original raw data attributes to construct new attributes (e.g., smoothing, aggregation, summarizing, discretization, etc.).

- *Data Reduction:* Huge volumes of high-dimensional data makes it difficult to extract useful information from them. Moreover, massive amounts of available data usually take a long time to process for data mining and data analysis algorithms which make the analysis unfeasible. Data reduction allows to transform those data to obtain a reduced or compressed representation of them while maintaining the information as whole as possible.

To carry out these data pre-processing tasks, over the years multiple techniques, methods and algorithms have been proposed in the literature. A review of some of the most influential ones can be found in [GLH15] and [GLH16]. However, not all of them are applicable to every type of data; and therefore, specific techniques have also been defined for some types of data. In the particular context of Smart Manufacturing, time-series data are the predominant type of data. Time-series data and time series data mining methods have some peculiarities (e.g., time order dependence) that make some techniques unsuitable for this kind of data. Therefore, many specific time series pre-processing techniques have been proposed.

With regard to time series pre-processing, in manufacturing scenarios, time-series data usually come from sensors and IIoT devices monitoring the manufacturing process, which are particularly subject to noise and outliers [EA12a]. Moreover, sometimes measurements are lost (i.e., missing values) due to sensor failures when capturing data, or during the transmission between the sensor and the machine in charge of gathering and centralizing the captured data [VVD+18]. These problems are usually handled with time series cleaning techniques. Cleaned data are then analyzed by different data mining and artificial intelligence techniques which may require some time-series data normalization and transformations in order to match their input/output specifications (e.g., segmentation, convert into a supervised learning problem through sliding-window approaches, etc.). Furthermore, some data analysis techniques, such as time series similarity search, time series clustering, and time series data mining, scale poorly to high-dimensional data and therefore, time-series data dimensionality reduction techniques are usually applied to optimize the performance of these techniques and their ability to extract useful patters from data [LKLC03].

A wide variety of techniques, methods and algorithms have been proposed in the literature to accomplish the different tasks involved in the time series pre-processing. However, in the same way that not all data pre-processing techniques are appropriate to time series, within the time series specific pre-processing techniques, there are also some ones that are more suitable than others depending on the type of time series. Knowing which are the most appropriate pre-processing techniques to apply for each type of data in such a scenario as the presented in the previous section, where multiple sensors and IIoT devices of different nature capture multiple heterogeneous time series, is an unfeasible task for a data engineer or the person in charge of the data pre-processing task.

Therefore, a more systematic approach is required when selecting which techniques to apply to each type of time series. In this regard, the first contribution of this research work aims to provide some recommendations and suggestions about which cleaning and reduction techniques are the most suitable ones for the considered time series in order to facilitate the pre-processing task. In the following subsections, some of the most popular techniques used to accomplish these pre-processing tasks are reviewed. Among those techniques, a subset of the most interesting ones have been considered when building the *recommendation system* in the first contribution of this research work (see them in Chapter 4).

3.2.1.1 Time-Series Data Cleaning

The main goal of the data cleaning techniques is to obtain an adequate representation of time series that will be part of the datasets used for different purposes, such as the control of product quality, the predictive maintenance of equipment, etc. When cleaning time-series data to ensure data quality, the most relevant tasks are related to handling missing values and detecting (and treating) noise and outliers [BRG+12]. Each of these tasks are presented below.

Handling Missing Values

In real-world manufacturing scenarios it is common to find missing values in a dataset [GLH16]. Missing values occur when no data value are stored for certain observations due to several reasons (sensor failures, failures in the transmission, etc.). The presence of missing values in a dataset can affect the performance of the data mining algorithms built from it, or even prevent its construction if the algorithm cannot handle them [AR04]. Therefore, missing values treatment is considered one of the basic steps in the data cleaning process [PL05], [GLH15], and different methods have been appearing in the literature to deal with them [KLTCM20], [MSBB+15], [Vil15] [VVD+18]. An example of them can be seen in Figure 3.6, where the *Kalman Smoothing on structural time series models* method is used for the treatment of missing values in an industrial time series. According to Pratama et al. [PPAI16], those methods can be grouped into three main categories which are presented next:

- *Simple Methods:* Some of the most trivial methods for dealing with missing values. Despite their simplicity, these methods are quite effective and widely used, especially when the ratio of missing values is low (e.g., lower than 1%) [AR04]. These methods include among others: *ignoring* them (if data analysis algorithms allow it), removing them (*Listwise* or *Case deletion*) or using some available data to treat them in a simple manner, such as using the last available data to impute the missing values (*Last Observation Carried Forward*) (LOCF) [Che14].

- *Imputation Methods:* Missing values are replaced with estimated ones based on some information available in the dataset. Some of the most simple methods are the imputation of statistical values obtained from the dataset, such us the mean, the median or the mode (e.g., *Mean Imputation*); sliding-windowing approaches, such as moving average based methods (e.g., *Weighted Moving Average* [MSBB+15]); and *Hot and Cold Deck Imputation* methods [JB12], where some missing values are imputed using the data from a selected "donor" (LOCF is a simple form of hot-deck imputation on which the "donor" is the previous instance). Moreover, there are also *Multiple Imputation* methods, such as MICE [RW+11], on which missing values are imputed various times with different estimates and analyzed, to finally, combine the results and yield a single combined estimates that formally incorporate missing data [Sre14]. These kind of methods are quite simple and effective for manageable rates of missing values (e.g., rates between 1-5%). However, for managing higher rates of missing values (up to 15%, since higher ratios may severely impact data analysis) more sophisticated methods are required [AR04].

- *Advanced Methods:* More advanced methods have also been proposed by adopting soft computing techniques from other fields [PPAI16], such as machine learning, neural networks, pattern matching, etc. This kind of methods usually learn models or patterns from the data

and then use them to impute the missing values. Some of these type of methods are: interpolation based methods such as *Linear Interpolation* [MSBB+15] or *Spline Interpolation* [MSBB+15]; *Maximum Likehood* based methods such as the use of the *Expectation Maximization* algorithm to find maximum-likelihood estimates for the missing data [PE04]; similarity based methods, such as the *k-Nearest Neighbour* (KNN) based *kNNImpute* method [Zha12]; prediction models based methods, such as *Kalman Smoothing on structural time series models* (or on ARIMA models); or methods that rely on neural networks such Recurrent Neural Networks (RNNs) [YZvdS18] or Long-Short Term Memory (LSTM) RNNs [LAL19]. In addition, in recent years several methods have been proposed attempting to recover large missing value blocks in time series. A wide spectrum of these kind of advanced methods is covered in [KLTCM20] where they are categorized into two groups regarding if they are *matrix-based* methods, among which *CDRec* [KBM15] or *TRMF* [YRD16] stands out; or *pattern-based* methods, within *DynaMMo* [LMPF09] or *ST-MVL* [YZZL16] stands out.

FIGURE 3.6: Missing values imputation with Kalman smoothing on a structural time series model

Noise Detection

Noisy data are data that have been corrupted, or distorted with an amount of additional meaningless information (data = true signal + noise) [VVD+18]. In real-world manufacturing scenarios it is common to find noisy data, which may come from various sources, such as sensors inaccuracies, interferences in the transmission of the data, a bad use of the sensors, process disturbances, equipment degradation, etc. The presence of noise in data is a common problem that may produces several negative consequences in data analysis, since some data mining methods or models are sensitive to noise [GLH15]. Therefore, noise must be detected and removed in order to improve data quality for further analysis. Different approaches have been proposed in the literature for dealing with noise in time series. An example of them can be seen in Figure 3.7, on which noise is removed from a time series, by using a *Frequency Filtering* technique

(low-pass filter). Moreover, some of the most popular noise removal techniques are presented next, which can be grouped in three main categories:

- *Smoothing Techniques:* These techniques are often employed to remove noisy data [ZSWY17] by increasing the *Signal-to-Noise* Ratio (SNR). For example, signal averaging techniques, such as *Simple Moving Average* (SMA) or *Exponentially Weighted Moving Average* (EWMA) can be applied to smooth a time series by averaging the data within a sliding window [ZSWY17].

- *Filtering-based Approaches:* Noise can be filtered and removed by using classical signal processing techniques like frequency filters or wavelet thresholding [EA12a]. Furthermore, in the literature different filtering based approaches have been proposed to remove noise from time series [KL05b]. Some of the most popular ones are *frequency filtering* techniques (e.g., low/high pass filters). In this kind of techniques usually time series are transformed from the time-domain to the frequency domain by using a transformation function such as the *Fourier Transform*, which converts a time series into a sum or integral of sine waves of different frequencies, each of which represents a frequency component. Figure 3.8 (i) shows an example of the decomposition of a signal into harmonic sinusoidal waves of different frequencies. With this idea, signals' noise can be removed by filtering the frequencies that generate it. For that, usually the time series are represented in the frequency domain and then, the frequency spectrum is analyzed. Figure 3.8 (ii) shows an example of using a low-pass filter to remove the highest frequencies from a time series, which are the corresponding ones to the noisy signals. Other methods, such as the *Kalman Filtering* [HYZ+17] or wavelets based filtering [AAF02] have been also used in the literature.

- *Noise Robust Approaches:* Instead of filtering the data or removing the noise, other approaches have attempted to diminish the effect of the noise in data analysis methods [GLH15]. This kind of approaches include among others, *noise robust similarity measures* [EA12a] for distance based algorithms, and noise robust learning models, such as the neural networks based time series prediction models presented in [LCST09] and [GLT01].

Outliers Detection

An outlier is an observation that is considerably distant from other observations [VVD+18]. In real-world manufacturing scenarios the presence of outliers may be due to faults on the sensor measurements, bad uses of the sensors, production anomalies, etc. In this regard, it is worth mentioning that, although outliers could hamper models performance, and thus, they should be treated; sometimes they could represent an event of interest so that the outlier itself should be analyzed [BGCML20]. Detecting outliers

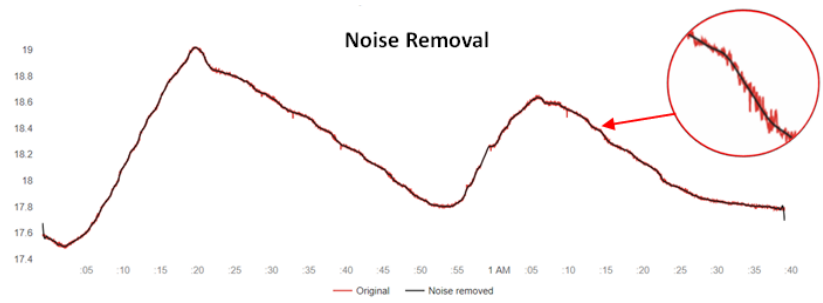


FIGURE 3.7: Noise removal with frequency filtering

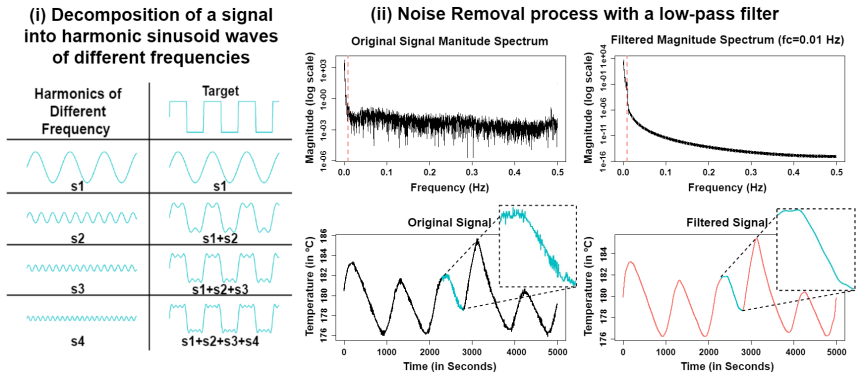


FIGURE 3.8: Frequency filtering

could be an arduous task, since the concept of an outlier varies depending on the characteristics of the data; for example, the nature of the data (univariate or multivariate time series), the type of outlier (single point outlier or sub-sequence of outliers), etc. Therefore, multiple methods have been proposed in the literature for detecting different types of outliers. A review of them can be found in [BGCML20], [GMM18] and [Man14].

Those methods have been proposed following different approaches ranging from simple statistics, such as the *Adjusted Tukey's Range Test* [HV08], which will classify as outlier any value that falls outside a defined range of values based on the *interquartile range* of the data; proximity based approaches, such as detecting outliers with an established threshold on the correlation of contiguous values (detecting as outliers those consecutive values in a time series between which the difference is greater than the established threshold); density based methods, such as the *Local Outlier Factor* (LOF) [BKNS00]; clustering-based methods [HXD03]; or prediction-based methods on which a model is trained using only past data and used to predict further observations so that points that are very different from their predicted values are identified as outliers (e.g., [MSDA19] and [MTR+16]). An example of a method for detecting outliers is shown in Figure 3.9, on which an *Adjusted Tukey's Range Test* is performed to detect outliers in a time series.

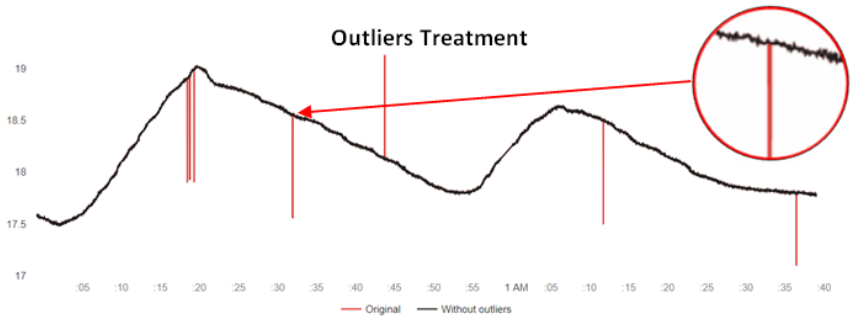


FIGURE 3.9: Outliers detection (and treatment) with Adjusted Tukey's range test

3.2.1.2 Time-Series Data Dimensionality Reduction

The increasing data generation and capture capability has led to huge datasets that makes it difficult to conduct knowledge discovery. On the one hand, much irrelevant and redundant information increases the difficulty to extract useful information from the data. On the other hand, the massive amounts of available data could take a long time to process for data mining and data analysis algorithms, and thus, makes their analysis unfeasible. In this regard, data reduction allows to obtain a reduced or compressed representation of the data which is much smaller in volume than the original data, while maintaining the information as whole as possible. If some information is lost, then the compression will be lossy, whereas if the original information can be recovered from the reduced representation, the compression will be lossless.

In general, time-series data are high-dimensional data [WMD+13], [EA12a] and working with these data in raw implies a high cost in terms of processing and storage. In fact, the inefficiency of processing and storing large volumes of raw time-series data has been explicitly mentioned as a strong motivation for the analysis of time series representation techniques [PVK+04], [EEC+09]. Time series representation techniques allows to reduce the dimensionality of time series, while still preserving their fundamental characteristics. Moreover, they also make it possible to optimize some data analysis techniques associated with time series processing [ASW15], [LKLC03], such as time series similarity search, time series clustering, and time series data mining, where most of the algorithms scale poorly to high-dimensional data. Therefore, time series representation has been stated of one of the most relevant steps in time series data mining [RGKG+17], [EA12a], [Fu11].

Various studies have addressed the representation of time series through the application of reduction or approximation techniques to time-series data. For example, in [Fu11], it is provided a very thorough classification of different techniques for the reduced representation of time-series data, grouping them in families and identifying the most representative

techniques in each family. Indeed, the selection of reduction and approximation techniques that are analyzed and compared is similar across various references discussing time series data mining [Fu11], [WMD+13], [EA12a], [PVK+04], [GGB12]. Next, these groups are presented together with some of their most representative techniques.

Sampling based Techniques

This kind of techniques sample the time series taking values from time to time. They divide the time series into several segments (k) from which they obtain a single sample and reduce the time series to k samples taken from the k segments. Despite their simplicity, these methods are quite effective and thus, they are widely used. Probably, the simplest method of this family of techniques is the *Sampling* method proposed in [Ås69]. This method, divides the time series into m/n segments (where m is the length of the time series and n the length to which it will be reduced) and obtains a sample of each of them.

An enhancement was introduced to the Sampling method by using the average (mean) value of each segment to represent the corresponding set of data points. This method, known as *Piecewise Aggregate Approximation* (PAA) [KCPM01] or *Piecewise Constant Approximation* (PCA) [YF00], divides the time series into k segments and obtains the average of each segment to represent the time series, as shown in Figure 3.10.

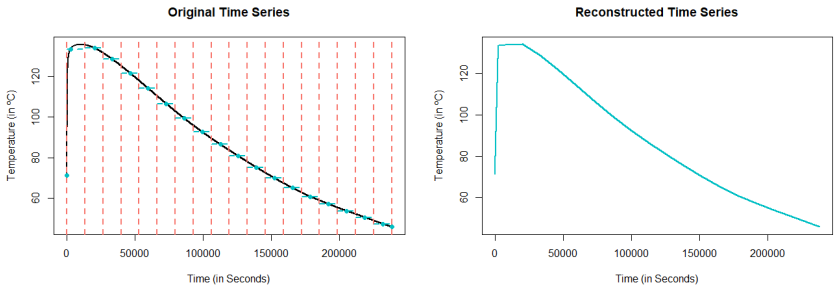


FIGURE 3.10: Time series representation with the Piecewise Aggregate Approximation (PAA) technique

An extended version of this method, called *Adaptive Piecewise Constant Approximation* (APCA) [CKMP02], or *Adaptive PCA*, was proposed, in which the length of the segments was not fixed, but could have a variable length that allow it to adapt more to the shape of the time series. This method also obtains the mean of the segments, but this time the length of the segments is not always the same, as shown in Figure 3.11. To obtain the size of the segments, the problem is converted into a wave compression problem in which the coefficients of the *Haar wavelet transform* of the time series signal are calculated and ordered according to the decreasing normalized magnitude, which allows to truncate the smaller coefficients and approximate them with mean values (see the algorithm in [CKMP02]).

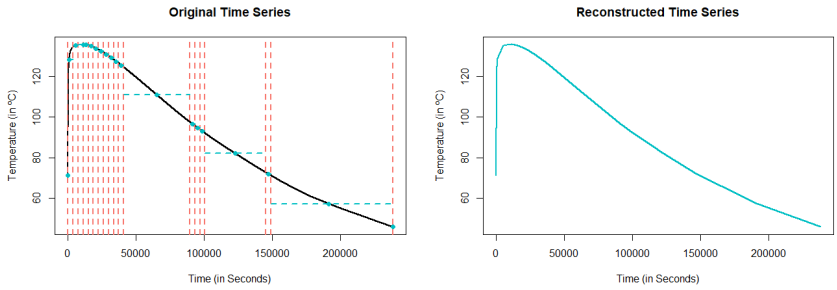


FIGURE 3.11: Time series representation with the Adaptive Piecewise Constant Approximation (APCA) technique

Some other approaches have been also proposed in the literature; for example, in [LKL03], the *Segmented Sum of Variation* (SSV) is used to represent each segment of the time series, and in [RKBL05] and [BRK+06], a bit level approximation is proposed on which a bit is used to represent each data point and then other techniques such as *Run Length Encoding* (RLE) [RC67] are used to represent the data.

Time Series Approximation with Lines

Another approach for reducing the dimensionality of time series was the approximation of the time series using lines. In this approach there are two main categories. The first one is linear interpolarization, on which some methods, such as *Piecewise Linear Regression* (PLR) [Keo97b], [Keo97a] have been proposed. PLR is an intuitive method that offers a high compression ratio and is relatively insensitive to noise when comparing time series. This method begins by approximating a time series S of length n with j linear segments where $j = \lfloor n/3 \rfloor$ (the integer part of the division). This way, all the segments start with at least 3 points (the last segment can have 4 or 5 if there is remainder in the $n/3$ division). Each segment is the line that better fits the segment points collection, determined by the classical regression equation. Once the time series has been segmented into j segments, the PLR works as a *bottom-up* algorithm which begins by joining two consecutive segments to produce a new approximation of the time series with *j-1* segments. The pair of segments to join in each iteration is the one with the lowest B_K value obtained for the next iteration, with B_K being the balance of the error for the k segments (B_K = std $(e_1, e_2, e_3..., e_k)$) and e the vertical distance from each point to the approximated line. This process is repeated until all the segments have been joined in a single linear approximation or until the desired number of segments has been reached.

The second approach is linear regression, which represents the subsequences with the best fitting lines. In [SZ96], a method which consists of dividing the time series into sub-sequences and then adjusting each of those sub-sequences with a function (e.g., linear regression) is proposed. One of

the simplest approaches is to divide the time series into n sub-sequences of the same length (in the same way as in the Sampling method) and adjust each of the sub-sequences using linear regression. However, there are also other methods for dividing the time series into sub-sequences (i.e., time series segmentation techniques) and for adjusting the sub-sequences (e.g., polynomial regression) [SZ96].

Dimensionality Reduction Preserving Salient Points

Another interesting approach to represent a time series is preserving the most significative points of the series. One of the most relevant methods from this approach is the *Perceptually Important Points* (PIP) [FTLN02], [TFLC08], which establishes that a time series has n points $P_1, P_2, P_3, ..., P_n$ that can be reordered due to their importance based on the PIP identification process. The process starts choosing the first and last points in the time series as the first two points of interest. Then, selects the furthest point from these two points, based on the vertical distance from the line formed by the adjacent PIPs, and joins the third PIP into the PIP points set. This process is repeated until the selected number of points is obtained or until all the points of the time series have been reordered. Figure 3.12 shows an example of a time series represented with this method.

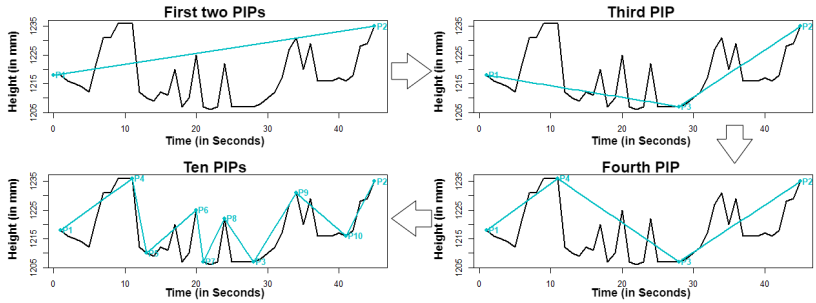


FIGURE 3.12: Time series representation with the Perceptually Important Points (PIP) technique

The idea is similar to other approaches [Fu11], such as the method proposed in [PF02] and [FPG03], that define as extreme points in a time series the minimum and maximum points, and compresses a time series by selecting only certain important points, based on the *extrema* points (the main idea is keeping the highest maximums and lowest minimums and discarding minor fluctuations). A *Critical Point Model* (CPM) [Bao08] method has also been defined for the preservation of critical points (maximum and minimum) within a fixed interval based on a level of oscillation; in [PWZP00], a landmark point model is used to identify the most important points of a time series; in [MW01], a "lattice structure" is proposed to represent the peaks and valleys identified in a time series; in [PHT08], a series of special points are introduced to restrict the error in the PLR method; and lastly,

in [LLTX09], the representation of a time series through a series of key points is suggested.

Symbolic Representation

Another common approach is transforming the numeric time series into a symbolic form. These methods start by dividing the time series into segments and then, discretize the time series by converting those segments into symbols. One of the most widespread and widely used method is the *Symbolic Aggregate Approximation* (SAX) [LKLC03]. This method, first transforms the time series to the PAA representation, and later, discretizes the PAA representation, converting the values of the segments obtained in this representation into symbols that form a string (as shown in Figure 3.13). The symbolic representation is obtained by dividing the distribution space (y-axis) into equiprobable regions, and then, each region is represented by a symbol (from a given alphabet), so that each segment can be mapped into a symbol corresponding to the region in which it resides.

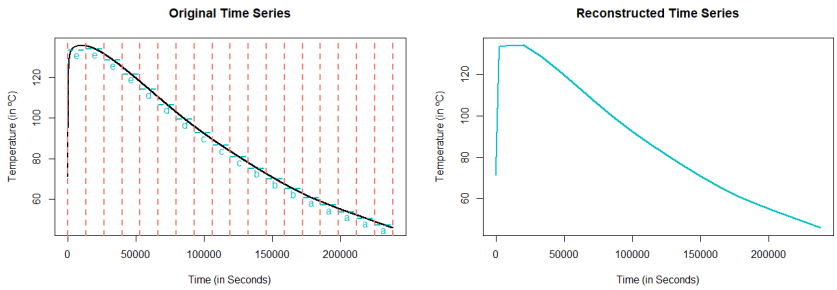


FIGURE 3.13: Time series representation with the Symbolic Aggregate Approximation (SAX) technique

Some other approaches use an alphabet to describe the time series. For example, in [AJB97], a method called *Shape Description Alphabet* (SDA) is proposed, on which symbols such as *"highly increasing transition"* or *"stable transition"* are used to describe a time series. Another proposal similar to the previous one is the one presented in [QWW98], where the use of gradient alphabets, such as *upward*, *flat*, or *download* as symbols is proposed; in [HY99], a method to transform the time series into a symbol string using the rate of change between contiguous points is proposed; and lastly, in [MLW04], a method to represent each segment with a keyword from a *codebook* of key-sequences is proposed. Moreover, other approaches for the transformation of time series to a symbolic form, were those proposed in [DLM+98], [BLJZ00], [HBM01], and [HH01], that generate the symbols by clustering sub-sequences of the time series. Finally, in [LYC98], a method known as *Multiple Abstraction Level Mining* (MALM), is proposed, to transform a time series into a sequence of symbols (a string), by clustering the characteristics of each segment, such as the regression coefficients, the average, etc.

Transformation based Approaches

One of the most popular transformation techniques in time series data mining is the *Discrete Fourier Transform* (DFT). This technique was first used for time series representation in [AFS93]. The discrete Fourier transform [BH95], is a mathematical transformation used to transform signals between the time (or spatial) domain and the frequency domain. This transformation decompose the original time series into sinusoidal components to which it assigns certain coefficients, known as the Fourier coefficients (as already shown in Figure 3.8). In a similar way to when filtering the noise, the dimensionality reduction is performed by choosing the Fourier coefficients of the lowest frequencies and using only those coefficients to represent the time series (an approximation to the original series can be reconstructed with the inverse Fourier transform).

Later, the wavelet transform was used to represent time series in [SS98], as well as the *Discrete Wavelet Transformation* (DWT) was used (along with the Haar transform) for the representation of time series in [KA99], [SS99], and [WW00]. The discrete wavelet transform [Add17], works similarly to the DFT, except that this time, instead of representing the original time series by sinusoidal components transformations, it is represented in terms of translated and expanded versions of a finite wave known as a *wavelet* [Dau92]. Moreover, in [STZ00], a multi-level representation of the wavelet transform is proposed; in [PM02], a large class of wavelet transforms to represent time series is used; and in [KU02], a combination of the Fourier transform and wavelet transform is used for time series representation.

Another approach in the domain of the transformations was the one proposed in [CN04] and [WMD+13], on which *Chebyshev polynomials* are used for the representation of time series. The representation of time series using Chebyshev polynomials works similarly to DFT and DWT based methods, except that the representation is done using Chebyshev polynomials instead of wavelets or sinusoidal components. The Chebysev polynomials are a family of orthogonal polynomials that are related to *De Moivre's* formula and can be defined recursively. In the same manner than the base functions in DFT and DWT, these polynomials have some coefficients associated, known as the *Chebyshev coefficients*, which are the ones used to represent the time series. Lastly, other transformation based approaches have been proposed to reduce multivariate time series dimensionality by using the *Principal Component Analysis* (PCA) to analyze financial time series in [LCL99]; and an enhancement of *Singular Value Decomposition* (SVD) method in [KJF97].

Model-based Approaches

Another group of time series representations methods are those based on a model [LL16]. Different types of models have been used in the literature for representing time series. For example, in [AN98], *Hidden Markov Models* (HMMs) are used for time series representation. *Holt-Winters exponential smoothing* has been used as another method of representation based on

this model [LL16]. Regression models, such as *linear regression* models and *polynomial regression* models [Sti74] have been used for representing time series [Wil17]. *ARMA* models and some of their generalizations (e.g., ARIMA) have also been used for time series representation [Wil17]; for example, in [ZL08], the coefficients of an AR model are used for time series classification in human activity recognition tasks. Moreover, some recent works have used neural network based *autoencoders* for obtaining a reduced representation of time series; for example, in [YTA18], ECG signals are compressed into a reduced representation by using deep convolutional autoencoders; in [WL18], a Recurrent Neural Networks based autoencoder is used to extract a vector representation for multidimensional time-series data; and in [Hsu17], a LSTM-RNN based autoencoder is used for obtaining a compressed representation of time-series data.

3.2.2 Time Series Database Management Systems

The deployment of Industrial Internet of Things devices in Smart Manufacturing scenarios allows to capture large-scale time-series data from the continuous operation of the manufacturing process that are usually stored for further analysis processes. The idea of collecting and analyzing time-series data is not precisely new. Indeed, during the years many people has been storing and processing time-series data in flat files and with general purpose DBMSs (such as RDBMSs), for different purposes (e.g., meteorologic predictions, financial analytics, etc.). However, what is it new, are the massive volumes of time-series data generated on an unprecedented scale.

Nowadays, it is not uncommon to deal with petabytes of data, even when carrying out traditional types of analysis and reporting [DF14]. The proliferation of IoT devices that capture and share large amounts of data over the Internet has played an important role in the explosive availability of such large-scale time-series data, which when properly managed, offer a great potential value. In this regard, although general purpose DBMSs (and in particular RDBMSs) can store time-series data, they are unsuitable to handle the velocity and volume of such large-scale time-series data. Therefore, in order to handle such amounts of data, it is necessary to scale the DBMSs accordingly, through the distribution across several nodes and the possibility to scale further if required [BKF17].

For this purpose, new NoSQL approaches have been proposed, which make use of non-relational databases with considerable advantages in flexibility and performance over traditional RDBMS [DF14]. These NoSQL DBMSs trade away some of the properties and capabilities of traditional RDBMSs (e.g., weaker relations and consistencies, semi-structured data, etc.), in order to provide a solution with more possibilities for distribution and improve scalability [BKF17]. With the new scalable NoSQL platforms and tools for data storage and access, now it is feasible to store large periods of raw time-series data. Furthermore, these NoSQL DBMSs have given rise to a specific type of NoSQL DBMSs for these type of data: Time Series Databases (TSDBs).

A Time Series Database (TSDB[2]) is a database management system optimized for storing and processing time-series data. Time-series data are measurements or events in the form of timestamp-value pairs which can be tracked, monitored, downsampled, and aggregated over time. These measurements may represent sensor data, IoT data, network data, server metrics, application performance monitoring, events, trades in a market, and many other types of analytics data. Those TSDB systems present some key architectural design properties that make them very different from other database systems. These properties include among others: timestamp data storage and compression, the use of time as a key index, time aware data life-cycle management, data summarization and down-sampling capabilities, the ability to handle large time series dependent scans of many records, time series aware queries, and some additional features related to the time series analysis (e.g., interpolation, filtering, etc.) [Inf19b].

During the last years, a vast amount of TSDBs have emerged. A ranking of them based on their popularity can be found in [Sol19]. Moreover, in the survey presented in [BKF17], a comparison of open-source and some proprietary TSDBs is presented. However, despite the vast amount of existing solutions and the wide-spreading of some of them (e.g., InfluxDB has more than 369,303 deployments [Inf19b]), the increasing scale of modern datasets, the velocity of data accumulation, and the variety of data sources still pose a major challenge for their efficient and scalable storage and processing, for which many improvements can be proposed and developed. Regarding the development of new proposals for Time Series Management Systems[3] (TSMSs), in the survey presented in [JPT17], an analysis and classification of TSMSs developed through academic or industrial research and documented through publications is presented. In the following subsections an overview of some of the most popular NoSQL DBMSs and TSDBs used for the storage of time series are presented.

3.2.2.1 NoSQL Database Management Systems for Time Series Storage

In real-world scenarios such as manufacturing environments, the captured time-series data by the large-scale sensor and IIoT devices networks flows continuously into an endlessly accumulating data stream that cannot be stored and managed within a bounded storage space such as traditional RDBMSs [DF14], [JPT17], [SSRG13]. Moreover, time series are usually data that require high concurrency, throughput and writes, but low reads [Zho19]. For such characteristics, it is very suitable to use NoSQL databases as the data storage layer such as, Apache HBase or Cassandra, or cloud services such as, Alibaba Cloud's Table Store [DF14], [Zho19], [MFP+19], [JHGJ11].

A vast amount of NoSQL DBMSs have been developed during the last years to overcome some of the limitations that traditional DBMSs

[2]In this work, TSDB is used interchangeably to refer to Time Series Database Management Systems and Time Series Databases.

[3]TSMS is one of the multiple names commonly used in the literature for referring these systems and is used interchangeably with TSDB.

(such as RDBMSs) present. These NoSQL DBMSs are usually classified according to four kinds of data models (*Key-Value Store*, *Document Store*, *Wide-Column Store*, and *Graph Database*), each of them with different advantages and limitations [DCL18]. Among them, Wide-Column Store and Document Store are the most common ones considered for dealing with big time-series data [MFP+19]. However, it is worth mentioning that in the last years, some authors have seen interesting opportunities in using the potential of graph databases to process and exploit data relationships for querying and extracting information from large-scale sensors networks data [CFMS04], [PMS17], [PMS18]. Next, these three types of NoSQL DBMSs are presented together with one of the most popular DBMS of each type.

Wide-Column Store

A wide-column store is a type of NoSQL database that store data in records, with the ability to hold large numbers of dynamic columns. In a similar manner to a relational database, a wide-column store uses tables, rows, and columns, but unlike a relational database, the names and format of the columns can vary from row to row in the same table. This type of databases have been inspired by Google's Bigtable [CDG+08], in which data are represented in a tabular format of rows and column-families (columns that are logically related to each other and usually accessed together) [DCL18]. A column-family has a flexible schema on which columns can be added or removed at dynamically runtime. Moreover, in wide-column stores, data can be efficiently partitioned by rows (horizontal partitioning) and by column-families (vertical partitioning), which make them suitable for storing huge datasets [DCL18]. There is a broad spectrum of wide-column stores available, a ranking of them based on their popularity can be found in [Sol19], where *Cassandra* appears as the most popular one.

Cassandra [The19] is an open-source distributed wide-column store, designed to handle large amounts of data across many commodity servers, providing high availability without a single point of failure. For querying and managing the data stored in the database, the Cassandra Query Language (CQL) [Cas19] is used. Although Cassandra is not a purpose-built database for time-series data, the underlying storage mechanism in Cassandra seems to be compatible with time-series data. Cassandra uses a Log Structured Merge Trees (LSM-Trees) [OCGO96] to implement a highly efficient storage backbone per column-family [DCL18]. In addition, Cassandra presents some interesting features that have encouraged people to use Cassandra to store their time-series data [Sch18], [RSS16]. For example, it accepts rapid writes; it scales to really large clusters with lots of disk space and processing capabilities; and it offers optimizations for storing and retrieving temporal data, such as reordering data according to the timestamp key or grouping data with different granularities.

There are two main approaches for storing time-series data in Cassandra (an example of each of them can be found in Figure 3.14). Storing time-series data as a column-family with timestamps as columns could

be the most intuitive way to do it; however, this is not a really scalable schema for large time-series data, and thus, Cassandra offers the possibility of sharding the data according to different granularities (e.g., by minute). Moreover, while declaring a column-family, composite primary keys can be declared. This kind of keys involve n attributes on which the first n - 1 attributes form the *partition-key* (used for partition data across nodes), and the last attribute forms the *clustering-key* (used to sort data within a node). Using the *time-key* as the *clustering-key*, sorts data by time, favouring the retrieval tasks based on time ranges [MFP+19].

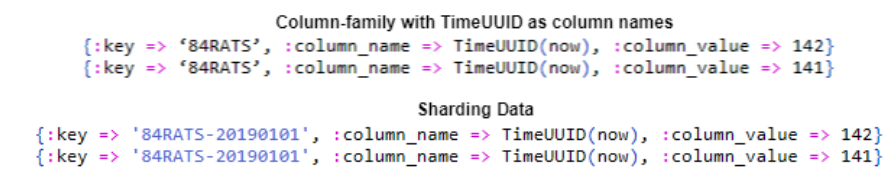


FIGURE 3.14: Cassandra approaches for time-series data storage

Document Stores

A document store is a type of NoSQL database that has been designed to store and query data encoded in standard semi-structured formats such as XML or JSON. In a document store database, each record and its associated data is stored within a single document. A document has a flexible schema on which its attributes (pairs of a name and one or more values) can be added or removed at run-time [DCL18]. These can be seen as extended *key-value stores* in which the value is represented as a document. The flexible, semi-structured and hierarchical nature of documents and document stores allow them to evolve according to the needs of the applications. Moreover, document stores make it easy for developers to store and query data in a database using the same document model format that they use in applications. Each document contains semi-structured data that can be queried against using various query and analytic tools of the DBMS. There are multiple available document store databases, a ranking of them based on their popularity can be found in [Sol19], where *MongoDb* appears as the most popular one.

MongoDB [Mon19a] is an open source, document-oriented distributed database designed with both scalability and developer agility in mind. MongoDB stores BSON (binary JSON) documents with dynamic schemas and uses JavaScript as the main query language [Mon19b]. MongoDB provides really interesting features and capabilities for meeting the demands of a highly performing time series applications. For example, it stores records composed of key-values pairs contained in BSON documents (a format often used for time series storage [RSS16]); it allows to scale horizontally through sharding with some optimization for temporal data, such as data distribution across nodes by a defined key value based on time [MFP+19]; it allows pre-aggregating data; and it provides data retention

and archival policies [Wal19]. Moreover, its flexible data model allows to optimally bucket data to yield the best performance and granularity for an application's requirements. This way, different schemas can be designed for different purposes. Figure 3.15 shows two different schemas with different granularities. On the first one, a document is stored for each time-series data point, while on the second one, time-series data are grouped into a bucket by each minute.

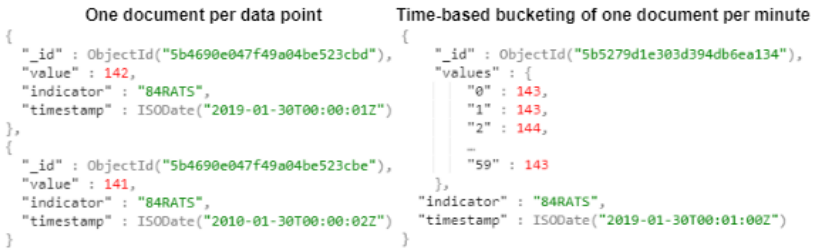


FIGURE 3.15: MongoDB schemas for time series storage

As a result, MongoDB has been widely adopted by researchers and practitioners for the storage of time-series data. For example, in [RSS16], authors report a successful adoption of MongoDB for the storage of discrete time-series data. Moreover, some companies, such as Man AHL's (in their *Artic* application), Bosch (in their IoT suite) or Siemens (in their *Monet* platform) have took advantage of MongoDB's time series handling capabilities for developing different applications.

Graph Stores

A graph database is a database that uses graph structures with nodes, edges, and (sometimes) properties to represent and store data. The graph structure, relates the data items in the store to a collection of nodes and edges, on which nodes represents items (records of data) and edges the relationships between nodes. Some graph databases also add the concept of properties, tags or labels; which are essentially, relationships grouping nodes into sets. A graph database leverages mathematical concepts from graph theory and uses them as a powerful and efficient engine for efficiently querying and processing not only data, but also relationships [PMS18]. Graph databases are part of the NoSQL databases emerged to address some of the limitations of the existing RDBMs. By design, graph databases allow simple and fast retrieval of complex hierarchical structures that are difficult to model in relational systems. The underlying storage mechanism of graph databases can vary; some of them depend on a relational engine, whereas some others use a *key-value store* or a *document-oriented database* for storage, making them inherently NoSQL structures. There exists several graph databases available, however, according to the ranking presented in [Sol19], the most widely used one is Neo4J.

Neo4J [Neo19b] is a highly scalable native graph database management system purpose-built to leverage not only data but also data relationships. Cypher [Neo19a], is the graph query language used to store and retrieve data from the Neo4j's graph database. Neo4j database has already been used by other authors to collect, store and analyze large amounts of data coming from sensors [PMS18]. For representing time-series data, Neo4J creates a similar graph structure to the one outlined in Figure 3.16.

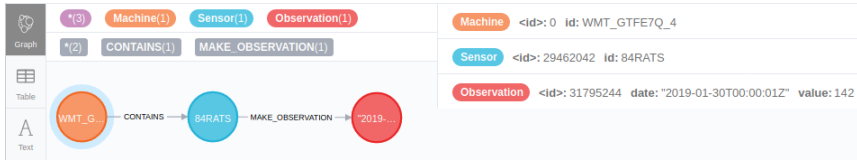


FIGURE 3.16: Neo4J schemas for time series storage

3.2.2.2 Time Series Database Management Systems

Within those NoSQL DBMSs a vast amounts of TSDB solutions have been developed. In the survey presented in [BKF17], a review of 50 open-source TSDBs is provided, on which the authors groups those TSBDs into three main groups. In the first group, there are those *TSDBs that depend on an existing DBMSs* (such as Cassandra, HBase, etc.) for the time-series data storage (those TSDBs requiring another DBMS to store metadata or other types of data that are not time series, are not considered in this group). In the second group, there are those *TSDBs which does not require any other DBMSs* for the storage of the time-series data (although they may require other DBMS for the storage of metadata or additional information). Lastly, in the third group, those *TSDBs that use RDBMSs* for the storage of the time-series data are considered. Moreover, authors also consider an additional (*Proprietary*) group with regard to those TSDBs commercially or freely available that are not open-source.

Among those 50 open-source TSDBs, authors selected the 12 most prominent ones to compare them under the perspective of 27 different criteria, grouped in six criteria groups: (i) *distribution/clusterability*, with regard to high availability, scalability and load balancing features; (ii) the supported *functionality* by the TSDBs; (iii) whether if the TSDBs support *tags, continuous calculations* (e.g., compute an average hourly), *long-term storage,* and *matrix time series* (i.e., time series that require two or more timestamps per value); (iv) *granularity*, with regard to the smallest possible granularity that can be used for time series storage and processing functions, and the ability of *down-sampling* requested results; (v) the offered *interfaces* (such as APIs, interfaces, client libraries, etc.), and *extensibility* (e.g., possibility of using plugins for extending the TSDB capabilities); and finally, (vi) the availability of a *stable version and commercial support*, and the used *license*.

The selected 12 TSDBs for the comparison corresponds to the five most popular TSBSs from the first two categories and the two most popular

ones from the third category. Some of the most prominent examples from that comparison are: *KairosDb* [Kai15] (a TSDB which uses Cassandra as a storage for time-series data), and OpenTSDB [Ope18] (a scalable TSDB which runs on Hadoop and HBase), under the category of TSDBs that depend on an existing DBMSs; *Druid* [Dru20], *MonetDB* [Mon08], *Prometheus* [Pro14] and *InfluxDb*, under the category of TSDBs which does not require any other DBMSs; and MySQL Community Server [Ora20] and PostgreSQL [Pos19] under the category of TSDBs that use RDBMSs. Furthermore, these TSBDs also appear between the most popular ones in the ranking presented in [Sol19],[4] on which InfluxDB is clearly the most popular TSDB (see Figure 3.17). Next, some properties of this TSDB are detailed.

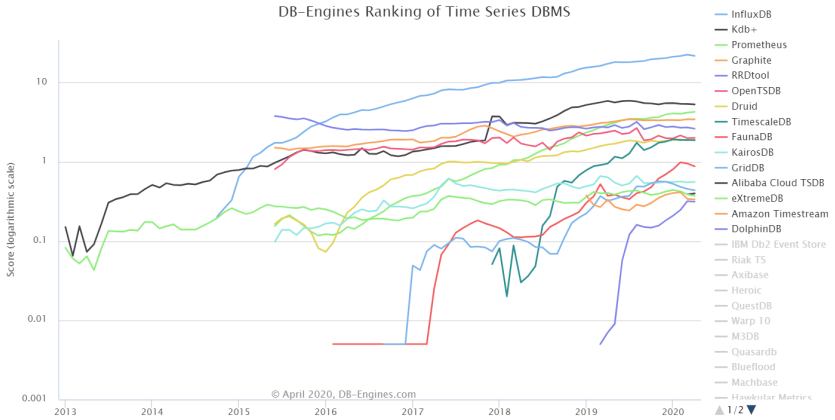


FIGURE 3.17: Time series database engines popularity trend (extracted from [Sol19])

InfluxDB [Inf19b] is an open-source TSDB optimized for fast, high-availability storage and retrieval of time-series data. InfluxDB is written in Go and has no external dependencies. It is used for monitoring and recording performance metrics and analytics in different fields such as operations monitoring, application metrics, IoT and sensor data, real-time analytics, etc. The data structure of an InfluxDB database is composed by points. A point has four components: a *measurement*, which is the way to associate related points that might have different *tagsets* or *fieldsets*; a *tagset*, a dictionary of key-value pairs to store metadata with a point; a *fieldset*, a set of key-value pairs (the data being recorded by the point); and a *timestamp*. The serialization format for points is defined by the *line protocol*. Figure 3.18 shows the syntax of the line protocol together with a point example.

In InfluxDB, each point is stored within a *database*, within a *retention policy*. A database is a container of *retention policies*, *points* and *users*. A retention policy configures the points *duration* (how long InfluxDB keeps

[4]Leaving aside those TSDBs that use RDMSs (which not appear under the TSDBs category of the ranking presented in [Sol19]).

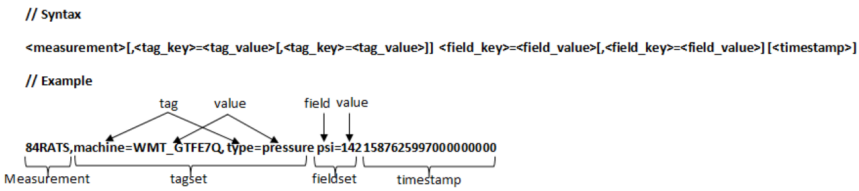


FIGURE 3.18: Line Protocol syntax and point example

points), the *replication factor* (how many copies of those points are stored in the cluster) and the *shard group duration* (determines how much time each shard group spans). The retention policy allows to drop, older data that are no longer needed, in an easy and efficient way. In InfluxDB, all points with the same retention policy, measurement, and tagset are members of the same *series* (a series is a shortcut for saying retention policy + measurement + tagset). The storage engine inside InfluxDB is composed by four main elements: the Write Ahead Log (WAL), which retains InfluxDB data when the storage engine restarts (it ensures data are durable in case of an unexpected failure); the *cache*, an in-memory copy of data points currently stored in the WAL; the *Time-Structured Merge Tree* (TSM), on which *TSM files* store compressed series data in a columnar format; and a *Time Series Index* (TSI), which ensures that queries remain fast as data cardinality grows by storing series keys grouped by measurement, tag, and field. For interacting with data in InfluxDB, InfluxQL [Inf19a], a SQL-like query language is used.

3.2.3 Artificial Intelligence in Manufacturing

Smart Manufacturing aims to use modern information technologies for transforming the captured data during the manufacturing process into manufacturing intelligence, in order to achieve meaningful improvements in all aspects of manufacturing [TQLK18]. The captured manufacturing data contains valuable information and knowledge that could be extracted and integrated within manufacturing systems to enhance productivity and improve decision making [CHT09]. Thus, the captured data are usually stored in manufacturing databases for further analysis and exploitation purposes. However, the massive amounts of data in manufacturing databases (usually in the form of large time-series data) that need to be simultaneously analyzed in order to discover and extract useful information and knowledge, make manual analysis unfeasible [CHT09]. Consequently, more intelligent and automatized data analysis methodologies are required.

In this regard, Knowledge Discovery in Databases (KDD) and data mining have become extremely important to achieve the objective of intelligent and automated data analysis, leveraging tools from other related fields, such as machine learning and artificial intelligence, to boost multipurpose data-driven applications. This section shows, first, some of the different data mining methods applied in the manufacturing context for

extracting knowledge and useful information from the data; next, some of the different applications to which these methods are targeted; and finally, some of the most popular techniques for developing these applications.

3.2.3.1 Data Mining Methods in Manufacturing

In addition to constituting the analysis step of the KDD process, data mining is an interdisciplinary field with the overarching goal of extracting information (with intelligent methods) from a dataset, and transform the information into a comprehensible structure for further use. It makes use of intelligent methods and automated tools, to discover hidden patterns, associations, anomalies and interesting structures from massive amounts of data stored in a data repository. Data mining tasks can be generally classified into two types based on the specific goals they aim to achieve: descriptive tasks and predictive tasks. Descriptive data mining tasks aim to discover interesting patterns to describe the data, while predictive data mining tasks aim to model the behaviour of the data, by using existing information from some variables, to predict and determining unknown or future values of other variables of interest [CHT09]. However, it is worth mentioning, that the boundaries between them are not really sharp (some predictive models can be descriptive, to the extent that they are understandable, and vice versa) [FPSS96]. The goals of prediction and description can be achieved by using a variety of particular data mining methods.

Based on the kinds of knowledge that can be discovered in the databases, data mining methods can be broadly structured into several categories which typically include: classification, regression, clustering, association and summarization [Fay96], [KBT11]. However, in the literature, specific categories from specific fields of research are also included, such as time series analysis [SA07]. Figure 3.19, presents a classification of some of the main data mining methods and next the most widely used ones in the manufacturing context are presented:

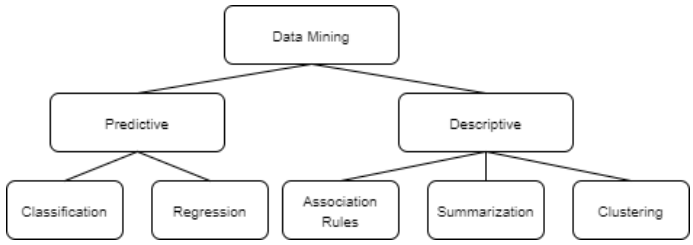


FIGURE 3.19: Classification of the main data mining tasks

Classification in Manufacturing: Classification is a predictive task which consists in learning a function that classifies (maps or associates) a data item into one of several predefined classes (groups or categories) [FPSS96]. In general, classification is performed in two steps. In the first step (training), a model is built using a classification algorithm that analyzes the training database objects (i.e., instances or tuples with different

variables) and finds relationships between the values of the predictor variables and the values of the target variable (i.e., label or class). These relationships are summarized in a model, which can be represented in different forms, such as classification rules, decision trees, or mathematical formulas. In the second step (testing), the model is used for classification of a different dataset (e.g., test dataset) in which the class assignments are unknown. Classification is a useful functionality in many areas of manufacturing [CHT09]. For example, for the classification of a quality characteristic or predicting the product quality [KBT11], for the classification of defects and faults to find failure patterns and derive rules to avoid them [CHT09], etc.

Clustering in Manufacturing: Clustering is a common descriptive method which consists in identifying a finite set of categories or clusters to describe the data [FPSS96]. In clustering, data objects (instances or tuples) have not a class (i.e., label) associated or it is unknown *a priori*. In general, in clustering algorithms, training data is partitioned into several clusters, where clusters are natural groupings of data items based on similarity metrics or probability density models. This way, clustering maps the data items in the training dataset into one of several clusters in a manner on which data objects within the same cluster are close to each other (intra-cluster similarity), but they are far from the data objects in other clusters (inter-cluster dissimilarity). Clustering contributes a variety of benefits to manufacturing, such as customer and market segmentation, optimizing order picking in logistics and supply chain [CHT09], production planning and scheduling, etc.

Prediction in Manufacturing: Prediction (or regression) is a common predictive task which consists on learning a function that maps a data item to a prediction variable. In the same way of classification, prediction involves developing a model based on the available data and then, this model is used for predicting future values of a new dataset of interest. In the model building process (training), a regression algorithm estimates the value of the target variables as a function of the predictor variables for each item in the training dataset. These relationships between predictor and target variables are summarized in a model, which can then be applied to a different dataset in which the target values are unknown. The main difference between classification and regression, is that the output variable in regression is numerical (or continuous) while in classification is categorical (or discrete). Prediction in manufacturing processes is crucial for predicting product quality, predictive maintenance of equipment, fault prognosis, etc.

Association in Manufacturing: Association rule learning (dependency modeling) is a descriptive method which consists in finding a model that describes significant dependencies between object variables. It is used to identify relationships between a set of items in a database. These relationships are not based on inherent properties of the data themselves (as with functional dependencies), but rather are based on co-occurrence of the data items. Association rules between requirements and constraints could provide additional useful information for the design context [CHT09]. For

example, in [SSHT06] association rules are applied to extract useful information in the context of fan blade manufacture, where the extracted rules contain very interesting information related to the geometry of the product.

Summarization in Manufacturing: Summarization (or concept description) is a descriptive method which aims to describe a dataset in a compact and concise summary that presents the most interesting properties of data. For that purpose, simple techniques, such calculating mean and standard deviations for all fields are often applied, although there are also more sophisticated methods which involve, the derivation of summary rules, multivariate visualization techniques, and the discovery of functional relationships between variables. These techniques are often applied for interactive exploratory data analysis and automated report generation [FPSS96]. In most manufacturing problems, it is necessary to view the summarized data in a descriptive and concise form, which provides an overall picture of the manufacturing domain data [CHT09].

These data mining tasks are usually involved in most of the data-driven Smart Manufacturing approaches. For example, in this research work, with the aim of performing a predictive maintenance of the equipment, classification and prediction tasks have been performed in order to predict the future measurements of the sensors implanted in an extruder machine and detect whether an alarm will be activated or not. On the other hand, with the aim of recommending the most suitable dimensionality reduction techniques for heterogeneous time series coming from multiple sensors, clustering tasks have been performed for grouping time series according to properties they share, that make them susceptible to be reduced with a particular selection of reduction techniques. Furthermore, in addition to these applications, data mining tasks are involved in a wide-spectrum of applications in the context of Smart Manufacturing. In the following subsection, some of the most common ones are presented.

3.2.3.2 Artificial Intelligence in Manufacturing: Applications

With the aim of transforming the captured data during the manufacturing processes into manufacturing intelligence, Smart Manufacturing adopts methods from data mining and other related fields, such as artificial intelligence and machine learning, to boost data-driven smart applications. These applications can yield meaningful improvements in different aspects of manufacturing, such predictive maintenance of equipment, fault diagnosis, product quality and process efficiency control, etc. Some of the promising applications that can be implemented during the manufacturing process are presented next.

Product Quality Control: The advances in IIoT and synergic technologies allow to capture massive amounts of product quality data from a variety of sources (e.g., sensors, RFIDs, machine vision applications, etc.). The captured data may include different properties of products and manufacturing processes during the whole product life-cycle, such as geometric properties, raw materials, location data, and machining parameters.

A proper analysis of these data can serve the overall quality monitoring, early warning of quality defects, and rapid diagnosis of root causes [KBT11]. Thus, product quality defects can be detected, diagnosed, and addressed in a timely manner [TQLK18]. As a result, not only low quality or products with defects can be automatically identified and removed, but also factors that result in quality defects can be identified and controlled to improve production processes. Different works have addressed quality control in manufacturing processes, for example, in a similar scenario to the one considered in this work, in [GSRP+18], different regression models are used for predicting the product quality, based on the predicted diameters of the extruded tubes, in order to optimize their production processes. Other works in product quality control are mentioned in [TQLK18]; and in [KBT11], a review of data mining applications for quality improvement in manufacturing industry is provided.

Smart Equipment Maintenance: In manufacturing contexts, equipment maintenance plays an important role, and directly affects the service life of equipment and its production efficiency [WTL+17]. As a consequence, manufacturers are introducing different IIoT devices for monitoring the production process and the equipment status, that allow to evaluate the health condition of manufacturing equipment. These devices allow to capture data from different machines and production processes, which include among others, device alarms, device logs, and device status. The captured data enables data analytics that allow to predict and diagnose equipment faults [WDH18] and remaining useful lifetime [SWHZ11], for a proactive maintenance of it [WTL+17]. In turns, a proactive maintenance of equipment allows prolonging equipment life and minimizing maintenance costs. Therefore, predictive maintenance of equipment is one of the most outstanding Smart Manufacturing applications and has attracted the interest of many researchers and practitioners who are developing different research efforts. A review of the state of the art on predictive maintenance of equipment can be found in [Has11]. Moreover, in [WTL+17], a big data solution for active preventive maintenance in manufacturing environments is proposed and, in [TQLK18], some data-driven approaches are presented for smart equipment maintenance.

Smart planning and process optimization: Production planning is crucial for the scheduling of the manufacturing processes in a production plant, as well as to determine the production capacity and the availability of materials and resources. In this regard, Big Data analytics and synergic technologies can make production planning more intelligent and efficient, by analyzing manufacturing data. On the one hand, data such as inventory data, customer orders, production capacities and resources data, enable a better analysis of the supply and demand relationship for more efficient management of manufacturing resources. On the other hand, the analysis of the data from the supply chain and manufacturing processes allow to evaluate and optimize technological processes. The analysis of these data allows to determine the correlation between different technological parameters and the effect of these parameters on production performance and quality. This way, technological processes can be adjusted in relation to

these parameters, in order to improve productivity and product quality, as well as reduce production costs. Production optimization is one of the applications of data mining and related fields, to which manufacturers are devoting important efforts; a review of them can be found in [CHT09].

Fault Detection and Diagnosis: Manufacturing systems are often subject to failure caused by degradation or abnormal operating conditions [WMZ+18], which have negative impacts in the production process, such as excessive load, bad quality products, machinery damages, overheating, or machine down-times. Consequently, these failures usually incur in lower productivity, higher operating costs, unplanned production stops, defects in products, etc. Therefore, in order to implement Smart Manufacturing, it is crucial to implement Fault Detection and Diagnosis (FDD) methods. FDD methods aims to detect the occurrence of a failure as early as possible and diagnose the root cause of failures and their type as accurately as possible. There are two main approaches for FDD, simple approaches, such as direct pattern recognition of sensor readings that indicate a fault, and model-based methods which were proposed to overcome the difficulties of the application of simple approaches to deal with massive amounts of manufacturing data. The model-based approaches involve building rigorous process models derived from the captured data and from the knowledge of the manufacturing processes. Several works have addressed methods for FDD through model-based approaches; a variety of them are reviewed in [DG13], and various deep learning models based approaches for FDD are reviewed in [WMZ+18] and [IMDK19].

The aforementioned applications of data-driven approaches for Smart Manufacturing yield important improvements for the performance of a manufacturing plant. Among them, predictive maintenance of equipment stands out, since it directly affects the service life of equipment and its production efficiency. In this regard, in this work an alarm prediction system that is able to predict different types of alarms that can be produced on a real manufacturing plant is proposed. Moreover, during the last years the rapid developments in machine learning techniques, which allow automatic and intelligent analysis of the massive amounts of data captured in manufacturing scenarios, have paved the way for the development of more and more data-driven smart applications. In the following subsections some of these techniques are presented.

3.2.3.3 Machine Learning in Manufacturing

Machine learning techniques are presented as a valid candidate to overcome some of the major challenges related to the analysis of large datasets generated by complex manufacturing systems. These data-driven approaches are able to find patterns in data and extract relationships among them, that are represented into data structures, so-called models, which are then applied for prediction, detection, classification, regression or forecasting [WWIT16].

Machine learning is a sub-field of Artificial Intelligence studying computer algorithms that automatically learn and improve from experience without being explicitly programmed [Sam59]. Machine learning methods

enable computers to build a data-driven models automatically through a systematic discovery of statistically significant patterns in an available dataset usually known as *training data*. The built model learns automatically to find patterns and relationships in the training data, in order to make predictions or decisions without being explicitly programmed for it. Many different machine learning algorithms have been proposed. According to the tasks or problems to solve, these algorithm are usually classified according to three learning paradigms: *Supervised Learning*, *Unsupervised Learning* and *Reinforce Learning*. However, it is worth mentioning that some authors also consider Semi-supervised Learning as a fourth learning paradigm, which falls between unsupervised learning and supervised learning. Next, these paradigms are presented:

- *Supervised Learning* is a machine learning paradigm that learns from *labeled* data (data which has already been tagged with the correct answer). This paradigm, attempts to discover mappings between input data (i.e., independent variables) and target data (i.e., dependent variables). Those mappings are represented in a structure referred to as a model. The model basically consist of a machine learning algorithm "taught" during a training process, to learn, from a training dataset, the relationships between the input data and the target data, so that when new input instances are feed into the model it knows which target to predict. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. Supervised learning tasks can further be categorized in *classification* or *regression* problems.

- *Unsupervised Learning* is a machine learning parading where there is no need to supervise the model with labeled data, instead, the model work on its own to discover information from the data. It studies how to infer a function capable to describe a hidden structure from unlabeled data. Unsupervised learning algorithms take a set of data that contains only inputs (without output or target data), and find structures in the data, in the form of groups or clusters of data. This kind of algorithms usually perform some kind of clustering or associative rule learning.

- *Reinforce Learning* is a machine learning paradigm on which algorithms or models are trained using a system of reward and punishment. A reinforcement learning model receives rewards by performing correctly and penalties for performing incorrectly. It, learns by interacting with its environment by maximizing its reward and/or minimizing its penalty.

- *Semi-supervised Learning* is a machine learning paradigm that falls between unsupervised learning and supervised learning. It combines both, labeled data with unlabeled data, to design and train algorithms that take advantage of such a combination.

Within these paradigms many different algorithms have been proposed to solve different data mining and machine learning tasks. Different reviews and surveys of the most popular ones can be found in the literature. For example, in [STS16], some of the most popular machine learning algorithms for supervised learning are reviewed; in [RW05], a survey of clustering algorithms is provided, which is the most popular method in unsupervised learning; and in [KLM96], a survey of methods and algorithms for reinforce learning is provided.

These techniques have been widely used to solve data mining tasks in many different fields, including, among others, time series data mining [Fu11], [EA12a] and data mining of manufacturing data in industrial contexts [CHT09]. For example, in [XPK10] and [ASW15], a survey of methods and approaches for time-series data (sequence data) classification and clustering are presented (respectively); and in [WWIT16], a review of machine learning techniques in the context of manufacturing is presented with examples of successful applications in a manufacturing environment. In this work, when building the proposed solutions, different machine learning algorithms have been used for different tasks, such as clustering, regression and classification. A taxonomy of them can be found in Figure 3.20.

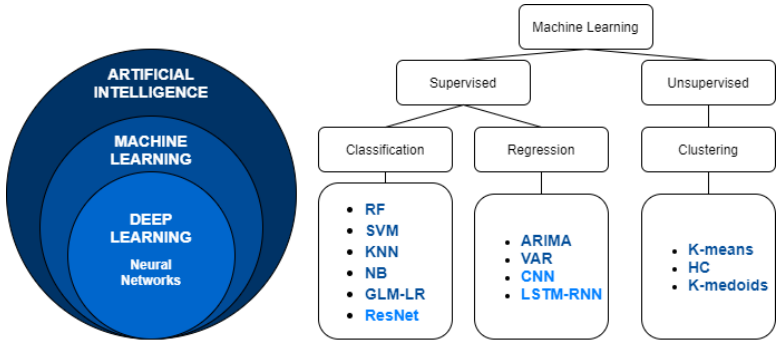


FIGURE 3.20: A taxonomy of the used machine learning algorithms in this research work

3.2.3.4 Deep Learning in Manufacturing

As shown in previous subsection, traditional machine learning techniques have achieved an impressive success in developing different data-driven applications for the manufacturing context [WWIT16]. However, the proliferation of IIoT devices and the recent advances of related technologies such as Big Data has led to massive amounts of manufacturing data (which are essential for approaches that use artificial intelligence techniques [LOD18]), that hamper traditional machine learning algorithms performance and difficult the knowledge extraction. Therefore, more advanced analytics tools are required.

As a breakthrough in artificial intelligence and machine learning, in the last years, *deep learning* has demonstrated outstanding performance

in various applications managing big volumes of data [WMZ+18], such as speech recognition, image recognition, natural language processing, etc. Deep learning is part of a broader family of machine learning methods based on artificial neural networks. Indeed, most of deep learning models are based on deep neural networks (i.e., artificial neural network with multiple layers between the input and output layers) that learn from large amounts of data. Deep learning uses representation learning, also known as feature learning, to map input features (similar to prediction variables in traditional machine learning models) to target variables [GT19]. This mapping process occurs inside multiple connected layers which each contain multiple artificial neurons. Each neuron is a mathematical processing unit, which combined with other neurons, is designed to learn the relationship between the input features and the output values [GT19]. For a detailed description of what neural networks are and how they learn see [SBS+20] and [GBC16].

The automatic feature learning and high-volume modelling capabilities of deep learning, provide advanced analytics tools [WMZ+18] for the massive amounts of manufacturing data captured by sensors and IIoT devices monitoring the manufacturing process. This, together with the increasing popularity of deep learning, has promoted the application of deep learning techniques to the manufacturing data and many researchers are advocating for their use to boost data-driven applications in Smart Manufacturing scenarios. A review of them can be found in [WMZ+18], where authors group different deep-learning based applications according with some of the most relevant functionalities of data-driven applications for Smart Manufacturing.

For building these data-driven applications, usually different models based on deep neural network architectures are built and trained. The neural network model and its architecture determine how the network transforms input data into an output. A vast amount of neural network architectures have been proposed in the literature powering different data mining tasks in different domains; a review of them can be found in [SBS+20]. In this regard, it is worth mentioning, that although most of these architectures can be adopted to different domains, depending on the application domain, some of them seems to be more suitable than others. For example, in the domain of computer vision, Deep Convolutional Neural Networks (CNN) have arisen an impressive success during the last years, whereas in the domain of natural language processing, Recurrent Neural Networks (RNNs) seem to be appropriate for different tasks related with text and sequence prediction.

With respect to time-series data, which is the main type of data in Smart Manufacturing scenarios, different neural network architectures have been proposed and adopted in the literature for both time series classification and time series prediction tasks. With regard time series classification tasks, different neural networks-based architectures have been widely used. A review of them can be found in [WYO17] and [IFFW+19] where authors implements various classifiers based on different neural network architectures and test them over different time series databases. Among the

built classifiers, those based on Fully Convolutional Networks and Residual Neural Networks stand out for achieving the best performance. Concerning time series prediction, Recurrent Neural Networks are typically used to solve tasks related to time-series data [Nvi20]. Different RNNs based architectures have been proposed in the literature, however, LSTM RNNs provide better performance compared to other RNN architectures [Nvi20] and thus, these are the most widely adopted architectures. In this work, different neural networks-based models have been built for predicting the activations of several alarms in a extruder machine. First, a LSTM-RNN based model has been built to predict the future sensor's measurements and then, distinct Residual Neural Networks have been used to determine whether the predicted measurements will trigger an alarm or not.

Chapter 4

A System that Efficiently Guides a Data Engineer in Time Series Pre-processing

Many manufacturing companies are seeing the interest in the adoption of Smart Manufacturing approaches to provide not only equipment, but also value-added, data-enabled services to their customers [NBI15]. The adoption of these approaches must be accomplished by meeting two main requirements: the compilation of manufacturing data and the application of manufacturing intelligence to those data [DEP+12]. However, the captured data must be first pre-processed, in order to obtain a master (pre-processed) dataset on the top of which knowledge discovery techniques can be applied for building those services.

Indeed, in general, dealing with the raw time-series data captured by industrial sensors operating in real-world factories introduces several inefficiencies that may impede their later use. On the one hand, these raw data come with noise and outliers (wrong measurements), which must be filtered out and with missing values (errors in measurement or lost in transmission), which must be filled in. On the other hand, in many cases industrial machine controllers are programmed in an inefficient way in terms of capturing data for analytical purposes. Sometimes, they may be sending a constant value for several hours, for example, to indicate that the machine is operating in manual mode, but these data are captured and stored anyway, occupying unnecessarily an important space that increases data storage costs. In this regard, data reduction techniques could represent a resource with potential to reduce data storage costs.

This chapter presents the first main contribution of this research work which consists in the design and development of *a system that efficiently guides a data engineer in the task of pre-processing raw time-series data* coming from industrial sensors. The proposed system is available as a visual-interactive Web application that provides various functionalities for facilitating the time-series data cleaning and dimensionality reduction

tasks. The main novelty of the proposed system resides in the development of a machine learning-based model that given a time series, recommends the most appropriate reduction techniques that allow obtaining an adequate reduced syntactic representation of raw time-series data, while preserving their main characteristics. Moreover, when dealing with those reduced representations, storage and transmission costs can be decreased, without limiting the future exploitation of the data in different processes.

However, as it is shown in [VVD+18], the performance of the reduction techniques improves when dealing with time series that have already been pre-processed (cleaned). Therefore, before obtaining the reduced representations of the time series, they must previously go through a pre-processing step; and thus, the proposed system also incorporates a *Pre-processing Service*. This service, not only offers a wide-spectrum of techniques for detecting and handling outliers, removing noise and imputing missing values in time series; but also provides some recommendations for the selection of appropriate techniques and the parameter values required by them. This service is also presented as aside contribution of this research work.

The chapter starts with an analysis of related work with respect to time series pre-processing; next, details of the considered pre-processing techniques and time-series data are provided; then, the steps followed for building the machine learning-based model that recommends the most suitable time series dimensionality reduction techniques are presented, as well as some performance results and an analysis of the built model; afterwards, an overview of the system architecture and the main elements involved in it, along with a demo of the built system are shown; and lastly, the conclusions of the realized work are presented.

4.1 Analysis of Related Work

There is an increasing interest among manufacturers in exploiting the potential of large volumes of manufacturing data for diverse exploitation purposes, such as the control of product quality, the predictive maintenance of equipment, etc. However, as already mentioned in Chapter 3, Section 3.2.1, in order to extract useful information and knowledge from those data, they must be first pre-processed to transform the captured raw data into quality data for accurate data analysis. Data pre-processing in Smart Manufacturing scenarios is a complex process, which involves different tasks necessary for example to clean data, reduce their dimensionality, etc.

In fact, the pre-processing of massive amounts of data, represents one of the major challenges in the field of Big Data [OLBO15a], mainly due to the multiple existing alternatives to treat specific problems, the diversity of treatments that each data may require, and the lack of structured methodologies and automatic approaches to determine which techniques to apply in each situation. Therefore, in recent years, the automatic selection of the most suitable techniques for the time-series data pre-processing process has been stated as a general claim, by different researchers and practitioners [KLTCM20], [DWS+19].

In this sense, the first contribution of this research work aims to facilitate the pre-processing task of the different time-series data captured in Smart Manufacturing scenarios, through an automatized approach that helps in the selection of the most suitable reduction techniques to apply to each time series. However, it has been observed that the performance of the reduction techniques improves when dealing with time series that have already been cleaned, and thus, the proposed system also provides different cleaning techniques and some guidelines on which techniques to apply to each time series.

Regarding time series reduction techniques, the inefficiency of storing large volumes of raw time-series data has motivated the development of different techniques for producing a reduced representation of them. Various studies have addressed the application of reduction or approximation techniques to time-series data. A review of them can be found in the technological background presented in Chapter 3, Section 3.2.1.2, on which different dimensionality reduction techniques are reviewed and grouped by families. Nevertheless, no reference has been found where it is determined which reduction techniques are the most suitable ones for each data type, that could be interesting in application scenarios such as the Industry 4.0, where the intrinsic heterogeneity of the sensors in each manufacturing process [NHMG20] leads to time-series data of very different nature, susceptible to be reduced by different reduction techniques, and with diverse reduction potential. In fact, the development of methods and systems, *"that interactively guide users to select appropriate approximations of time-series data with a pre-specified error bound without the need for an exhaustive search,"* has been stated as a future research direction in Time Series Management Systems (TSMSs) in the survey presented in [JPT17].

Furthermore, in that survey, authors also provide their vision for a next-generation of TSMSs and a summary of research directions proposed by other researchers in the field for new storage methods. Among those research directions proposed by other authors, Cuzzocrea in [Cuz15], highlights the development of new storage systems with high focus on scalability. In this regard, the authors of the survey, propose a distributed TSMS with a physical layer storing approximated representations of the time series using mathematical models. These models, in addition to reduce data storage space, provide functionalities for data analysis such as interpolation or forecasting and favor the reduction of query processing time in TSMSs. In that research direction of proposing new storage methods, the system presented in this chapter makes it possible to obtain approximated (reduced) representations for raw time series, using data reduction techniques.

With respect to the dimensionality reduction techniques considered in the system, although some lossless compression techniques are also considered, such as the Run Length Encoding (RLE), the considered techniques are mainly *lossy*. In this regard, data compression (lossless) has also been used by some other authors to reduce the resources required to store time

series. For example, in [BWSR13] (one of the systems included in the aforementioned survey), a body-tracking data-store compressed binary format is presented as a storage layout. However, the proposal presented in this work has focused on time series dimensionality reduction techniques instead of general-purpose data compression techniques because, although both techniques can be used to reduce data storage and transmission costs, dimensionality reduction techniques also make it possible to optimize some data analysis techniques associated with time series processing [ASW15], [LKLC03]; such as time series similarity search, time series clustering, and time series data mining; where most of the algorithms scale poorly to high-dimensional data. Moreover, Approximate Query Processing (AQP) [JPT17] cannot be used for data compressed with general purpose data compression techniques.

Lastly, concerning data cleaning, different works have addressed time series cleaning tasks for example, to impute missing values [KLTCM20], detect outliers and anomalies [BGCML20], and removing noise [HYZ+17]. However, although there is a large number of works that consider time-series data cleaning techniques (a review of them can be seen in Chapter 3, Section 3.2.1.1), only few of them propose solutions that allow data engineers to automatize the selection of the most suitable techniques to apply to each type of time series. For example, in [BRG+12], a system that allows data engineers to visual-interactively compose time series pre-processing pipelines by themselves is presented, and in [JAF+06], a declarative query processing tool is presented, that uses spatio-temporal aggregations and integration of the data coming from several receivers to clean up the data. In this line, more recent approaches, like the one proposed in [DWS+19], are adopting for the automatic selection of the most suitable time series cleaning techniques through machine learning-based models. In this regard, in this work, before obtaining the reduced representations of the time series, they have been previously pre-processed (cleaned) by using the *Pre-processing Service* presented in the system architecture in Section 4.5.2.2. The goal of this service is to facilitate the time series cleaning task, and thus, for each time series type it provides some guidelines on which technique is appropriate and has more potential to work properly.

In summary, the contribution of the work presented in this chapter consists in the development of a system that incorporates: on the one hand, a machine learning-based model that recommends the most suitable reduction techniques to obtain time series reduced representations; and on the other hand, a service that recommends adequate techniques for cleaning time series. Neither of the works mentioned above supports both functionalities together, in such a way that efficiently guides the work of the data engineers in charge of performing the time series pre-processing task.

4.2 Pre-processing Techniques and Time-series Data

This section presents the pre-processing techniques considered in the proposed system for cleaning and obtaining a reduced representation of the time series, together with the time series used to build and test the system.

4.2.1 Cleaning Techniques

The *Pre-processing Service* of the proposed system, incorporates data cleaning techniques that are related with the processes of imputation of missing values, removing noise and detecting and handling outliers. Among the most frequent techniques for each process, presented in the technological background in Chapter 3, Section 3.2.1.1, a selection has been done, with the aim of covering a broad spectrum of type of series (e.g., periodic series, noisy series, discrete series, etc.) and user requirements (e.g., computational costs).

In particular, the *Pre-processing Service* offers the five missing values imputation techniques shown in Table 4.1. Moreover, in order to guide data engineers in the process of selecting the most adequate methods for missing values imputation, for each technique, the type of series for which the technique is appropriate and has more potential to work properly has been defined, as well as its computational cost. Those guidelines are also shown show in Table 4.1. An example of missing values imputation can be seen in Figure 3.6 in Chapter 3, Section 3.2.1.1, where the *Kalman Smoothing on structural time series models* method is used for the treatment of missing values in an industrial time series.

Table 4.1: Missing values treatment techniques and recommendations

Imputation Technique	Type of Series	Computational Cost
LOCF [Che14]	With short intervals of missing values, in categorical or discrete series	Low
Linear interpolation [MSBB+15]	Linear or with low curvature	Low
Spline interpolation [MSBB+15]	With intervals of missing values of curved shape Note: Sensitive to noise	Medium
Weighted Moving Average [ZSWY17]	With monotonically increasing or decreasing intervals of missing values Note: Not really useful for modeling peaks or valleys	Medium
Kalman Smoothing [Har90]	Multiple, especially series that show a constant trend or seasonality (e.g., periodic series)	High

With regard to the outliers, as already mentioned in Chapter 3, Section 3.2.1.1, their identification is difficult to carry out automatically; thus, either fixed values or a range of proper values are proposed for each of the parameters of the two techniques considered, taking also into account the type of time series. Table 4.2 shows a brief description of the considered methods for handling outliers and their parameters. An example of a method for detecting outliers is shown in Figure 3.9 in Chapter 3, Section

3.2.1.1, on which an *Adjusted Tukey's Range Test* is performed to detect outliers in a time series.

Lastly, with regard to noise removal techniques, two techniques are available in the *Pre-processing Service*. Table 4.3 shows these techniques together with some possible values of each parameter that are theoretically defined, so that data engineers will have the opportunity to adjust the value of each parameter in the corresponding range in the system (see the interface controls for adjusting these parameters in Figure 4.14). An example of the use of these techniques can be seen in Figure 3.7 in Chapter 3, Section 3.2.1.1, on which noise is removed from a time series, by using a *Frequency Filtering*.

TABLE 4.2: Outliers detection and removal techniques and recommendations

Cleaning Technique	Type of Series	Parameter	Recommendation
Sequential correction of contiguous observations[1]	Series with a low change ratio between samples	ϵ Maximum difference allowed between two consecutive observations	ϵ_{min} median of the changes of the series ϵ_{max} 95th percentile of the changes of the series
Adjusted Tukey's range test [HV08]	Series with a high change ratio between samples or with non-uniform behaviours along time	k Width of the interquartile range used to define the outliers w Width of the window used to process the signal	k Described in the original formula (see [HV08]) w An interval in which the time series behaviour is uniform

TABLE 4.3: Noise removal techniques and recommendations

Cleaning Technique	Parameter	Recommendation
Frequency filtering (low-pass filter) [MY12]	fc Cutoff frequency	fc_{min} median of the frequencies magnitudes' accumulated sum fc_{max} 95th percentile of the accumulated frequencies magnitudes' sum
Moving Average Filter [NLM11]	w Windows size d_{min} & d_{max} Minimum and maximum difference accepted between the processed and the original signal	w_{min} and w_{max}, window values that suppose an error roughly equal to d_{min} and d_{max}, respectively

4.2.2 Time Series Reduction Techniques

The different nature of the sensors capturing the time series in the application scenario leads to heterogeneous time-series data susceptible to be reduced by different techniques, with diverse reduction potential and different reconstruction error rates. Moreover, it could happen that a technique that works well for a type of time series, may not work well for another type of series (see Figure 4.1). Regarding the techniques applied to each

[1] A naive method in which all the v_i observations that do not satisfy $v_i \leq v_{i-1} + \epsilon$ are considered outliers, and they are replaced by the previous value.

type of series, taking into account the intrinsic characteristics of each type of series, two main groups can be distinguished in the considered scenario: continuous time-series data and discrete time-series data.

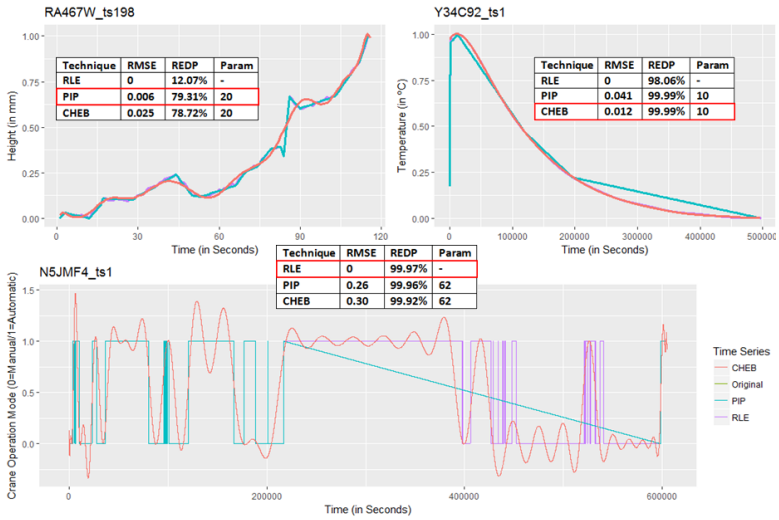


FIGURE 4.1: Heterogeneity in time series reduction techniques

With respect to *continuous time-series data*, the relevant technique families described in [Fu11] and the specific techniques used in [WMD+13] were leveraged for the initial selection of reduction techniques. Based on these references, the selected techniques are listed in Table 4.4, which also includes the meaning of a formal parameter that adjusts a bigger or smaller dimensionality for the reduced representation obtained by applying each technique (see Chapter 3, Section 3.2.1.2, for a detailed description of the considered techniques).

TABLE 4.4: Selected reduction techniques for continuous time-series data

Reduction Technique	Parameter of each technique
Sampling (SAM) [Ås69] Piecewise Aggregate Approximation (PAA) [KCPM01] Adaptive Piecewise Constant Approximation (APCA) [CKMP02] Perceptually Important Points (PIP) [FTLN02]	n = Number of selected points for the reduced representation
Piecewise Linear Regression (PLR) [Keo97b]	s = Number of segments to be approximated by linear regression
Polynomial Regression (PRE) [SZ96]	d = Degree of the polynomial
Chebyshev Polynomials (CHEB) [CN04]	c = Number of Chebyshev coefficients considered
Discrete Wavelet Transformation (DWT) [KA99] using the Haar filter [SS99]	l = Resolution level of the Haar transform

Concerning *discrete time-series data*, it should be noted that these time series represent different operating modes and status of the production equipment. For instance, they can represent whether a specific conveyor

belt is running forwards (and therefore moving a product unit into the starting point of a sub-process) or backwards (and therefore extracting the product unit once the sub-process is completed). Therefore, so as not to hamper the correct assignment of data segments to process steps, lossless reduction techniques must be used in these cases. Hence, the well-known RLE *(Run-Length Encoding)* [RC67] technique has been chosen for discrete time-series data, and the further analysis has been centered on continuous time-series data.

In relation to the reconstruction error, an important aspect to keep in mind about these reduction techniques is that although some lossless compression techniques are also considered, such as the RLE, the considered techniques are mainly lossy. Consequently, some information is lost when reconstructing the approximated time series from the reduced representations. It is difficult to quantify *a priori* the consequences of this information lost when approximating the time series, since those consequences do not depend only on the relevance of the data (if it is critical to keep the original data or it is enough to use an approximation), but also on the further data exploitation purposes. However, as it is stated in [CFM+04]: *"exact answers to queries are often not necessary, as approximate ones usually suffice to get useful reports on the world monitored by the sensors."*

Moreover, other authors have already examined the effect of performing data analytics over time series compressed with lossy reduction techniques. For example, in [AMCD20] and [ZD18], it is shown that when using these type of techniques, the classification performance of the used deep learning and Support Vector Machine (SVM) models (respectively) remains stable until a considerable compression ratio; and in [KS14], a validation of the usefulness of the proposed dimensionality reduction technique is carried out, where the experiments show that even for a significant reduction of the dimensionality, sufficient information about the time series is retained, for time series classification and clustering. In this work, in order to ensure data quality for further analysis purposes, a threshold has been established in order to ensure that the reconstruction error of a reduction technique is lower than 5% in terms of Root Mean Squared Error (RMSE).

Lastly, with regard to the reduction potential (REDP) of a reduction technique, this is expressed as the ratio between the disk space required to store the original time series and the disk space required to store the corresponding reduced time series representation (in %). In general,[2] the REDP of a technique is controlled by the parameters of the technique (see Table 4.4). Furthermore, those parameters also control the reconstruction error when approximating the time series. Indeed, lower parameter values lead to higher compression ratios (REDP) and higher reconstruction errors, while higher parameter values lead to lower compression ratios and lower reconstruction errors.

Therefore, when building the system proposed in this research work, with the aim of reaching a compromise between the reconstruction error

[2]Some techniques, like RLE, do not require any parameter and thus, their reduction potential does not depend on those parameters.

and the REDP, different reduction techniques with different parametrizations have been tested, and the parameter that obtained the greatest reduction with an error of lower than a 5% in terms of RMSE has been selected for each technique. The applied reduction techniques together with the obtained REDP (with the selected parameter), have been then reflected in a reduction potential vector that has served to build the models that recommend the most suitable techniques to apply to each type of time series (those models are presented in the following section). Table 4.5 shows an example of the corresponding reduction potential vectors for the time series *RA467W_ts771* and *RA467W_ts772* (those series belong to series of type *Heights* presented in Table 4.6 in Section 4.2.3).

TABLE 4.5: Time series Reduction Potential (REDP) of each reduction technique for two example series

Time series	SAM	PAA	APCA	PIP	PRE	PLR	DWT	CHEB
RA467W_ts771	29.13	40.95	18.47	17.73	21.90	32.62	120.98	41.62
RA467W_ts772	20.45	15.73	12.06	10.09	14.93	14.33	154.64	18.72

4.2.3 Raw Time-Series Data

In general, in this research work, the used data to build and test the different software artifacts that compose the solutions proposed as contributions, come from the real-world manufacturing context described in Chapter 3, Section 3.1. However, despite the massive amounts of data generated by the extruder machines supplied by the CEM from this scenario, there is not high variability in the nature of the data (most of the sensors produce data with similar characteristics that can be grouped into three main different groups according to those characteristics). Thus, in order to build and test the proposed system with more heterogeneous time series, the data from another CEM supplying another manufacturing sector with which the ITS Provider collaborates have been used.

This CEM aims to offer to their customers (other manufacturing companies from the polyurethane foam blocks production sector), smart services driven by the data captured by the sensors it has implanted in the manufacturing plants to which it supplies equipment. For example, Figure 4.2 shows a general perspective of a polyurethane foam block plant where the CEM has installed various sensors and data capture and collection devices. More specifically, Figure 4.3 shows an element of the plant, in particular a measurement arc, where six ultrasonic sensors are installed to take different measurements from the foam blocks (two sensors to measure the block height and four sensors to measure the block width; two for the top part and another two for the bottom part).

The plant under study has about 308 indicators obtained from various sensors connected to the data capture system for monitoring. These indicators register time-series data with continuous measurement at 1Hz (one measurement per second) of a variety of equipment parameters and physical magnitudes (temperatures, lengths, weights, capacities, etc.) related to

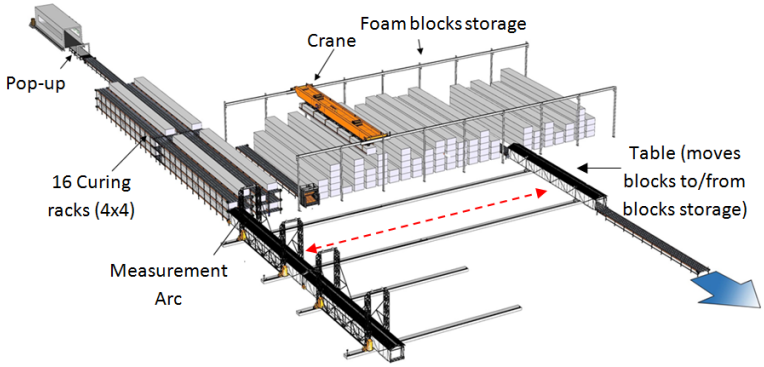


FIGURE 4.2: Polyurethane foam blocks production plant

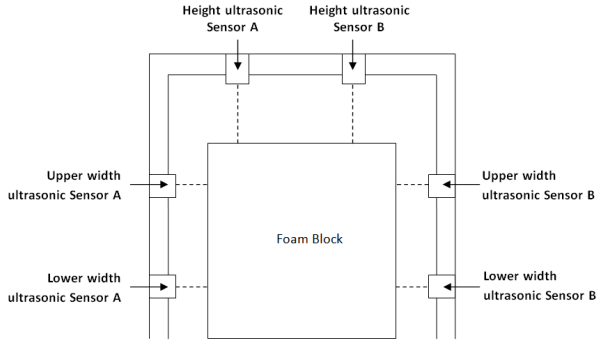


FIGURE 4.3: Measurement arc scheme with six ultrasonic sensors

the produced goods, raw materials, production processes, industrial equipment and environmental conditions. Table 4.6 shows some properties of the time series captured from those indicators during a complete week of plant operation. In particular, the type of time series, the type of sensor that captured the time series, the number of sensors of that type installed in the plant, the size in Mb occupied by that time series on disk, the average number of points of time series of that type, whether they represent product-driven data (i.e., where data for more than one single product are present) and whether they are continuous or discrete data is registered. This study ended up dealing with 1949 continuous time series. However, in order to build the proposed system with a wider range of time series types (from different domains), this study also considered 2139 time series that belong to the 10 datasets with the longest time series in the UCR Time Series Classification Archive [CKH+15] (see Table 4.7).

TABLE 4.6: Polyurethane foam plant time-series data properties

Time-series	Sensor type	N° of Sensors	Size in disk (Mb)	N° of points (Series mean)	Product driven	Data Type
Curing Temperatures	Thermal (°C)	32	575.5	246884	Yes	Continuous
Block Heights	Ultrasonic (mm)	9	163.8	67	Yes	Continuous
Block Widths	Ultrasonic (mm)	3	48.87	64	Yes	Continuous
Block Weights	Ultrasonic (mm)	8	147.01	216	Yes	Continuous
Tank Temperatures	Thermal (°C)	15	315.37	604826	No	Continuous
Tank Levels	Ultrasonic	11	231.63	604826	No	Continuous
Operation Mode & Equipment Status	Digital	170	2759.2	600289.5	No	Discrete Binary/N-ary
Conveyor Speed	Speed (rpm)	60	985.35	604811.8	No	Discrete N-ary

TABLE 4.7: UCR Archive time-series data properties

Series Type	N° of Series	Size on disk (Mb)	N° of points (Series mean)	Product driven	Data Type
CinC_ECG_torso	40	0.64	1380	Yes	Continuous
Coffe	28	0.112	286	Yes	Continuous
HandOutlines	370	9.62	2709	Yes	Continuous
Haptics	155	1.81	1092	Yes	Continuous
InlineSkate	100	1.95	1882	Yes	Continuous
MALLAT	55	0.66	1024	Yes	Continuous
Phoneme	214	2.5	1024	Yes	Continuous
StarLightCurves	1000	11.7	1024	Yes	Continuous
UWaveGesture-LibraryAll	896	10.5	945	Yes	Continuous
Worms	77	0.808	900	Yes	Continuous

4.3 Building the Machine Learning-Based Model

The main novelty of the proposed system resides in the development of a machine learning-based model that given a time series, recommends the most appropriate reduction techniques that allow obtaining an adequate reduced representation of the time series. This model has been built in two steps (see Figure 4.4). In the first step *(Discovering Time Series Families)*, an unsupervised classifier has been used to discover the time series groups or families to which a new time series could be assigned. In the second step *(Time Series Classification)*, a supervised classifier has been used to obtain a model that classifies new time series into those families. Details of both steps and the used models are provided in the following subsections.

4.3.1 Discovering Time-Series Families

An important aspect of building the model relies on having time series families for which the most appropriate reduction techniques have already been defined. However, it has not been found any well-defined set of families that fulfill that purpose. Therefore, this study has confronted the task of obtaining these families. The families have been obtained using

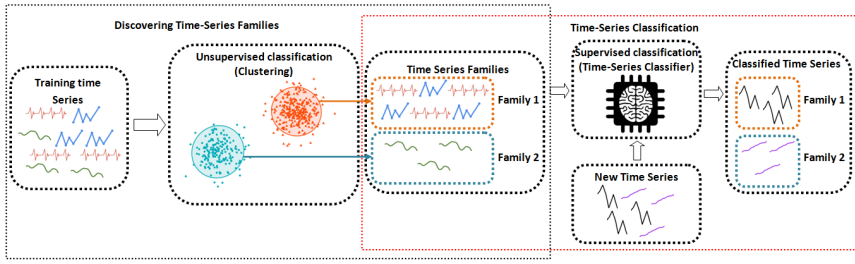


FIGURE 4.4: Machine learning-based model

unsupervised classification techniques. The most appropriate clustering configuration has been selected among several possibilities, considering different clustering types, similarity measures, number of clusters, and agglomeration methods.

Regarding time series clustering, although several time series clustering types can be found in the specialized literature, in [ASW15], most of these are classified into three main categories: *whole time series clustering*, *sub-sequence clustering*, and *time-point clustering*. However, with regard to those categories, some limitations have been identified. *Sub-sequence clustering* is performed on a single time series, not among time series [ASW15]. *Time-point clustering* is also applied to a single time series and its objective is finding clusters of time-points instead of clusters of time-series data [ASW15]. Therefore, considering that the goal of this research was to discover time series groups, the previous types of clustering are meaningless (this consideration appears also in [KL05a]), and for that reason, this study has been focused on *whole time series clustering*.

For whole time series clustering, various approaches exist, such as shape-based, feature-based, and model-based. However, model-based approaches have often been found to present scalability problems [Mit10], and their performance deteriorates when the clusters are close to each other [VGD], [ASW15]. Hence, this study has been focused on feature-based and shape-based approaches. The following subsections show how the time series families have been obtained for each approach.

4.3.1.1 Feature-based Clustering

Taking into account that the goal, in the present application context, is to discover families of time series for which one or more reduction techniques could find interesting application, as already mentioned in Section 4.2.2, for each raw time series a reduction potential vector has been constructed as a feature vector. Dealing with those vectors conventional clustering algorithms (such as k-means) could be applied with conventional distance/similarity measures (such as Euclidean distance) [ASW15]. Those type of vectors have been used successfully by other authors managing time series. For example, in [GJZ08], authors transform the time series into lower dimension feature vectors using the ICA algorithm and then they apply a modified version of the k-means algorithm; and in [RRJP13],

they combine features extracted from network intrusion detection system time-series data, to cluster the time series using the k-means algorithm.

When dealing with these reduction potential vectors, it has been noted that the considerable variation among the lengths of the time series (see the numbers of points in Table 4.6) is reflected in the very different reduction values obtained by the reduction techniques. Hence, to make comparisons among time series fairer, the values of each vector have been ordered according to the REDP obtained by each technique (leaving for last those that did not achieve a RMSE lower than 5%). Figure 4.5, illustrates the process of converting a raw time series into an REDP vector.

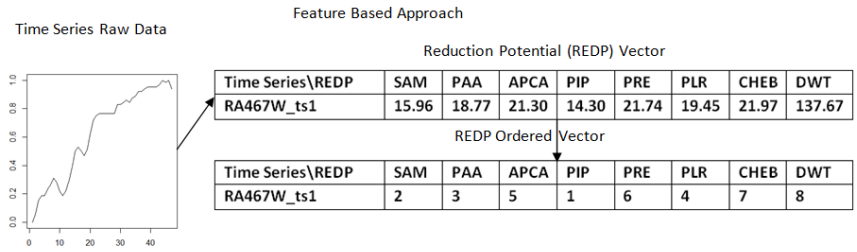


FIGURE 4.5: Feature based approach: process of converting a raw time series into a REDP vector

Afterwards, those REDP values that appear in the reduction potential vectors have been used to compound a distance matrix (using Euclidean distance as a distance measure). That matrix has served as input to the multiple clustering algorithms that have been used to build the various clustering configurations considered for grouping the time series into families. First, the hierarchical clustering algorithm implemented in the R package *stats* has been used with *ward.D, ward.D2, single, complete, average, McQuitty, median, and centroid* as agglomeration methods and, the Euclidean distance as a similarity measure, followed by k-means clustering with the *Hartigan-Wong, Lloyd, Forgy, and MacQueen* algorithms. These clustering algorithms have been used to obtain a number of clusters k, ranging from 5 to 15.

These algorithms have already been used in Smart Manufacturing scenarios: for example, in [TQLK18] a hierarchical clustering analysis is applied to discover energy consumption patterns for a fixed period of time in order to improve energy efficiency in a silicon wafer production line; in [KGHS14] they apply an approach based on the k-means clustering algorithm for detecting cyber-attacks that cause anomalies in Modern Networked Critical Infrastructures (NCI) in industrial control systems (because of its performance and simplicity); and the PWC company uses the k-means and hierarchical clustering algorithms for clustering in its Smart Manufacturing analytics platform [PWC].

4.3.1.2 Distance-Based Clustering

Time series distance-based clustering works directly with the raw time-series data by computing the distance between each pair of time series. Several approaches exist to compute the distance between time series (such as the *Euclidean distance* [WMD+13], the *edit distance on real sequences* [COO05], etc.). Most of these distances represent a one-to-one comparison of points (i.e., they compare the i-th point of a time series to the i-th point of another), which makes them inappropriate to compare time series of different lengths. However, other measures (known as *elastic measures*) allow one-to-many point comparison; for example, *Dynamic Time Warping* (DTW), introduced by Berndt and Clifford [BC94], allows comparison between time series of different lengths (see Figure 4.6).

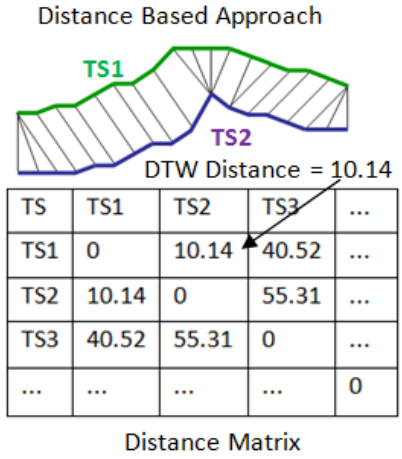


FIGURE 4.6: Distance based approach: process of computing the DTW distance matrix between time series

Due to the requirements of the application context (heterogeneous time series with different lengths coming from different sensors), this study has been focused on this last kind of distance measures and has chosen the well-known DTW algorithm as a similarity measure, which can find the optimal alignment between two time series. It has also been used successfully by other authors managing time series. For example, in [HNF08], authors use the DTW distance to measure the distance between raw time series for k-means and hierarchical clustering algorithms and in [NR07], authors use the DTW distance to measure the distance between raw time series extracted from multimedia data for k-means and k-medoids algorithms. However, DTW has a quadratic time and space complexity that limits its use only to small time series datasets.

Different approaches have attempted to overcome this limitation; for example in [SC07], Salvador and Chan proposed an approximated version of the DTW, called *FastDTW*, which has linear time and space complexity. However, despite the improvements introduced by Salvador and Chan to the DTW algorithm, the FastDTW algorithm was still observed to have

limited performance on very high-dimensionality time series. In [SMN+10], a parallel version of the algorithm has been proposed to speed up the computation of this distance measure. In this work, the application of the FastDTW has been parallelized to construct the distance matrix that would serve as input to the various clustering algorithms used.

For grouping the time series into families using the distance-based approach, the same clustering configurations as for the feature-based approach (see previous section) have been used. However, since k-means is designed to minimize variance, not arbitrary distances, the k-medoids clustering algorithm (available in the *kmed* R package) was chosen instead of k-means. This clustering configuration was used to obtain a number of clusters k ranging from 5 to 15.

4.3.1.3 Clustering Evaluation

To select the most appropriate set of families, both in the case of feature-based clustering and in the case of distance-based clustering, the performance of each clustering configuration has been evaluated using a five-fold cross-validation process. In each iteration of the cross-validation process, the 80% of the initial data (four folds) have been used to train the cluster algorithm and the remaining 20% (one fold) to validate it. First, each clustering process grouped the time series in the training dataset into k families. The ordered average of the arranged REDP vector of each time series assigned to a family was the reduction recommendation obtained for each time series family (Table 4.8 shows an example of the reduction recommendation obtained for two families). Then, the time series of the test dataset were assigned to those families using a KNN (k-nearest neighbor) classifier with k=1, based on the Euclidean distance between the test series REDP vector and the obtained reduction recommendation.

Table 4.8: Reduction recommendations for time series families

Family / Technique	SAM	PAA	APCA	PIP	PRE	PLR	DWT	CHEB
Family 1	3	6	7	1	2	4	8	5
Family 2	2	4	5	1	6	3	7	8

The Spearman's correlation coefficient has also been computed between the test series REDP vector and the reduction recommendation of the family to which the test series were assigned. This coefficient represents the strength of the monotonic relationship between the test series REDP vector and the reduction recommendation. The criterion used to measure the goodness of each clustering configuration was the average of the Spearman's correlation coefficient obtained for every time series in the test dataset for each iteration of the cross validation process. A threshold was fixed at 0.9 (which indicates a strong correlation between the proposed reduction techniques and the optimal reduction techniques [HWJ03]), to ensure that the recommendations obtained were appropriate. Moreover, to avoid compromising the sustainability and scalability of the proposed system, clustering configurations with fewer time series families were selected.

This ensured that if the need arises to redefine the time series families (because it is discovered that for a considerable number of new time series types, the recommendation obtained is sub-optimal) the minimum number of new time series families will be introduced to the system.

As a result, the clustering configuration that obtained a Spearman's correlation coefficient greater than the established threshold with the minimum number of families was the one that used k-means clustering with the *MacQueen* algorithm, that grouped the time series into eight families (see Figure 4.7).

FIGURE 4.7: Clustering configuration selection

4.3.2 Time Series Classification

Once the families have been selected, for building the classifier that would classify new time series into the defined groups, supervised classification techniques have been used with the two approaches mentioned before: feature-based and distance-based. However, for the feature-based approach, a feature vector is required; and, taking into account that the goal is to build a model that would predict and recommend the best reduction techniques for new time series, the reduction potential vector generated before (see Section 4.3.1.1) could not be used as a feature vector, because that would involve applying the combination of all the reduction techniques with their different parameterizations to the new time series, which would defeat the purpose. Therefore, another feature vector is needed in order to classify new time series properly according to the previously defined families. This section shows, first, the process of transforming a time series into a feature vector for the feature-based approach, and then, the selected model for building the time series classifier for each approach.

4.3.2.1 Feature Selection

Several studies have addressed the problem of transforming a time series into a feature vector for further data mining processes [MML16], [Ful17], [FJ14], [RK09], [NAM01]. However, although most feature extraction methods are generic in nature, the extracted features are usually application-dependent, meaning that a set of features that work well on one application might not be relevant to another [RK09], [Lia05]. Furthermore, as Timmer et al. stated: *"The crucial problem is not the classificator function (linear or nonlinear), but the selection of well-discriminating features. In addition, the features should contribute to an understanding..."* [TGDH93]. Therefore, the first task to accomplish is to choose a proper set of relevant features that allow classifying new time series properly according to the previously defined families.

Feature selection is a well-known challenge that has been addressed by several studies [GE03], [SIL07], [TAL14], [BCSMAB13], due to its multiple benefits, such as data dimensionality reduction, irrelevant data removal, and improvements in accuracy and in results comprehensibility [YL03]. In the considered scenario the well-known time series features extraction package *tsfresh* [CKLF16] has been used to extract features from the time series. Moreover, the feature selection process described in the subsequent paragraphs has been followed to select an adequate set of features for the considered scenario under study. This process considered two aspects: relevance of the features for the classification purpose, and simplicity (as suggested in [NAM01]).

Using the set of functions available in the Python package *tsfresh* [CKLF16], hundreds of time series features can be extracted. However, some of these functions require certain parameters for which a specific feature is generated as a result (e.g., the function that counts the number of points crossing a point m generates a feature for each value of m), which limited the number of different functions to 62. From the initial set of 62 functions, those functions that were not applicable to some time series, had high computational cost (to avoid making the data engineer wait a long time to obtain the syntactic characterization), needed specific tuning for each time series type (not generic enough), or produced complex outputs (e.g., functions that return a set of features such as different model coefficients) were discarded. This selection reduced the initial set of 62 functions to the 23 functions listed in Table 4.9. Details of what each function does are available on the *tsfresh* web site.[3]

From the 23 selected functions, the 49 features listed in Table 4.9 were extracted. The feature selection process started by focusing on those functions from which more than one feature was obtained (due to different parameterizations, as shown in Table 4.9). For those functions, the most relevant extracted features were selected based on the *Information Gain* (IG), which measures how much *information* a feature gives about the class or the family (i.e., its reduction of entropy [TAL14], [SIL07]). Table 4.10 shows an example of the information gain of the features extracted

[3] Tsfresh web site: `https://tsfresh.readthedocs.io/en/latest/index.html`

from different parameterizations of the *c3* function's coefficients and *autocorrelation* respectively. The selected feature is the one with the highest information gain coefficient. This made it possible to reduce the initial set of 49 features to 23 features (see, in Table 4.9, the colored features in the *Features* column).

TABLE 4.9: Summary of the selected features

Function	Parameters	Value	Features
abs-energy	-	-	abs-energy (F1)
mean-change	-	-	mean-change
number-crossing-m	m	mean	number-crossing-m (F2)
percentage-of-reoccurring-datapoints-to-all-datapoints	-	-	percentage-of-reoccurring-datapoints-to-all-datapoints (F3)
percentage-of-reoccurring-values-to-all-values	-	-	percentage-of-reoccurring-values-to-all-values
range-count	max, min	max=0.5, min = 0	range-count-max-0.5-min-0
		max=0.75, min=0.25	range-count-max-0.75-min-0.25
		max=1, min=0.5	range-count-max-1-min-0.5 (F4)
standard-deviation	-	-	standard-deviation
variance	-	-	variance (F5)
kurtosis	-	-	kurtosis
ratio-value-number-to-time-series-length	-	-	ratio-value-number-to-time-series-length (F6)
skewness	-	-	skewness
time-reversal-asymmetry-statistic	lag	1	time-reversal-asymmetry-statistic-lag-1
		5	time-reversal-asymmetry-statistic-lag-5
		10	time-reversal-asymmetry-statistic-lag-10
		15	time-reversal-asymmetry-statistic-lag-15
		20	time-reversal-asymmetry-statistic-lag-20
agg-autocorrelation	agg function	mean	agg-autocorrelation-f-agg-mean
		median	agg-autocorrelation-f-agg-median (F7)
		variance	agg-autocorrelation-f-agg-variance
autocorrelation-lag	lag	1	autocorrelation-lag-1
		5	autocorrelation-lag-5
		10	autocorrelation-lag-10
		15	autocorrelation-lag-15
		20	autocorrelation-lag-20
binned-entropy	max bins	5	binned-entropy-max-bins-5
		10	binned-entropy-max-bins-10
		50	binned-entropy-max-bins-50
		100	binned-entropy-max-bins-100 (F8)
c3-lag	lag	1	c3-lag-1
		5	c3-lag-5
		10	c3-lag-10
		15	c3-lag-15
		20	c3-lag-20 (F9)
augmented-dickey-fuller	attribute	p-value	augmented-dickey-fuller-p-value
		test-stat	augmented-dickey-fuller-test-stat
		used_lag	augmented-dickey-fuller-used_lag
index-mass-quantile	q	0.25	index-mass-quantile-q-0.25
		0.5	index-mass-quantile-q-0.5 (F10)
		0.75	index-mass-quantile-q-0.75
mean-abs-change	-	-	mean-abs-change (F11)
mean-second-derivate-central	-	-	mean-second-derivate-central (F12)
ratio-beyond-r-sigma	r	0.25	ratio-beyond-r-sigma-r-0.25
		0.5	ratio-beyond-r-sigma-r-0.5 (F13)
		0.75	ratio-beyond-r-sigma-r-0.75
		1	ratio-beyond-r-sigma-r-1
percentage-of-peaks	-	-	percentage-of-peaks
percentage-of-valleys	-	-	percentage-of-valleys (F14)

Next, a correlation test was performed on the previously selected 23 features to see whether irrelevant features were still present [Hal00]. To determine this, the correlation between each pair of features was computed, conforming the correlation matrix shown in Figure 4.8. A correlation filter was applied to that matrix to remove some redundant features. A threshold of 0.95 was established to remove those features whose correlation with other features was greater than the threshold. For each pair of correlated features, the feature with the greater IG was selected. This filter reduced the feature set from 23 to 20 (the removed features are the ones highlighted in red in Table 4.9). However there were still some highly correlated features (see Figure 4.8).

TABLE 4.10: Feature selection based on Information Gain

Feature	Parameter	1% series length	5% series length	10% series length	15% series length	20% series length
autocorrelation	lag	0.6390	0.4854	0.3543	0.4043	0.2674
c3	lag	0.3303	0.3383	0.3337	0.3545	0.3773

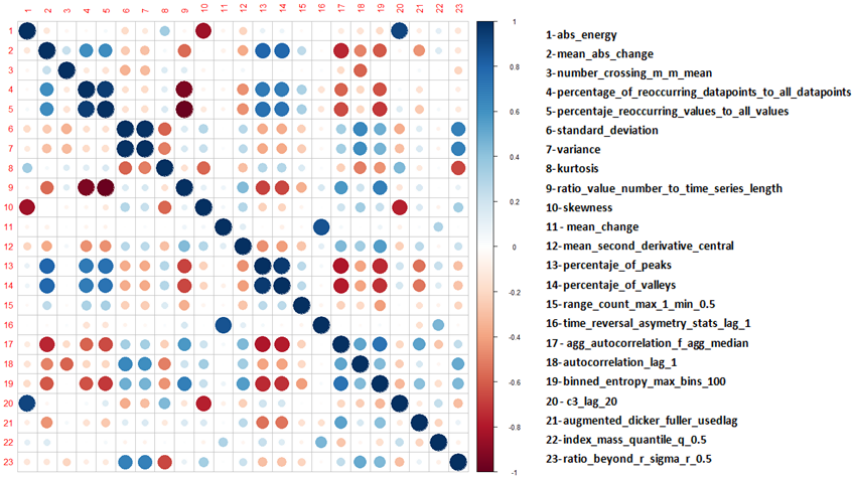


FIGURE 4.8: Correlation between extracted time series features

To select an even smaller number of features, more sophisticated feature selection techniques such as *filtering techniques*, *wrapper methods*, and *embedded techniques* [TAL14], [SIL07] were considered. Among these techniques, some of each type were used. The techniques used included the *RelieF* filter technique [TAL14]; two wrapper methods, the hill-climbing strategy and the greedy search strategy with forward selection [TAL14]; and the well-known random forest embedded technique.

Finally, all the features selected by the techniques used were examined, and those that were considered by more than one method were chosen. The resulting set of features is described in Table 4.9 (the ones highlighted in green, removing those highlighted in yellow) and contains 14 features.

TABLE 4.11: Classifiers performance evaluation

Feature-Based Approach			Distance-Based Approach		
Classifier	Parameter	Correlation	Classifier	Parameter	Correlation
RF	-	0.8451	KNN	k = 1	0.8417
SVM	-	0.8366	KNN	k = 5	0.8442
KNN	k = 1	0.8100	KNN	k = 10	0.8313
KNN	k = 5	0.8104			
KNN	k = 10	0.8084			
NB	-	0.7805			
GLM-LR	-	0.8073			

4.3.2.2 Supervised Classification Model Selection

Before building the supervised classifiers, the selected clustering configuration (see Section 4.3.1.3) has been applied to 80% of the available time series to classify them into the selected families. This process made it possible to label the time series for the supervised classifier construction. The remaining 20% of the time series have been used to test the classifiers. Table 4.11 shows the classifiers built for each approach with the parameters used and the obtained Spearman's correlation coefficient.

To measure the goodness of the classifiers, it was not possible to use classical indicators such as accuracy because the time series considered for testing did not have labels associated. Instead, the reduction recommendation assigned to each family has been compared to the original reduction recommendation for each time series, using the Spearman's correlation coefficient. The mean of this coefficient for all the time series composing the test time series collection has been the value used to measure the goodness of the classifier.

In the case of feature-based approaches, the built classifiers included a Random Forest (RF) classifier (using the R package *randomForest*), Support Vector Machine (SVM) and Naive Bayes (NB) classifiers (using the R package *e1071*), K-Nearest Neighbor (KNN) classifiers for $k = 1$, 5, 10 (using the R package *class*), and a version of the Generalized Linear Model (GLM) that applies the Logistic Regression (LR) for more than two categories (using the R package *nnet*). On the other hand, for the time series distance-based approach, this study used the DTW distance matrix computed using the FastDTW algorithm (see Figure 4.6) combined with a K-Nearest Neighbor classifier (for $k = 1$, 5, 10) that can yield high performance [Ful17], [BLB$^+$17].

The selected algorithms have been widely used in a vast range of industrial applications and in several use cases [NLMBC17], including time-series classification problems in Smart Manufacturing scenarios. For example, the PWC company uses the Random Forest, Logistic Regression and Support Vector Machine classifiers for classification in its Smart Manufacturing analytics platform [PWC]; in [MI17], they state the use of PCA with other methods including one-class Support Vector Machine and K-Nearest Neighbor as the predominant methods for anomaly detection as

typical advanced process control applications and analytic approaches in today's semiconductor manufacturing facilities; in [VJT+17], authors decided to use the most commonly used predictive methods and techniques (including among others the Random Forest, Support Vector Machine and the K-Nearest Neighbor classifiers), for the data gathered from a real manufacturing system of an automotive industry component supplier.

Table 4.11 shows that the highest Spearman's correlation coefficient was obtained by the Random Forest classifier on the feature-based approach and the K-Nearest Neighbor classifier (with $k = 1$ and $k = 5$) on the distance-based approach. However, the *Random Forest* classifier was selected in this study because its performance was better for very high-dimensionality time series. Finally, Table 4.12 presents the reduction ratio (in %) obtained for each time series family when assigning the time series to the families by using this classifier.

TABLE 4.12: Obtained reduction ratio (in %) for each time series family

Family	1	2	3	4	5	6	7	8
Reduction Ratio (in %)	86.27%	63.18%	99.06%	87.53%	98.4%	55.69%	95.56%	83.34%

Moreover, once the different time series families to which the time series would be assigned were discovered, and the classifier that would made those assignations was selected, the machine learning-based model has been built using the available time series (see Section 4.2.3). The average space saving achieved by the whole system for those time series is approximately the 90.24%.

4.4 Model Analysis

The selected Random Forest classifier has been analyzed from three perspectives: firstly, the distribution of the feature values across the families; secondly, the relevance of the features to determine to which family each time series belongs; and lastly, the interpretation of the model that underlies in the classifier.

4.4.1 Values Distribution of Features across Families

To ensure that the selected features enabled the differentiation between time series from different families, the significance of the difference between feature values across the time series families has been tested. The first step was to apply the Wilk-Shapiro [SW65] normality test (whose null hypothesis states that the population is normally distributed), to check whether the data under study follows a normal distribution. For each feature its distribution was checked across the time series families, and a p-value was obtained. Table 4.13 shows the calculated p-values.

Using a p-value of 0.05 as the rejection threshold the null hypothesis could not be rejected in only 2 of the 120 cases (the highlighted ones).

TABLE 4.13: Normality test of feature distributions across families

Family/ Feature	1	2	3	4	5	6	7	8
F1	1.51e-10	4.17e-07	1.86e-29	0.005858	1.31e-09	0.0032	5.19e15	1.47e-07
F2	1.14e-28	3.58e-17	2.03e-41	1.53e-22	3.48e-28	3.91e-26	7.82e-36	3.09e-31
F3	3.85e-18	0.001715	5.15e-55	6.6e-08	3.21e-25	2.35e-23	3.94e-34	0.01964
F4	3.74e-27	1.66e-12	9.97e58	1.83e-16	9.26e-33	1.64e-28	4.58e-42	1.26e-19
F5	8.71e-19	3.79e-08	6.67e-14	5.81e-05	3.1e-21	3.84e-12	2.812-05	1.97e-09
F6	2.59e-18	1.07e-09	9.06e-54	9.82e-09	1.082-23	8.38e-24	7.11e-29	4.05e-06
F7	1.11e-17	3.45e-06	1.15e-52	6.21e-13	8.66e-25	1.01e-17	5.63e-25	1.4e-13
F8	6.15e-13	0.032961	1.27e-49	2.3e-07	4.11e-10	1.15e-09	1.58e-21	0.001454
F9	2,18e-16	8.11e-13	3.99e-35	7.2e-13	7.43e-13	4.34e-11	6.36e-32	1.04e-15
F10	5.74e-11	0.069072	4.56e-29	1.12e-06	1.56e-19	9.1e-21	4.06e-08	0.001456
F11	1.63e-11	7e-06	1.2e-50	2.42e-08	4.692-27	7.27e-11	4.59e-34	0.3708
F12	2.10e-23	1.53e-06	1.37e-57	7.28e-12	5.84e-30	1.62e-25	3.18e-42	9.27e-13
F13	1.36e-24	0.00106	1.93e-28	3.13e-10	7.54e-23	0.000122	6.25e-18	1.91e-10
F14	2.1e-13	5.58e-05	6.64e-50	5.51e-09	6.85e-27	3.82e-20	1.8e-34	0.031614

TABLE 4.14: Kruskal-Wallis test p-values.

p-value	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
0.05	4.48e-170	0	9.45e-260	6.22e-311	0	3.73e-58	3.64e-14	0	0	4.36e-210	2.32e-168	9.64e-291	3.48e-291	1.96e-95

So, it could not be asserted that the data followed a normal distribution. Therefore, the Kruskal-Wallis non-parametric test was applied to check the similarity of feature values across families, resulting in the *p-values* shown in Table 4.14. All the p-values obtained were lower than the significance threshold (0.05), and therefore, the null hypothesis stating that the population distributions are similar was rejected at the rejection level.

Once the hypothesis that the population distributions of feature values were similar across families had been rejected, Dunn's pair-wise rank sum test was applied to each feature as a *post-hoc* test, to check which families significantly differed with respect to the rest. For each feature, a lower triangular matrix was obtained (see Table 4.15). By means of the p-values, the families that showed a similar population distribution (significance level of 0.05) for each feature could be identified (see Table 4.15). Table 4.16 shows, for each feature a summary of the pairs of time series families for which the null hypothesis of similarity was rejected. The sum of family pairs with similar population distributions (with respect to the total number of family pairs) serves as an indicator of the discriminatory power of each feature to differentiate between families (the lower, the better).

TABLE 4.15: Dunn Matrix: p-values obtained by Dunn's test for the feature *abs_energy*

Family	1	2	3	4	5	6	7
2	< 2e16						
3	< 2e16	5e-06					
4	4.2e-06	9.6e-06	0.47442				
5	1.6e-10	< 2e16	< 2e16	< 2e16			
6	< 2e16	0.47442	3e-07	7.6e-06	< 2e16		
7	< 2e16	0.0025	< 2e16	< 2e16	< 2e16	9.3e-07	
8	0.0003	4.5e-11	0.00052	0.30524	< 2e16	1.9e-13	< 2e16

TABLE 4.16: Dunn's test identified pairwise identical feature distributions

Family/ Feature	1	2	3	4	5	6	7	8	Sum
F1		6	4	3 8		2		4	6
F2	4			1					2
F3	8		8		7		5	1 3	6
F4	8				6 7	5 7	5 6	1	8
F5	4 6			1	7	1	5		6
F6	3 4 5	6	1 4 5	1 3	1 3	2	8	7	14
F7	3 4 6 7 8		1 4 6 7 8	1 3 6 7 8	6 8	1 3 4 5 7 8	1 3 4 6 8	1 3 4 5 6 7	34
F8	4	6 8		1		2		2	6
F9		8		5 6	4 6	4 5		2	8
F10		5			2		8	7	4
F11		3	2	6		4			4
F12	8				6 7	5	5	1	6
F13		8	7	5 6	4	4	3	2	8
F14	5 6 8	6 8		5 7	1 4 6 8	1 2 5 8	4	1 2 5 6	20

4.4.2 Feature Relevance across Families

After checking the existence of statistically significant differences in feature values across time series families, the next objective was to detect those features that had greater discriminatory power for identifying time series families. The one-against-all strategy (as proposed in [Trz10]) was used to evaluate the influence of the input variables on the decision function obtained by the Random Forest classifier as a two-class classifier. The importance of each feature for each class was measured, using the *Mean Decrease in Accuracy (MDA)* [Nic11] function of the *randomForest* R package [LW02].

Figure 4.9 shows the calculated degree of relevance of each feature across time series families. Although in most families a large difference was not observed, for others (e.g., family 7), a subset of the features showed a notably greater relevance and higher importance than the others (e.g., *mean_abs_change* and *agg_autocorrelation_f_agg_median*).

FIGURE 4.9: Feature relevance across different time series families

4.4.3 Rules Learned by the Random Forest Classifier

The Random Forest classifier is a tree-based learning method that shows a strong ability to generalize on real datasets. Nevertheless, despite its competitive performance in different domains, its major drawback resides in its "black box" nature and its lack of comprehensibility for domain experts as a wide forest made up of trees and rules is learned [MG15]. In the present case, the Random Forest model is composed of 500 single trees. However, some researchers have tried to make this classifier easier to interpret [SS11], [Den14]. This study has used the *inTrees* (interpretable trees) framework [Den14] that extracts, measures, prunes, and selects rules from a tree ensemble and then calculates the most frequent variable interactions to output a set of rules from the Random Forest model.

The first step was to extract the rules from the learned Random Forest classifier. A total of 59397 rules were extracted (with a maximum depth of eight levels for each tree). As the extracted rules may include irrelevant variable-value pairs, these were pruned using the *leave-one-out pruning* method proposed in [Den14]. Next an example of the first rule extracted from the first three is shown before and after pruning.

```
Before Pruning:
autocorr-f-agg-median <= 0.9
& autocorr-f-agg-median <= 0.36
& autocorr-f-agg-median <= 0.34
& %-of-reoccurring-dp <= 0.019
& percentage-of-valleys <= 0.075
& ratio-beyond_r-0.5 <= 0.71

After pruning:
autocorr-f-agg-median <= 0.36 & %-of-reoccurring-dp <= 0.019
```

Since the number of rules extracted from a tree ensemble can be large and (partially) redundant, a compact rule set containing relevant and non-redundant rules was derived, using the method proposed in [Den14] and [DRTB14]. Using this method the initial set of 59397 rules was reduced to 13 rules.

Finally, the rules extracted from a Random Forest can be summarized into a rule-based learner, also known as a Simplified Tree Ensemble Learner (STEL) [Den14] (see Table 4.17). The STEL has been built using the selected rules and has been used for class prediction over the same instances previously used for the Random Forest classifier (see Section 4.3.2.2). Then, the classes predicted by the Random Forest algorithm have been compared with those predicted by the STEL classifier obtaining a correspondence of 80%.

4.5 Time Series Pre-processing System

The proposed pre-processing system is available as a visual-interactive web system with two main interfaces, the Interface for Time Series Reduction System (I4TSRS) [VDI+18] and the Interface for Time Series Pre-processing System (I4TSPS) [VVD+18], that support the different tasks (reduction and cleaning respectively) related to the time series pre-processing. These interfaces are supported by two different web services

TABLE 4.17: Rules extracted by the STEL model

Rule	Condition	Family
R-1	autocorr-f-agg-median>0.7 & mean-2-deriv-centr<=-0.00012	5
R-2	mean-abs-change>0.0054 & num-crossing-mean>11 & rat-val-num-ts-length>0.985	6
R-3	num-crossing-mean>5 & percentaje-of-valleys<=0.0107	3
R-4	mean-2-deriv-centr>-0.000051 & mean-abs-change<=0.0207 & percentaje-of-valleys>0.011	7
R-5	mean-abs-change>0.085	2
R-6	mean-abs-change>0.0674	2
R-7	binned-entropy-100>2.8 & mean-2-deriv-centr>-0.0036 & num-crossing-mean<=7.5	4
R-8	index-mass-q-0.5<=0.33	4
R-9	binned-entropy-100<=3.54 & mean-2-deriv-centr>-0.005 & mean-abs-change<=0.072 & num-crossing-mean<=60 & percentaje-of-valleys>0.047	8
R-10	autocorr-f-agg-median<=0.9084527139165 & mean-2-deriv-centr>-0.0036 & mean-abs-change<=0.072 & %-of-reoccurring-dp>0.027	8
R-11	mean-abs-change<=0.015 & %-of-reoccurring-dp<=0.00049	3
R-12	%-of-reoccurring-dp<=0.0089 & rat-val-num-ts-length<=0.99	7
R-13	mean-abs-change<=0.086 & mean-abs-change>0.061 & num-crossing-mean<=9.5	1

respectively: the *Reduction Service* and the *Pre-processing Service*. Moreover, these services are part of a global framework that is being developed to provide multiple interconnected services in Smart Manufacturing scenarios (see [VBD+18]).

The following subsections present, first, the architecture of the proposed time series pre-processing system together with the two different interfaces for performing the different tasks supported by the system; next, the services supporting them; and lastly, an example of use of each of them to clean and obtaining an adequate reduced representation of industrial time-series data (respectively).

4.5.1 System Architecture

The system is based on Firebase,[4] a cloud platform that data engineers can access from multi-platform devices. It has been developed using Firebase, a Figure 4.10 shows the system architecture, in which four main software components can be distinguished: *Web App, Back-end Service, Intermediate Service* and *Web Services*. A brief description of them and their modules is presented in the following subsections.

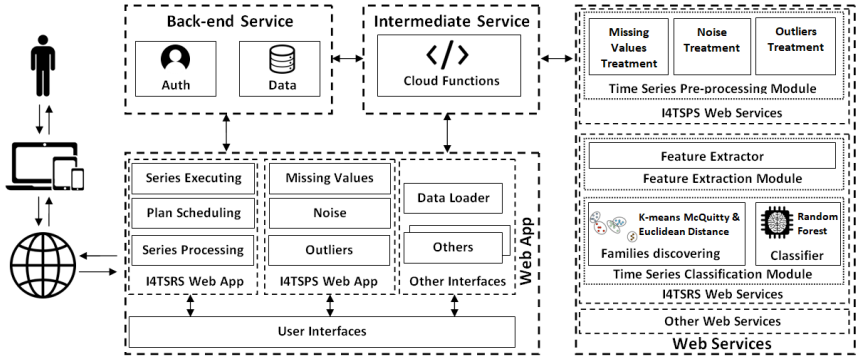


FIGURE 4.10: Time Series Pre-processing System architecture

[4]Firebase web site `https://firebase.google.com/`

4.5.1.1 Web App

This component allocates the Web application front end. It includes different interfaces used to interact with the various modules composing the web application. These interfaces can be grouped into two main components according to the different data pre-processing tasks that can be performed: the Interface for Time Series Reduction System (I4TSRS), and the Interface for Time Series Pre-processing System (I4TSPS). Each of them is presented below. Moreover, there is another component to take into account, and its the *Data Loader*. This component provides the entry point for data input (i.e., the collection of all the time-series data from different indicators) to the data storage in the *Back-End Service*.

I4TSRS Web App

The I4TSRS Web App allocates the different interfaces that allows to obtain an adequate reduced representation of time-series data. There are three main modules that compound the I4TSRS Web App:

- *Series Processing*. This module invokes two modules of the *Reduction service*. In particular, the *Feature Extraction* module and the *Time Series Classification* module, that are used to identify the most suitable reduction techniques for the captured time series.
- *Plan Scheduling*. This module is in charge of grouping the series according to the family to which they have been assigned. Each family is represented by a group. The groups are ordered taking into account the reduction potential (REDP) of all the series that belong to that family. Inside each group, the most adequate reduction techniques for the family are represented and ordered according to their reduction potential (REDP) in terms of data storage space saving.
- *Series Executing*. This module is in charge of applying a selected reduction technique over a time series; of visualizing the results as a chart that reflects a comparison between the original and reduced time series; and of presenting a summary table showing the reduction in storage space achieved and the RMSE obtained between the original time series and the reconstructed approximation.

I4TSPS Web App

The I4TSPS Web App allocates the different interfaces that allow to perform different tasks related with the time-series data cleaning process. These interfaces allow executing the different time series cleaning techniques available in the *Pre-processing Service* (see Section 4.5.2.2). Moreover, for each cleaning task, a description of the different techniques available is provided, along with some recommendations to select the appropriate techniques and parameter values required by them. There are three main modules that compound the I4TSPS Web App, each of them associated with a particular cleaning task.

- *Missing Values Imputation Module:* This module is in charge of imputing missing values in a time series by using the techniques available in the *Pre-processing Service*.
- *Outliers Detection Module:* This module leverages different techniques available in the *Pre-processing Service* to detect and handle outliers in a time series.
- *Outliers Detection Module:* This module accomplish the task of executing the different techniques available in the in the *Pre-processing Service* for removing the noise present in a time series.

4.5.1.2 Back-End Service

The *Back-End Service* component provides the Web application with the required user security tools and data storage resources. *The Back-End Service* is made up of modules that can be accessed and managed through the Back-End API:

- *Data Storage.* This module implements data persistence. It stores the analyzed time series, their characterizations, their assigned families, the recommended techniques, the reduced representations, the cleaned time series, etc.
- *Auth.* This module contains different user and role descriptions that are used to manage various aspects, such as user authentication and security levels, to verify that users access only resources for which they have appropriate permissions.

4.5.1.3 Intermediate Service

The goal of the *Intermediate Service* component is to avoid inefficiencies in the *Web App* component related to the waiting times required to process high-dimensionality time series in the Reduction and Pre-processing services. For this purpose, this *Intermediate Service* enables asynchronous communications between both type of components.

4.5.2 Web Services

As mentioned before, the different functionalities available in the pre-processing system are supported by different services that are part of a global framework that is being developed to provide multiple interconnected services in Smart Manufacturing scenarios. Two main services have been developed in this research work to support the pre-processing system presented as the first main contribution: the *Reduction Service* and the *Pre-processing Service*. An overview of them is presented in the following subsections.

4.5.2.1 I4TSRS Web Service

The purpose of the regarding *I4TSRS Web Service* component is to recommend the most suitable reduction techniques for the captured time series. This service leverages the built models (presented in Section 4.3) to obtain a group of time series families and then classifies new time series into the identified families, based on features extracted from the series. This service is composed of two main modules: the *Feature Extraction* module and *Time Series Classification* module.

- *Feature Extractor*. This module extracts the features that conform the syntactic characterization of the time series (e.g., recurrent values, variance, number of valleys, etc.). Section 4.3.2.1 presents more details about these features.
- *Time Series Classification*. This module uses the features extracted by the *Feature Extractor* to select the most suitable reduction techniques for the captured time series. This association is made by a novel machine learning-based model constructed as part of this study, which has already been described in Section 4.3.

4.5.2.2 I4TSPS Web Service

The *Pre-processing Service* of the system proposed in this chapter, incorporates data cleaning techniques that are related with the processes of imputation of missing values, removing noise and detecting and handling outliers. Among the most frequent techniques for each process, presented in the technological background in Chapter 3, Section 3.2.1.1, a selection was done, with the aim of covering a broad spectrum of type of series (e.g., periodic series, noisy series, discrete series, etc.) and user requirements (e.g. computational costs). In particular, the *Pre-processing Service* offers the different techniques presented in Section 4.2.1.

4.5.3 Demonstration of the Time Series Pre-processing System

As mentioned before, the time series pre-processing system proposed in this research work allows to accomplish two task with respect to the time series pre-processing. This section shows the way data engineers can use the two main functionalities provided by this system. However, in order to ensure privacy and confidentiality, data engineers only have access to the resources they have been authorized to see. Therefore, first of all, they have to authenticate themselves, and then, they can use the two functionalities that are presented in the following subsections.

4.5.3.1 Time Series Reduction with the I4TSRS Web App

The I4TSRS Web App (see Section 4.5.1.1) allows to obtain an adequate reduction representation of a time series.[5] For that, first, users

[5] A demo video is available at `https://fdai-b5221.firebaseapp.com/#/demo`

will have to upload the time series (e.g., series numbered 5QYA5X_ts5, and RA467W_ts771 from *examples series*, a set of real-world time series that can be downloaded from the Web App). Those series belong to series of type *Curing Temperatures* and *Heights* presented in Table 4.6. In Figure 4.11, it can be observed that once time series are uploaded, their features are also extracted (see the highlighted panel in blue in Figure 4.11) and the assignment of the time series to their families is done using the built machine learning model. Also some characteristics of the assigned family are included, such as an adequate reduction technique with the associated optimal parameter (see the highlighted panel in red in Figure 4.11) and the expected reduction ratio.

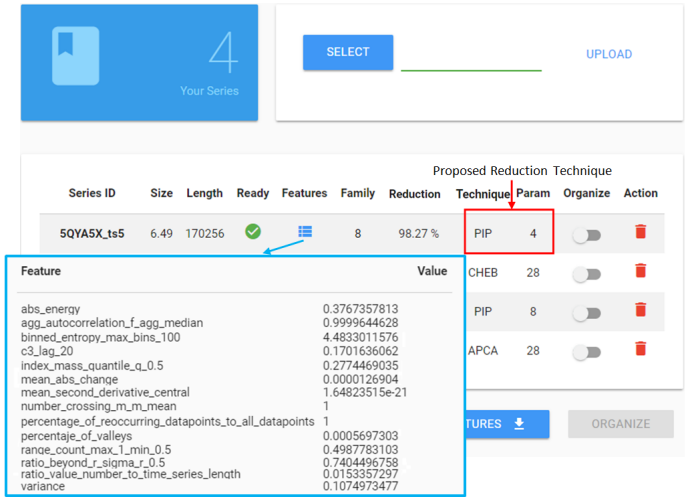


FIGURE 4.11: Time series feature extraction and family assignment in I4TSRS Web App

Once the data engineer has uploaded the time series, a set of time series could be selected to conduct the reduction analysis. The reduction analysis will group the time series according to the families in which the model classifies them and will show, for each group, various reduction plans which are represented in blue cards (see Figure 4.12 top left panel). For each plan there appear the time series family to which the series have been assigned, the recommended technique (and its parameters values) for the series within that family, and the expected RMSE and REDP criteria respectively. Those plans are ordered according to the expected reduction potential.

Once the data engineer has selected a concrete plan, the plan could be loaded and executed. When executing the plan, a chart that reflects the relation between the reduction potential and the error is generated (see Figure 4.12, top chart in the bottom panel). While that chart is generated, a table showing the reduction analysis results of each time series is being updated. In that table the data engineer could select a concrete value and explore the reduction technique to see a chart that reflects a

comparison among the original series and the reduced (see Figure 4.12, bottom chart in the bottom panel). Moreover, the data engineer could play with different parameters and techniques for fine tuning the selection of the most adequate reduced representation (see Figure 4.12, top-right panel). Notice that when executing the proposed plan by the system, the examples series could be reduced from 6.49 and 0.00135 Mb to 0.0001 and 0.000136 Mb respectively, saving 99.85% and 88.99% of the storage space used for those series.

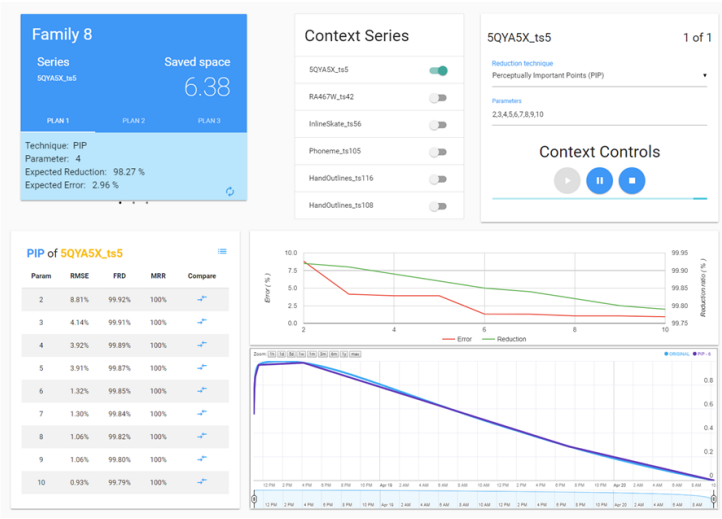


FIGURE 4.12: Time series reduction plan executing in I4TSRS Web App

4.5.3.2 Cleaning Time Series with the I4TSPS Web App

The I4TSPS Web App (see Section 4.5.1.1) allows executing the different time series cleaning techniques associated with the processes of imputing missing values, detecting and handling outliers and removing noise. For that, the data engineer will select the I4TSPS Web App in the main menu. Once in the I4TSPS Web App, the data engineer will see the *Data Cleaning Techniques Catalogue* (see Figure 4.13) on which the different available techniques are presented in different tabs (one for each time series cleaning task).

Once the data engineer has selected the cleaning task to perform and the concrete technique to apply, a new interface is shown where the system shows the original time series and the cleaned version of the time series. Moreover, it allows to change the technique to apply and to play with different parameter values of the technique. As an example, Figure 4.14 shows how the I4TSPS Web App allows to impute the missing values in a time series by using the *Kalman Smoothing on structural time series models* method.

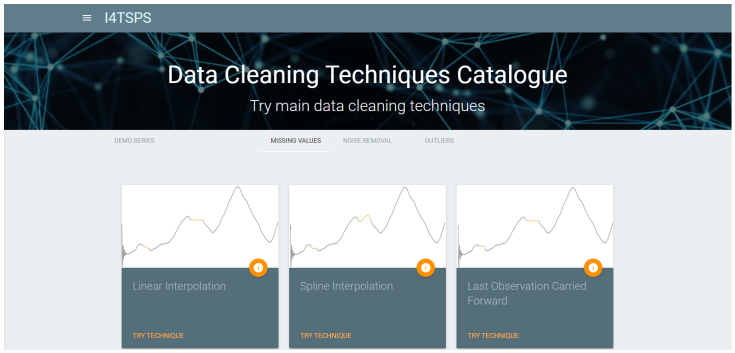


FIGURE 4.13: Data cleaning technique catalogue in I4TSPS Web App

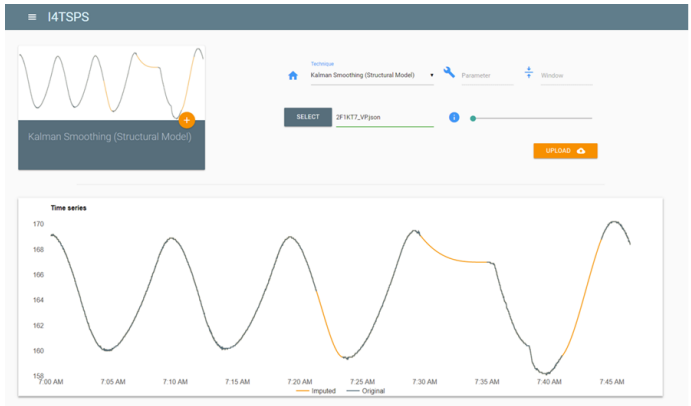


FIGURE 4.14: Missing values imputation in I4TSPS Web App

4.5.3.3 Reduction of Time Series that have already been Pre-processed

Previous sections have shown how the proposed system allows to obtain a reduced representation of the time series and cleaning them. However, there is an important aspect to consider when obtaining a reduced representation of the time series, and is that the time series reduction techniques considered by the system show a better performance when dealing with time series that have already been pre-processed (cleaned). For example, in Figure 4.15, another example series (series numbered 4H6EML_ts1) is used to illustrate the relevance of cleaning up the time series before obtaining their reduced representation. In this example the time series goes from occupying 5.69 Mb to 341.4 Kb saving the 99.4% of the storage space used for that series, with an RMSE of 20.22% for the raw series and an RMSE of 1.24% for the cleaned series.

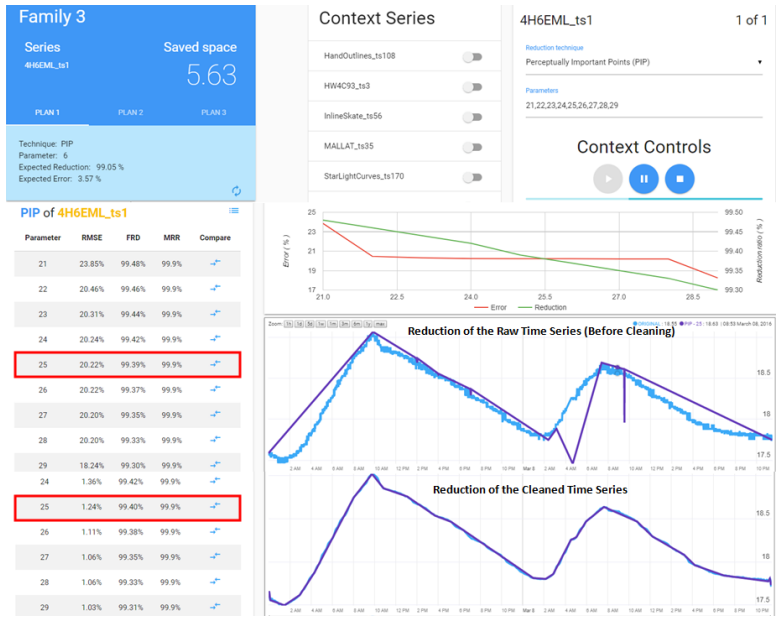


FIGURE 4.15: Comparative of the reduction of a time series before and after cleaning

4.6 Conclusions

This chapter presents *a system that efficiently guides a data engineer in the task of pre-processing raw time-series data* coming from industrial sensors. The main contributions of this system can be noted from two different perspectives: scientific and real applicability.

From the scientific point of view, the main contributions are related to the built machine learning-based model that recommend the most suitable time series reduction techniques to apply for each type of time series. These contributions are the following: the clustering of heterogeneous industrial time series into families (in particular, eight families have been identified); a classification mechanism that classifies new time series into the identified families, based on features extracted from the series; and a set of features that adequately collect the singularities of different types of industrial series (in particular, 14 distinct features have been selected).

From the real applicability point of view, the contribution is twofold: on the one hand, the proposed system provides a wide range of time series pre-processing techniques for the different tasks related to time series cleaning and dimensionality reduction; and on the other hand, it provides some recommendations on which techniques are more suitable and have more potential to work for each type of time series in Smart Manufacturing scenarios, where heterogeneous sensors capture time series of different nature susceptible to be pre-processed by different techniques. These recommendations could serve as effective guidelines for the person in charge of time series pre-processing in order to facilitate such a complex task.

With respect to the time series cleaning techniques, among the variety of existing techniques in the literature, the system provides a set of representative techniques that cover a broad spectrum of type of series and users requirements. In particular, it provides five techniques for estimating missing values, two techniques for removing noise, and two techniques for detecting and handling outliers. Moreover some general information about: the characteristics of a time series for which the application of a concrete technique is most appropriate, the computational cost of its application, and about an appropriate range of parameters values that different techniques require, is also provided by the system. With regard to the reduction techniques, the system provides nine different techniques from the most representative families analyzed and discussed in the literature. The selection of the most suitable techniques is carried out by the aforementioned machine learning-based model.

Moreover, the solution presented in this work contributes to considerable savings in storage and transmission costs, for time-series data managed by manufacturing companies that are developing Big Data driven services for their customers, by obtaining an adequate reduced syntactic representation of these time series. Indeed, in the following chapter a novel architecture for Time Series Management Systems is proposed that leverages the reduction techniques proposed by this system to considerably reduce the required data storage resources (and its associated costs) in the cloud. Moreover, the obtained reduced representations do not limit the future use of the data for further analysis purposes.

Finally, it should be mentioned that part of this chapter has already been published in two conference papers: the first one, entitled *"I4TSRS: A System to Assist a Data Engineer in Time-Series Dimensionality Reduction in Industry 4.0 Scenarios"* [VDI+18] in the *27th ACM International Conference on Information and Knowledge Management (CIKM 2018),* where the *Reduction Service* was presented; and the second one, entitled *"I4TSPS: a Visual-Interactive Web System for Industrial Time-Series Preprocessing"* [VVD+18] in the *2018 IEEE International Conference on Big Data*, where the *Pre-processing Service* was presented.

Chapter 5

A Three-Level Hierarchical Architecture for an Efficient Storage of Time-Series Data

The deployment of Industrial Internet of Things (IIoT) devices in Industry 4.0 scenarios allows to capture big amounts of data, regarding different magnitudes or indicators of interest that are usually stored for further analysis processes (product quality or process efficiency control, fault diagnosis, predictive maintenance of equipment and other purposes). Most of those data are time series generated by large-scale sensor networks monitoring the continuous operation of the manufacturing processes or equipment to be analyzed. The captured time-series data flows continuously into an endlessly accumulating data stream that cannot be stored within a bounded storage space, such as traditional Database Management Systems (DBMSs). Thus, although these systems have been successfully deployed in many applications, they are unsuitable to handle the velocity and volume of the time-series data generated by large-scale sensor networks [SSRG13], [DF14], [JPT17], and therefore, specialized Time Series Management Systems (TSMSs) are being developed to overcome the limitations of general purpose DBMSs for time series management.

Regarding the development of new proposals for TSMSs, in the survey presented in [JPT17], an analysis and classification of TSMSs developed through academic or industrial research and documented through publications is presented. In that survey authors provide a summary of the research directions proposed by other researchers in the field and also their vision for the next generation of TSMSs. In this regard, they propose a distributed TSMS with a physical layer storing approximated (and reduced) representations of the time series using mathematical models.

Moreover, in [CFR13], it is shown how the use of cloud infrastructures in combination with sensor networks enables the development of new types of TSMSs, and the new open problems that appear related to it. One of those problems is related to the storage of massive amounts of data in distributed Cloud Computing infrastructures, where the cloud storage

resources and their associated costs, limit the scalability of the TSMSs. In fact, the analysis of cloud costs presented in [GHMP09], shows that data storage consumes 45% of the total cost, while infrastructure consumes 25%, and the network and power draw consume 15% each. In order to alleviate those data storage costs, frequently, the oldest data are removed to free resources for new data, losing valuable historical data for further analysis processes [Ama20]. However, as mentioned in [Kus17]: *"although high volumes of rapid measurements in smart manufacturing scenarios cost more to store, long-term data are essential in these scenarios."*

In this regard, this chapter presents the second contribution of this research work; which consists in the design and development of *a three-level hierarchical architecture for Industry 4.0 time-series data storage* on cloud environments. The proposed architecture allows to reduce the required data storage space and consequently, its associated costs. It follows a multi-temperature data management paradigm [SCS+12] on which the temperature tiers hot, warm and cold are considered, and thereby, it is materialized as a three-level hierarchical architecture. In the first level of the architecture, the most recent raw time-series data can be stored for a short-period of time on Solid-State Drives (SSDs), providing fast access for real-time applications; in the second level, recent raw time-series data can be stored on magnetic Hard Disk Drives (HDDs), for a medium period of time; and in the third level, a reduced representation of the time series obtained by applying time series reduction techniques can also be stored in HDDs for a longer period of time.

The main novelty of the proposed architecture relies on the nature of the third level, where a reduced representation of the time series is obtained, by using different types of time series dimensionality reduction techniques (presented in Chapter 3, Section 3.2.1.2), that allow to decrease the required data storage resources without almost hampering the use of those data for further analysis purposes. The proposed architecture, has been implemented by using some of the top DBMSs from four different categories: Time series DBMS, Wide-column stores, Document stores and Graph DBMS (these categories together with the considered DBMS have already been presented in Chapter 3, Section 3.2.2). Thus, the behaviour of those storage systems in a real industrial scenario is also presented as a contribution. It has been tested by using industrial time series coming from a real manufacturing environment, and with four different types of queries proposed by domain experts. The performance results regarding storage space, storage costs and total query time for each DBMS are shown and contrasted in this chapter.

The chapter begins with an analysis of related work regarding the development of TSMSs; then, an overview of the proposed architecture is presented along with details of the real manufacturing context from which the data used to test the architecture come; afterwards, some results of the performed tests using the four different types of DBMSs are presented under two different main dimensions: used storage space (and an estimation of the associated costs), and query answering time; then, how the use of Design of Experiments (DoE) techniques can help to select an adequate

implementation of the proposed architecture for the real-world scenario of this research work is analyzed; and finally, some conclusions are presented.

5.1 Related Work

In Smart Manufacturing scenarios, the high volume of time-series data generated during the manufacturing processes need to be captured, processed and stored in order to perform advanced analytics, that allow to extract knowledge from them [CHT09]. In these scenarios, the volume of the captured data is really huge [Ris18] [Zho19]. For example, in [Ris18], it is stated that in 2010 manufacturing companies were already generating 1.800 petabytes of data per year, and that time series databases are experiencing an explosive growth in recent years. In the following paragraphs, three relevant aspects when managing those time-series data are considered: data storage, data access and query capabilities.

With respect to data storage, in [Zho19], it is stated that those industrial time-series data, usually, require high concurrency, throughput and writes; but low reads; and thus, the use of NoSQL databases in the data storage layer, such as Apache HBase or Cassandra; or cloud services, such as Alibaba Cloud's Table Store, is very suitable. Notice that this is coherent with Shafer et al. [SSRG13] that had already pointed, that general purpose frameworks and databases were not appropriate for numeric time series, such as those generated during manufacturing processes.

In the survey presented in [JPT17], different specialized TSMSs developed to overcome the limitations presented by general purpose DBMSs are shown. The implementation of those TSMSs is based on different technologies, such as Relational DBMSs, NoSQL DBMSs (e.g., Apache HBase, MongoDB, CouchDB, etc.); proprietary solutions; or services in the cloud, such as Amazon S3 or Microsoft Azure. Other TSMSs which are based on NoSQL DBMSs are ranked in [Sol19] (e.g., OpenTSDB, Time-scaleDB, KairosDB, etc.), and in [MFP+19], a recent study shows the suitability of three of these NoSQL DBMS (*Cassandra, MongoDB and InfluxDB*) for the storage of IIoT time-series data. In the architecture proposed in this research work, those DBMSs mentioned in [MFP+19] have been used; InfluxDB, a time series specific DBMS, and MongoDB and Cassandra, general purpose NoSQL DBMSs. Moreover, in order to test the proposed architecture with a system developed with a different specific purpose, it has also been implemented with Neo4J, a native graph database platform that has already been used for time-series Big Data management [PMS18].

Furthermore, with regard to cloud services used for storage purposes, some research is being performed to develop techniques that allow to optimize data storage on those environments. For example, [LSN19] addresses the optimization through the reduction of the replicated data blocks in the nodes, an adequate distribution of the replicas in different storage mediums based on data popularity, and the compression of the least accessed data block replicas. In this work, the proposed architecture is deployed on the cloud, and the storage resources optimization is addressed through the use of time series dimensionality reduction techniques.

With respect to the data access, in [Zho19], they assert that there is distinctive hot and cold data access, and that recently written data is accessed more frequently. Therefore, choosing a storage medium with higher Input/Output operations per second for hot data improves the overall query efficiency. For example, in [IMMR16], a method for managing a database in real-time is provided where data are stored on different physical locations (SSD, Fibre Channel System Attached Storage and Serial Advanced Technology Attachment (SATA)), based on their storage priority. In fact, there exists a tendency to follow a multi-temperature database management paradigm, such as the one proposed by IBM DB2 [SCS+12], that allows storing data in a tiered fashion in different types of devices to reduce the total cost of disk storage [IMMR16]. The architecture proposed in this work, follows the multi-temperature paradigm by storing *hot* and *warm* data in different types of devices according to its popularity. Moreover, it stores *cold* data in a reduced representation, which enables an additional reduction of data storage costs.

With respect to query capabilities, in [JPT17], the authors discuss about the functionality offered by the TSMS; whether they offer support for Approximate Query Processing (AQP), and in particular, whether that AQP is sampling-based or model-based. To describe the functionality of the surveyed TSMSs, an uniform set of well-known terms (expressed using SQL) has been used. However, these are general purpose functionalities and not specific for time-series data. With the goal of establishing a standard benchmark of evaluating Time Series DataBase (TSDB) systems, in [LY19], an IoTDB-Benchmark framework specifically designed for TSDBs and IoT application scenarios is presented. In that benchmark three main categories of queries can be distinguished. To test the performance of the architecture proposed in this work, a query from each category (see Section 5.3.2) has been used. Moreover, an additional query type has been contemplated based on the survey presented in [JPT17], where an extra query capability is considered with regard to the data analytic capabilities of the compared systems.

Finally, regarding AQP, in [KMB+18], it is presented a model-based AQP approach where it is possible to answer queries with filter predicates and aggregation operations by using generative models learned over the complete database. In [KFP15] (and later in [LBP20]), *Plato*, a centralized TSMS that provides fast approximate analytics on time series, by precomputing and storing model-based compressed representations of time series is presented. In the proposed architecture in this work, since different time series approximation techniques are considered, sampling-based AQP is supported, by the reduced representations of time series obtained by applying techniques such as *Perceptually Important Points* (PIP), but also, model-based AQP, through the use of reduced representations based on mathematical models such as *Discrete Fourier Transform (DFT)*.

In summary, the proposed architecture combines the use of different approaches that had previously only been considered independently. In the following section, an overview of the proposed architecture is presented.

5.2 Overview of the Proposed Architecture

As mentioned before, in order to minimize the costs of storing endlessly accumulating sensor data, a three-level hierarchical architecture is proposed following the multi-temperature data management paradigm [SCS+12]. In the proposed architecture, the temperature tiers hot, warm, and cold are considered:

- *Hot storage:* In Smart Manufacturing scenarios, providing fast access to the latest data is essential for real-time applications [CFM+04]. Thus, in the first level of the proposed architecture, the most recent raw time-series data are stored on electronic non-volatile storage, such as SSDs. The high concurrency required for both, write and read operations over the data captured by thousands of sensors, make crucial the high write/read speeds of the SSDs. However, since the average cost of a SSD on the cloud is remarkably more expensive than the average cost of a HDD (0.196€ 1 GB per month and 0.046€ 1 GB per month respectively [Goo19b]), the most recent data are stored only for a short-period of time (e.g., one day), and then, are moved to the second level of the architecture.

- *Warm storage:* In the second level, recent raw time-series data are stored on HDDs (lower-cost devices which offer slower data transfer speed than SSDs). However, despite the lower-cost associated to these devices, the massive amount of data generated by thousands of sensors implanted in multiple manufacturing plants distributed worldwide during long periods of time, still generates a problem related to the considerable costs associated with the storage resources (even when storing the data in HDDs). Therefore, in the second level of the architecture data are stored only for a medium period of time (e.g., one month).

- *Cold storage:* In the third level, as data get older and are more rarely accessed, a reduced representation of the data, obtained by applying time series reduction techniques, is stored in HDDs for a longer period of time (e.g., more than one year). The lower-cost associated to these devices, together with the reduced storage space required to store the reduced representation of data, reinforce the interest of using this type of storage (even more, when the existing alternative is deleting the oldest data once the medium period of time defined for the warm storage has elapsed [Ama20]).

Figure 5.1 shows a high-level description of the proposed architecture and the data-workflow across the different services involved in it. The time-series data are captured by the sensors of a machine and are accessed by using a REST API (see Section 5.3.1). Then, those data are stored first, on the first level of the architecture (*Hot Storage*), for a short period of time (e.g., one day). Beyond that period of time, as new data arrives, the oldest data (e.g., the previous day's worth of data) are retrieved from the *Hot Storage* and are stored on the second level of the architecture (*Warm*

Storage), for a medium period of time. However, it is worth mentioning, that before time-series data are stored in the *Warm Storage*, they must be pre-processed by using the *Pre-processing Service* to ensure data quality for accurate data analysis [KCH+03]. After a period of time, as new data arrives, the oldest data (e.g., the oldest day's worth of data) are retrieved from the *Warm Storage* and reduced by using the *Reduction Service* before storing them in the third level of the architecture (*Cold Storage*), for a large period of time. When accessing the data, the *Storage Service* retrieves the data from the *Storage Infrastructure* and process them to answer different types of queries. When dealing with data from the third level of the architecture, data must be reconstructed from the reduced representations by using the *Reduction Service*.

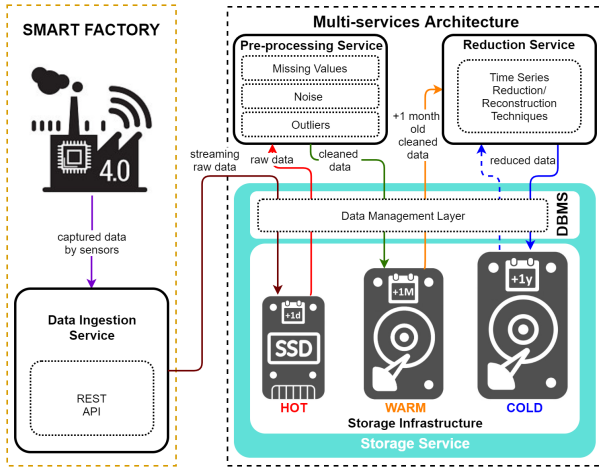


FIGURE 5.1: Three-level hierarchical architecture for time-series data storage on cloud environments

However, it should be noted that the proposed architecture is presented at a logical level; that is, although graphically the three levels are represented separately, its implementation could be materialized in different ways (using different partitions and disks distributed across the multiple nodes of a cloud computing infrastructure). For example, it could be materialized using only two disks, a SSD with raw data (hot storage) and a HDD with two partitions, one with raw data (warm storage) and another partition with reduced data (cold storage), or it could be materialized using three or more disks (those disks could be placed in a single node or distributed across various nodes).

Regarding the time series cleaning and reduction techniques used to pre-process and reduce the dimensionality of the data, many different techniques have been proposed in the literature (a review of them can be found in Chapter 3, Section 3.2.1). In the proposed architecture, the time series data have been cleaned and reduced by using the *Pre-processing* and *Reduction* services available in the system presented in Chapter 4. On the one hand, the *Pre-processing Service* provides different techniques for

detecting and handling outliers, to remove noise and to impute missing values in time series together with recommendations to select the appropriate techniques and parameter values required by them. On the other hand, the *Reduction Service* allows to obtain an adequate reduced representation of the time-series data, while preserving their main characteristics by applying the time series reduction techniques suggested by a machine learning-based model that given a time series, recommends the most appropriate reduction techniques.

Table 5.1 presents the different types of time series collected from an extruder machine of the real manufacturing scenario (see Section 5.3.1); the type of pre-processing and reduction techniques applied to them with their corresponding parameters; and the required *Data Storage Space* (in GB) for the files with the raw time series and the files with the reduced representation of the series. Moreover, the average reduction achieved (i.e., Reduction Potential (REDP)) by the selected reduction technique for each type of time series together with the lost information when approximating the time series (in terms of RMSE between the original time series and the approximated ones), are also presented in Table 5.1. For example, the following reduction techniques were applied: Discrete Fourier Transform (DFT) to *Head Zones Temperatures* time series, Perceptually Important Points (PIP) to *Operation Mode* time series and Run Length Encoding (RLE) to *Machine Time* time series. Regarding pre-processing techniques, *Last Observation Carried Forward (LOCF)*, *Linear Interpolation (LI)* and *Kalman smoothing on structural time series models* techniques were used to impute Missing Values (MV) to the time series.

Table 5.1: Properties of the time-series data and the applied pre-processing and reduction techniques

Time series types	**Pre–processing** Type:Tech.	**Reduction** Tech.:Param.	**Data Storage Space** Raw	Reduced	**REDP** (Avg)	**RMSE** (Avg)
Extruder Zones Temp. (°C)			97.94	4.84	95.06	0.00507
Filter Zones Temp. (°C)	MV: Kalman	DFT: 8640	105.46	5.21	95.06	0.01033
Head Zones Temp. (°C)			97.96	4.95	94.94	0.00543
Melting Temp. (°C)			16.33	0.80	95.08	0.00260
Operation Mode	MV: LI	PIP: 432	8.79	0.02	99.76	0.00024
Machine Times	MV: LOCF	RLE: -	14.51	0.04	99.76	0 (lossless)
Machine stop-working time	MV: LI	PIP: 432	19.41	0.05	99.73	0.00011
Cycling Time	MV: LI	PIP: 432	9.64	0.02	99.74	0.00023
Production counter	MV: LI	PIP: 432	23.95	0.05	99.78	0.00005
Screw speed (rpm)	MV: Kalman	DFT: 8640	8.16	0.63	92.29	0.00708
Melting Pressure (psi)	MV: Kalman	DFT: 8640	7.59	0.52	93.09	0.00927
Extruder motor consumption (amp)	MV: Kalman	DFT: 8640	8.14	0.63	92.29	0.0367

In order to evaluate the proposed three-level architecture, four different types of database engines have been considered in the *Storage Service* of the architecture. In the ranking presented in [Sol19], they are situated in the first position of their corresponding category. The selected DBMSs are: InfluxDB (a Time-series DBMS), Cassandra (a Wide Column store), MongoDB (a Document store) and Neo4J (a Graph DBMS). A detailed description of each of them can be seen in the technological background presented in Chapter 3, Section 3.2.2. For the evaluation, each database engine has been deployed on a Google Compute Engine instance. The

used machine for each instance has been a *n1-highmem-4*[1] *(4 vCPUs, 26 GB RAM)* with a standard persistent disk. The captured time-series data and their reduced representations have been inserted into each deployed database engine in two different databases: in one database, the values measured by the sensors and their associated timestamps; and in the other database, the name of the applied reduction technique, the first timestamp and the reduced representation of the time series.

The performance of the proposed architecture has been tested using the aforementioned DBMSs, under the perspective of two different main dimensions: used storage space (and an estimation of the associated costs), and total query time. Those dimensions have a real impact in the management of time-series data on the cloud, in order to reduce the associated costs while not hampering the suitability of the systems for real use cases. However, taking into account that the only difference between the first and the second level of the architecture lies in the speed of the type of disk considered, SSD in the first level, and HDD in the second level; and that the use of SSDs to store the raw data in the first level is mandatory (due to the high concurrency required for both, write and read operations over the captured data), the tests have only been focused on the comparison of the performance of the second and third level of the architecture (in sections 5.4 and 5.5, performance results are presented).

5.3 Setting of the Proposed Architecture

The proposed architecture has been tested by using industrial time series coming from the real-world context within this research work has been carried out, and with four different types of queries proposed by domain experts from this context. In this section it is presented, first of all, the manufacturing setting from which the data comes; next, the type of queries, proposed by domain experts from that manufacturing setting; and finally, some features of the evaluation framework.

5.3.1 Smart Manufacturing Scenario: Data Provenance

The data used for testing the performance of the proposed architecture come from the real-world manufacturing context presented in Chapter 3, Section 3.1. In particular, the data came from an extruder machine from a plastic bottles production plant based on an extrusion process, on which the CEM has installed 51 sensors. Those sensors, generate 51 time series of the different types shown in Table 5.1. For the tests, a month's worth of data captured by those sensors have been gathered by using a REST API, through which the ITS Provider grants the access to the data stored in their platform solution.

Furthermore, in order to test the performance of the proposed architecture with a higher volume of data, the data generated during a year by

[1] Machine types in Google Compute Engine: `https://cloud.google.com/compute/docs/machine-types`

an entire manufacturing plant, on which 10 extruder machines are operating, has been simulated by generating synthetic time-series data. Those synthetic data have been generated by first, replicating the time series from the original machine, and then, adding some Gaussian noise [Kaf86] following the approach proposed in [KBSM14], on which white noise is generated by using a random number generator following a normal distribution N(μ, σ^2) with a mean of 0 ($\mu = 0$) and a standard deviation of 0.05 ($\sigma = 0.05$). This ensures that the synthetic data is not identical to the original one, in order to avoid data to be compressed in the databases for being identical, keeping the shape and the properties of the time series produced by the sensors of that nature.

5.3.2 Type of Queries

Four different types of queries proposed by domain experts from the CEM have been used in the testing task over three different time-windows (day, week and month). Three of them correspond to the three main query categories of the benchmark proposed in [LY19]: time-based filtering queries *(Query type 1)*; aggregated queries *(Query type 2)*; and value-based filtering queries *(Query type 3)*. Moreover, an additional type of query *(Query type 4)* has been used to detect timestamps on which the defined correlations among the values of different sensors are not held. Next, the queries used in the test are presented:

- *Query type 1:* Gets data from a single sensor for each time-window. For example, *get data of Melting Pressure from t_0 to t_n*. These types of queries get information that is shown by the majority of platforms that provide visual surveillance of all raw data captured from the sensors implanted in the manufacturing plant through multi-purpose dashboards.[2]

- *Query type 2:* Gets data from a single sensor during a period of time but, unlike the previous type of query, an aggregation function (e.g., min, max, avg, etc.) is applied over the obtained data by using three different time-lapses (minute, hour and day). For example, *get avg(data) of Melting Pressure from t_0 to t_n by minute.*

- *Query type 3:* Gets data from a sensor when other sensors' data are equal, greater or lower than a specific value. Given the typology of the question, only queries over related sensors have been considered. For example, *get data of Melting Pressure from t_0 to t_n when Screw Speed $< v_0$*. This type of queries are not usually supported by the mentioned multi-purpose dashboards.

- *Query type 4:* Gets the timestamps when the defined correlations between the values of different sensors are not held. For example, it is established as a normal operation (by the domain experts from the CEM) that when the *Spindle speed* increases, the *Melting pressure*

[2]Example of a multi-purpose dashboard: `https://logz.io/blog/grafana-vs-kibana/`

and the *Extruder motor consumption* increase as well, therefore, each variation in the predefined correlation can be used to locate anomalies in the data.

In order to perform the tests as objectively as possible, first, the queries have been written in a standard SQL language; and then, they have been translated into the corresponding query expression, using the query language supported by each DBMS. Next an example of the *Query Type 2* is presented, by using, first, the standard SQL language, and then, the concrete queries used on each DBMS.

```
SELECT COUNT(*), AVG(value), MIN(value), MAX(value)
FROM WMT_GTFE7Q
WHERE sensor_id = 'VMTKD6' AND timestamp >= '2019-01-01 00:00:00Z' AND
      timestamp < '2019-01-02 00:00:00Z'
GROUP BY DATEPART(hour, timestamp)
```

Example of Query Type 2 with Standard SQL

```
SELECT COUNT(*), MEAN(value), MIN(value), MAX(value)
FROM WMT_GTFE7Q
WHERE sensor_id = 'VMTKD6' AND time >= '2019-01-01T00:00:00Z' AND time < '
      2019-01-02T00:00:00Z'
GROUP BY time(1h)
```

Example of Query Type 2 in InfluxDB

```
SELECT sensor_id, date, hour, count(value), avg(value), min(value), max(value)
FROM WMT_GTFE7Q
WHERE sensor_id='VMTKD6' AND date >= '2019-01-01' AND date < '2019-01-02'
GROUP BY sensor_id, date, hour
ALLOW FILTERING
```

Example of Query Type 2 in Cassandra

```
db.WMT_GTFE7Q.aggregate(
  [{"$match": {timestamp: {$gte: ISODate("2019-01-01T00:00:00Z"),
  $lt: ISODate("2019-01-02T00:00:00Z")}}},
  {$group: {
        _id: {year: {$year: "$timestamp"},
            month: {$month: "$timestamp"},
            day: {$dayOfWeek: "$timestamp"},
            hour: {$hour: "$timestamp"}},
        avg: {$avg: "$values.VMTKD6"},
        min: {$min: "$values.VMTKD6"},
        max: {$max: "$values.VMTKD6"},
        count: {$sum:1}}}])
```

Example of Query Type 2 in MongoDB

```
MATCH (s:Sensor{id:'VMTKD6'})-[:MAKE_OBSERVATION] -> (o:Observation{year:2019,
      month:1, day:1})
WITH s.id as sensor, o.year as year, o.month as month, o.day as day, o.hour as
      hour,
    count(o.value) as count, avg(o.value) as avg, min(o.value) as min, max(o.
        value) as max
RETURN sensor, year, month, day, hour, count, avg, min, max;
```

Example of Query Type 2 in Neo4J

The queries have been executed to access the data stored in raw format on the second level of the architecture and in the reduced representation on the third level. However, as mentioned in Chapter 4, Section

4.2.2 some information is lost when reconstructing the time series from the reduced representation (due to the use of lossy dimensionality reduction techniques), and thus, it is worth mentioning that this can impact on the results retrieved from the queries. Therefore, next the possible consequences of using the approximated series on each type of queries are shown:

- *Query type 1* and *Query type 2*: The first and the second type of queries usually are performed for data visualization purposes, and thus, the consequences only depends on the relevance of the data itself, and according to domain experts from the manufacturing setting of this work, an approximation of the data is sufficient for the services that they provide at present (even more when the existing alternative is not offering an answer because the data has been removed to free resources for new data).

- *Query type 4*: The fourth type of query is related to an analytical use case based on pattern matching, so as mentioned in Chapter 4, Section 4.2.2, using approximated series should be enough. For example, for the use case mentioned in this work, in order to locate anomalies in the data, since the underlying shape of the time series is maintained in the reduced representation, the analysis can be performed also over the approximated series.

- *Query type 3*: The third type of query could be the one on which the use of approximated series had the greatest impact. For example, when looking for an exact value (e.g., *get data of Melting Pressure from t_0 to t_n when Screw Speed = 180*) it could happen that some of the values around 180 have been slightly changed (e.g., to 180.01 or 179.98), due to the approximation, and thus, the query result may not match exactly. However, usually this type of queries are used with greater or lower operators that are less sensitive to this problem, and usually, the equal operator is used over discrete data (e.g., *get data of Melting Pressure from t_0 to t_n when Operation Mode = 1*), that usually are reduced with the RLE technique (lossless compression).

5.4 Data Storage Space

This section provides details about the obtained results when implementing the architecture with the different database engines. First, the reduction potential of the dimensionality reduction techniques and the reduction of data storage resources through the use of these techniques in the third level of the architecture are presented. Then, the interest of applying these techniques to reduce data storage resources in different Smart Manufacturing scenarios is shown; and finally, the potential reduction of the costs associated to the data storage resources is estimated.

5.4.1 Data Storage Space: Reduction Potential

To test the performance of the proposed architecture under the perspective of data storage space, first the required data storage resources on each database engine have been measured. For that purpose, the data provided by the CEM together with the generated synthetic time series have been inserted on each database engine (in raw and reduced format) and the occupied disk space has been measured across the time.

In Table 5.2, it can be observed the difference between the space required to store the data in raw format and in the reduced format in each database engine. It can be noticed that Neo4J uses 1968 GB to store all the raw time series and 19 GB to store the reduced ones, while InfluxDB just uses 49 GB for the original and 11 GB for the reduced data. However, it is worth mentioning that InfluxDB is a specific system for time series storage, and therefore, it is optimized for the storage of this kind of data, while Cassandra and MongoDB are more general purpose systems; and Neo4J is a graph oriented DBMS (they are not optimized for time series storage). As a result, when dealing with raw time series, there is a bigger difference in the required data storage space by each database engine than when dealing with the reduced representations, as they do not deal with time-series data as such.

Table 5.2: Data storage space (in GB) per each database engine for reduced and raw data across months & average Reduction Potential (REDP %±std)

DBMS	1 Month		6 Months		12 Months		REDP
	Raw	Red.	Raw	Red.	Raw	Red.	(%±std)
InfluxDB	4.11	1.02	24.44	5.27	49.39	11.04	78.05 ± 0.011
Cassandra	12.97	1.00	77.33	5.82	155.83	11.73	92.45 ± 0.001
MongoDB	20.39	1.40	119.08	8.19	240.13	16.53	93.12 ± ≈ 0
Neo4J	167.17	1.58	976.06	9.22	1968.30	18.60	99.06 ± ≈ 0

Moreover, in Table 5.2, it can be observed the percentage of space needed to store the reduced representation versus raw storage for each database engine (i.e., Reduction Potential (REDP)). Although a considerable variation on the reduction potential of each system can be observed; for example, InfluxDB allows to reduce the storage space on a 78.05% (as it is already optimized for this kind of data), while Neo4J allows to reduce it on a 99.06%; notice that the proposed architecture allows to achieve an important reduction (greater than 78%) of the data storage resources in all the considered systems. It is worth mentioning that the achieved reduction potential keeps constant when reducing different months' worth of data from different machines (see the standard deviation in Table 5.2 which is close to 0). Therefore, the reduced data storage space is proportional to the raw data volumes (the amount of months' worth of data and the number of machines), which allows to extrapolate the analyzed results to larger manufacturing scenarios.

The reduction of data storage space has been achieved by applying time series reduction techniques to the different time series available in

the considered scenario. However, the heterogeneous nature of the time series [NHMG20], makes them susceptible to be reduced with different reduction techniques, each of them with a different reduction potential. In order to see how the considered techniques have contributed to the achieved reduction of the data storage space, Table 5.3 shows, for each type of time series mentioned in Table 5.1, the required data storage space (in raw and reduced format) to store the data on each DBMS; and the achieved reduction (in terms of percentage of the total reduction potential).

TABLE 5.3: Captured time-series data storage size (in GB) on each DBMS together with the Reduction Potential (REDP)

Time series types	InfluxDB			Cassandra			MongoDB			Neo4J		
	Raw	Red	REDP	Raw	Red	REDP	Raw	Red	REDP	Raw	Red	REDP
Extruder Zones Temp.	13.40	2.92	20.50	36.58	3.30	26.02	57.00	4.48	25.21	451.39	5.03	26.78
Filter Zones Temp.	13.09	3.13	22.01	38.18	3.55	27.95	61.73	4.81	27.12	488.54	5.41	28.80
Head Zones Temp.	17.69	3.05	21.46	39.48	3.42	26.94	57.02	4.61	25.96	451.40	5.18	27.58
Melting Temp.	1.79	0.48	3.38	5.79	0.54	4.29	9.50	0.74	4.18	75.23	0.83	4.44
Operation Mode	0.08	0.01	0.06	2.54	0.01	0.08	4.77	0.02	0.11	48.06	0.02	0.12
Machine Times	0.16	0.01	0.10	6.85	0.003	0.02	9.01	0.03	0.18	75.25	0.06	0.32
Mach. stop-working time	1.54	0.02	0.16	5.92	0.03	0.20	9.28	0.05	0.28	103.63	0.06	0.31
Cycling Time	0.08	0.01	0.06	2.54	0.01	0.08	4.77	0.02	0.13	49.15	0.03	0.15
Production counter	0.33	0.04	0.27	7.97	0.04	0.32	13.52	0.05	0.28	112.83	0.06	0.31
Screw speed	0.34	0.49	3.45	3.14	0.28	2.18	4.51	0.61	3.42	37.61	0.68	3.64
Melting Pressure	0.19	0.38	2.70	2.90	0.28	2.23	4.51	0.50	2.82	37.61	0.56	2.99
Extruder motor consump.	0.69	0.50	3.49	3.95	0.27	2.16	4.51	0.61	3.41	37.61	0.68	3.62
Total	49.39	11.04	77.64	155.83	11.73	92.47	240.13	16.53	93.12	1968.30	18.60	99.06

As mentioned before, Table 5.2 reflects a comparison between storing all the data in raw format in the second level architecture (i.e., *Warm Storage*) and storing the reduced representation of all the data in the third level (i.e., *Cold Storage*). Nevertheless, in a real scenario, the data are stored in the second level of the architecture for a fixed period of time, and beyond this period of time, a reduced representation of them is generated and stored in the third level of the architecture (removing the original raw data from the second level and freeing up space for new incoming raw data). In these scenarios, the required data storage resources for each level of the architecture vary according to the companies policy when implementing the multi-temperature database paradigm. Consequently, the proposed three-level architecture can be deployed to store different time-windows of data on each level.

As an example, Figure 5.2 shows how the required data storage resources (in GB) varies when different percentages of the data are stored at each level of the architecture on each database engine. It can be noticed that, for example, in Neo4J the storage of the data in raw format (i.e., using only the second level of the architecture) requires 1968 GB while storing the 25% of the data in reduced representation and the 75% in raw format requires 1481 GB; storing the 50% of the data on each level of the architecture requires 993 GB and lastly; storing the 75% of the data in reduced representation and the 25% in raw format requires 506 GB. Moreover, in Table 5.4, it can be seen the obtained reduction potential considering different percentages of the data stored in each level of the architecture. It can be observed that the storage resources saving is more significant as the percentage of data stored in the third level increases. The required

data storage space to store the data with the proposed architecture, can be computed with the Equation 5.1.

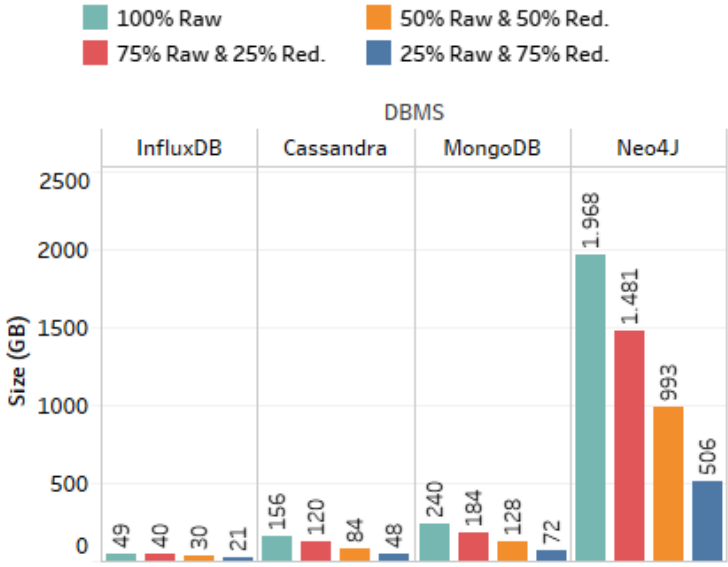


FIGURE 5.2: Data storage space comparison between different distributions

TABLE 5.4: Data Storage Space (DSS) and Reduction Potential (REDP) across different distributions

DBMS	DSS & REDP	D_1	D_2	D_3	D_4
InfluxDB	DSS (in GB)	49.39	39.82	30.25	20.67
	REDP (in %)	0	19.38	38.76	58.15
Cassandra	DSS (in GB)	155.83	119.81	83.78	47.75
	REDP (in %)	0	23.12	46.24	69.36
MongoDB	DSS (in GB)	240.13	184.23	128.33	72.42
	REDP (in %)	0	23.28	46.56	69.84
Neo4J	DSS (in GB)	1968.3	1480.87	993.45	506.02
	REDP (in %)	0	24.76	49.53	74.29

- D_1=100% Raw data
- D_2=75% Raw data 25% Reduced data
- D_3=50% Raw data 50% Reduced data
- D_4=25% Raw data 75% Reduced data

$$DSS_{DBMS} = (\text{Raw}_S * \text{Raw}_P) + (\text{Red}_S * \text{Red}_P * (1\text{-REDP}_{DBMS})) \quad (5.1)$$

where: DSS_{DBMS} = Data storage space in a particular DBMS
Raw_S = Storage space of the raw data
Red_S = Storage space of the reduced representation
Raw_P = % of data in the 2^{nd} level
Red_P = % of data in the 3^{rd} level
$REDP_{DBMS}$ = Reduction potential of a particular DBMS

5.4.2 The Three-Level Hierarchical Architecture on Different Industry 4.0 Scenarios

As stated before, the amount of data considered in the previous evaluations corresponds to a single manufacturing facility of the considered CEM, supplying plastic bottles production plants based on an extrusion process. However, considering the real Smart Manufacturing scenario, these data represent, only, a subset of the real amount of data generated by all the machines from all the manufacturing plants distributed worldwide supplied by this CEM (totaling about[3] 168 machines). Moreover, regarding data generation capabilities, the machines of this CEM have a low profile compared with the machines of other CEMs providing different manufacturing sectors, to which the ITS Provider supplies services: for example, the machines of the polyurethane foam production plant presented in Chapter 4, Section 4.2 have about 400 sensors capturing data at 1Hz, and the machines produced by a CEM supplying manufacturing plants in the aerospace machining and grinding sector could have about 2000 sensors capturing data at 1Hz. In this work, these three CEMs have been considered to test the performance of the proposed architecture with different companies managing different volumes of industrial data, however, for simplicity, from here on these CEMs will be named as *Small CEM*, *Medium CEM* and *Big CEM* respectively, taking into account their data generation capabilities.

In order to see the potential impact of the proposed architecture in other CEMs with more powerful machines (in terms of data generation capabilities), a prototype machine (M_P) has been defined as a reference unit by using the configuration of the extruder machine (M_{Small}) shown in Section 5.3.1. This reference unit allows to extrapolate the conducted analysis to more powerful machines from other companies by approximating different machines in terms of the prototype machine. For example, a machine of the *Medium CEM (M_{Medium})* could be represented as $8M_P$ $((400sensors/51sensors) * (1Hz/1Hz) \approx 8M_p)$ and a machine of the *Big CEM (M_{Big})* could be represented as $40M_P$ $((2000sensors/51sensors) * (1Hz/1Hz) \approx 40M_P)$.

[3]The number of machines has been estimated considering the number of machines in a manufacturing facility and an approximation of the number of facilities on each plant and the number of plants to which the CEM supplies equipment.

Table 5.5, shows the number of machines prototype (M_P) for each CEM mentioned previously (considering that all of them supply the same number of machines as the *Small CEM*, that is, 168). Furthermore, the scenario of the ITS Provider has also been included in the evaluation process. For this scenario, the number of prototype machines (M_P) of the three CEMs previously considered has been computed and applied to the total number of CEMs to which the ITS Provider from the real-world scenario supplies data exploitation services (in particular, 12 CEMs [Niñ17]).

TABLE 5.5: Size of companies in terms of M_P depending on their machine types

Company	Sector	Sensors	Machine Type	Prototype Machines
Small	Plastic products based on an extrusion process	51 (1Hz)	$1M_{Small} = 1M_P$	$168M_P$
Medium	Polyurethane foam production	400 (1Hz)	$1M_{Medium} = 8M_P$	$1344M_P$
Big	Aerospace grinding and machining	2000 (1Hz)	$1M_{Big} = 40M_P$	$6720M_P$
ITS Provider	Smart Manufacturing and Smart Services	51 (1Hz)	$1M_P$	$32928M_P$

Figure 5.3 shows a comparison of how the required data storage resources increase along the time for the different scenarios described in Table 5.5 and the ITS Provider when storing the data in raw format and their reduced representation in Cassandra (see the Appendix A for the other database engines). The heat map of the figure has been established following the data generation capabilities of different manufacturing scenarios of different magnitudes categorized in [Cut14]. That categorization states, that conservative estimates, based on data generated from manufacturing devices and sensors in different types of manufacturing facilities show that: a small facility could generate 2 to 10 terabytes in a year; a medium facility 5 to 25 terabytes in a year; and a large one 16 to 80 terabytes in a year.

In the heat map, it can be seen that when dealing with the raw data, the volumes of data start to acquire a great magnitude in large, medium and short terms for the Medium CEM, Big CEM and the ITS Provider respectively, while when dealing with the reduced representation, only starts to acquire a considerable magnitude in a large term for the ITS Provider. As mentioned before, the reduced data storage space is proportional to the raw data volumes, thus, although scenarios on which the volumes of data are not really big could benefit from storing data in a reduced representation, the real interest of applying the proposed architecture starts when storing big amounts of data during long periods of time, and it is in a scenario with huge volumes of data (such as the one of the ITS Provider from the real-world scenario of this research work), where the proposed architecture acquires a greater relevance.

Finally, regarding the time-window of the stored time series, in Figure 5.4, it can be observed the amount of months' worth of data that can be stored in a space of 80 TB by using different distributions of the data in the proposed architecture (considering the scenario of the Big CEM). It can

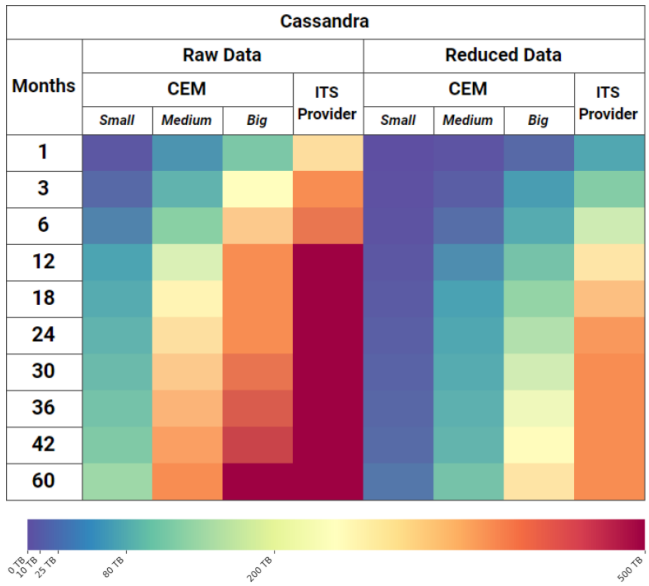


FIGURE 5.3: Volumes of data generated (in TB) across time for different scenarios in Cassandra

be noticed that the amount of months' worth of data that can be stored is remarkably bigger for the reduced representation of the data. Those 80 TB would allow to store about nine months' worth of raw data in Cassandra, six months' worth of raw data in MongoDB and a single month's worth of raw data in Neo4J, while they allow to store more than six years' worth of the reduced representation of the data. It would be possible to store 30 months' worth of raw data using InfluxDB; however, there are other scenarios, such as ITS Providers or CEMs with bigger data generation capabilities, where the handled volumes of data are larger and therefore, using reduced representations can also be interesting when dealing with InfluxDB.

5.4.3 Data Storage Space: Costs

In order to give some idea of the costs of storing the different representations of the data in the considered database engines, it has been considered as unit of cost, the one that corresponds to store 1 GB on a standard persistent disk in the Google Cloud Platform (0.046€ per month) [Goo19b], and this cost has been applied to the scenario of the Big CEM. For that, the size occupied by a single sensor measurement on each database engine has been computed by averaging the size of all the measurements of all sensors in the considered scenario. Those sizes have been the reference used to compute the data storage cost for that CEM in terms of disk storage price (leaving aside the costs related to the database server, etc.). Figure 5.5 shows how the storage costs associated to different distributions of the data in the second and the third level of the architecture increases

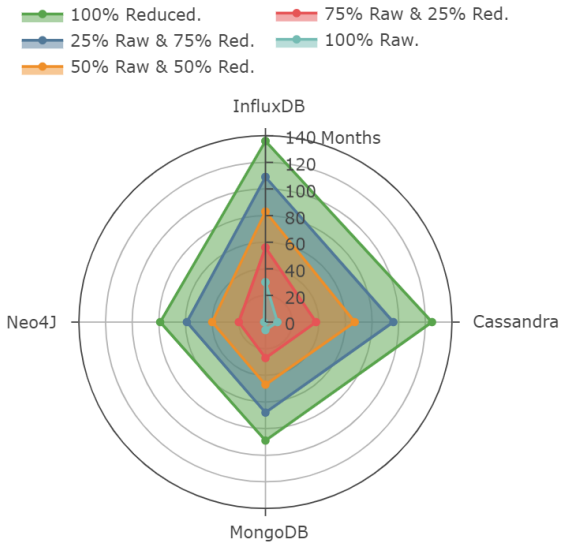


FIGURE 5.4: Months' worth of data stored in a 80 TB Storage. Comparison between different distributions of the data in the proposed architecture for a Big CEM

along the time. It can be observed, that the costs associated to the raw data greatly increases as time goes by and larger volumes of data are gathered, while for the reduced representation, the growth is remarkably less pronounced.

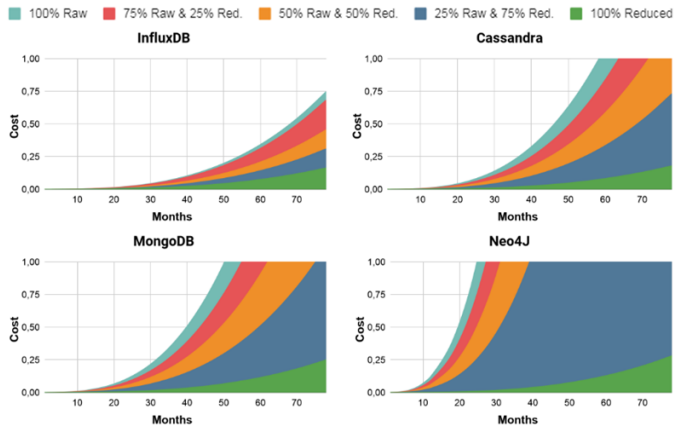


FIGURE 5.5: Data storage associated costs of a Big CEM along the months

From the business perspective of a company in the category of the ITS Provider, as time goes by, the storage of the data results less profitable, since at a given time, the data storage cost will be the cost associated to store the new data generated at that time, plus the accumulative cost of having stored the data in the past. Therefore, the more data are kept

for longer periods of time, the more interesting it results to store data in a reduced representation. The following formula (Equation 5.2) can be used to calculate the average economic savings of storing the reduced representation of the data for a given DBMS:

$$\text{Average savings in €} = \alpha \times \beta \times \frac{m(m+1)}{2} \times \delta_{DBMS} \tag{5.2}$$

where: m = Number of months to calculate savings
α = Size of one month's worth of raw data (GB)
β = Monthly storage cost per GB (€)
δ_{DBMS} = Avg. reduction potential for DBMS (see Table 5.2)

5.5 Total Query Time

Section 5.4 has shown the reduction of data storage resources that can be achieved by applying time series dimensionality reduction techniques in the third level of the architecture. Moreover, the proposed architecture should not hamper the suitability of the DBMSs for accessing the data, in a context such as the Industry 4.0, where the use cases that can deal with those data can be very different. With respect to the expected response times, although a short latency is desirable for all the use cases, this is not always mandatory, in fact, as it is stated in [FSBR17]: *"Today, many Industry 4.0 scenarios do not require a short latency between data collection and output reaction, but it is expected that short latency services would be seen by the market as a distinctive quality."* Furthermore, the response times expected for those use cases may vary based on many factors: the time-window, the data exploitation purpose, the accessed data period, etc.

In the particular context on which this research work has been carried out, according to domain experts from the manufacturing setting, until now, data is stored with two main purposes: on the one hand, real time visualization of the data for providing visual surveillance of the production process through multi-purpose dashboards; and on the other hand, for further data analytics over old data records. According to those experts, for the first purpose, real-time access to the data is mandatory; therefore, the most recent data (which is usually the most accessed data in these scenarios [CFM+04]) are stored on SSDs, on the first level of the architecture. For the second purpose, there is not any special requirement on the data access times; however, the introduction of the third level should not increase excessively the response times of the second level of the architecture. Thus, this section compares the response times of the second and the third level of the architecture, when answering four different types of queries proposed by the domain experts (see Section 5.3.2).

Those queries have been executed five times over the month's worth of data captured from the real extruder machine and the total average times have been computed. Figure 5.6 shows the types and execution order of the different types of times used in this section. Some of them are not

required depending on the case (e.g., the data reconstruction time does not exist for raw data). Table 5.6 and Table 5.7 shows a summary of the total query time for each query type (Q.) executed on each DBMS and over three different time-windows (day, week and month) per each type of data (Raw and Reduced Series respectively). It can be observed that the query execution time (values in orange) when dealing with reduced representations is shorter than the query execution time over the raw data, since the amount of data to retrieve in the reduced representation is lower than in the raw format. For example, a single record is recovered for a particular day in the reduced representation, in contrast to the 86400 records (one per second) recovered when accessing the data in the raw format.

FIGURE 5.6: Diagram of the different types of times

TABLE 5.6: Total Query times summary (in ms) including Query Execution, Data Reconstruction and Data Processing times for Raw Data

Q.	DBMS	Raw Series Total Time = (QueryTime) or (QueryTime+ProcTime) Day	Week	Month
1	InfluxDB	79.23 (79.23)	635.03 (635.03)	2045.13 (2045.13)
	Cassandra	223.41 (223.41)	1456.31 (1456.31)	5346.62 (5346.62)
	MongoDB	260.15 (260.15)	1605.99 (1605.99)	4973.87 (4973.87)
	Neo4J	2401.26 (2401.26)	6095.39 (6095.39)	9094.42 (9094.42)
2	InfluxDB	74.44 (74.44)	484.84 (484.84)	1544.01 (1544.01)
	Cassandra	337.55 (337.55)	2135.27 (2135.27)	6728.42 (6728.42)
	MongoDB	365.15 (365.15)	2290.91 (2290.91)	7155.95 (7155.95)
	Neo4J	2248.41 (2248.41)	6462.28 (6462.28)	11507.77 (11507.77)
3	InfluxDB	71.60 (71.60)	58.33 (58.33)	664.89 (664.89)
	Cassandra	1019.29 (1019.29)	6027.16 (6027.16)	22544.35 (22544.35)
	MongoDB	139.46 (139.46)	877.08 (877.08)	2718.43 (2718.43)
	Neo4J	2417.06 (2417.06)	3079.56 (3079.56)	7410.76 (7410.76)
4	InfluxDB	4657.44 (357.44+4300.00)	24157.45 (2707.45+21450.00)	101837.36 (8487.36+93350.00)
	Cassandra	4980.36 (680.36+4300.00)	25329.06 (3879.06+21450.00)	107283.92 (13933.92+93350.00)
	MongoDB	4556.20 (256.20+4300.00)	23026.40 (1576.40+21450.00)	98278.40 (4928.40+93350.00)
	Neo4J	11828.87 (7528.87+4300.00)	36806.39 (15356.39+21450.00)	119963.15 (26613.15+93350.00)

However, as data cannot be directly accessed on the reduced representations, each reduced time series must be first obtained from the DBMS, and then, reconstructed, which requires an extra time *(RecTime)*. Furthermore, some analytic operations cannot be applied directly over the reduced representations of the data (for example, aggregate functions that appear in queries of type 2), thus, the reconstruction of the time series is also followed by the application of the processing operations over them, which

TABLE 5.7: Total Query times summary (in ms) including Query Execution, Data Reconstruction and Data Processing times for Reduced Data

Q.	DBMS	Reduced Series Total Time = (QueryTime+RecTime) or (QueryTime+RecTime+ProcTime) Day	Week	Month
1	InfluxDB	488.69 (0.02+488.67) 491.39	605.68 (0.04+605.64) 608.01	2628.39 (0.09+2628.30) 2634.59
	Cassandra	(2.72+488.67) 488.74	(2.37+605.64) 605.74	(6.29+2628.30) 2628.52
	MongoDB	(0.07*+488.67) 500.89	(0.10*+605.64) 629.62	(0.22*+2628.30) 2679.77
	Neo4J	(12.22+488.67)	(23.98+605.64)	(51.47+2628.30)
2	InfluxDB	2849.52 (0.02+488.67+2360.83) 2852.22	10253.35 (0.04+605.64+9647.67) 10255.68	29102.50 (0.09+2628.30+26474.11) 29108.70
	Cassandra	(2.72+488.67+2360.83) 2849.60	(2.37+605.64+9647.67) 10253.50	(6.29+2628.30+26474.11) 29102.83
	MongoDB	(0.10*+488.67+2360.83) 2861.14	(0.19*+605.64+9647.67) 10276.63	(0.42*+2628.30+26474.11) 29161.40
	Neo4J	(11.64+488.67+2360.83)	(23.32+605.64+9647.67)	(58.99+2628.30+26474.11)
3	InfluxDB	778.73 (0.05+128.69+649.99) 784.94	3971.79 (0.08+219.14+3752.57) 3978.71	21458.12 (0.20+3024.51+18433.41) 21476.32
	Cassandra	(6.26+128.69+649.99) 778.78	(7.00+219.14+3752.57) 3972.04	(18.40+3024.51+18433.41) 21458.74
	MongoDB	(0.10*+128.69+649.99) 794.75	(0.33*+219.14+3752.57) 4028.24	(0.82*+3024.51+18433.41) 21611.70
	Neo4J	(16.07+128.69+649.99)	(56.53+219.14+3752.57)	(153.78+3024.51+18433.41)
4	InfluxDB	4788.71 (0.04+488.67+4300.00) 4790.38	22055.73 (0.09+605.64+21450.00) 22057.33	95978.51 (0.21+2628.3+93350.00) 95980.62
	Cassandra	(1.71+488.67+4300.00) 4788.70	(1.69+605.64+21450.00) 22055.69	(2.32+2628.3+93350.00) 95978.42
	MongoDB	(0.03*+488.67+4300.00) 4826.98	(0.05*+605.64+21450.00) 22116.05	(0.12*+2628.3+93350.00) 96128.92
	Neo4J	(38.31+488.67+4300.00)	(60.41+605.64+21450.00)	(150.62+2628.3+93350.00)

* As the precision for the execution time of MongoDB queries is given in ms, the obtained average was 0 ms. However, the provided time value has been estimated by using the incremental ratio respect to InfluxDB.

introduces additional extra time *(ProcTime)*. Those processing operations have been executed using *Spark*, an open-source distributed general-purpose cluster-computing framework [Spa19], over the reconstructed time series. Finally, the sum of the data reconstruction time (red values in Table 5.7) and the data processing time (values in blue) must also be added to the query execution time (values in orange) over the reduced representations, resulting in the total query time (values in black).

Regarding raw data, notice that for query types 1, 2 and 3; the data can be accessed and processed by using the queries directly (no reconstruction and processing times need to be added), therefore, the total query time is simply the query execution time. For query type 4 an extra data processing time is required to detect anomalous behaviours, both in raw and reduced data.

Figures 5.7 and 5.8 show the results of the total query time measures (in logarithmic scale) for the first query type (accessing a day's and a month's worth of data of a single sensor, respectively). It can be seen that the required time to reconstruct the time series from the reduced representations penalizes the total query time. Nevertheless, this penalization is less significant as the time-window increases because the reconstruction of each time series is parallelized.

Regarding the reduced data, when the queries involve some processing operations over the data, the data processing time, in contrast to the data reconstruction time, increases linearly to the time-window. However, as

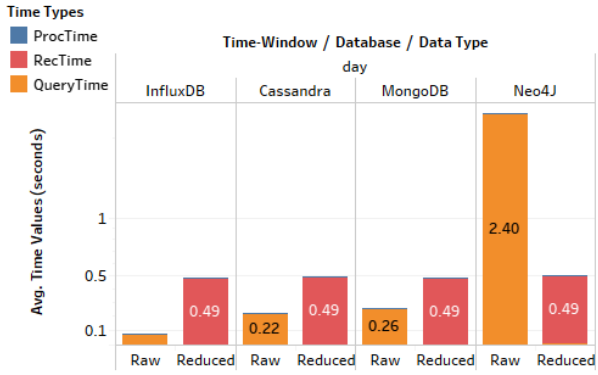


FIGURE 5.7: Query Type 1 execution times for one day's worth of data

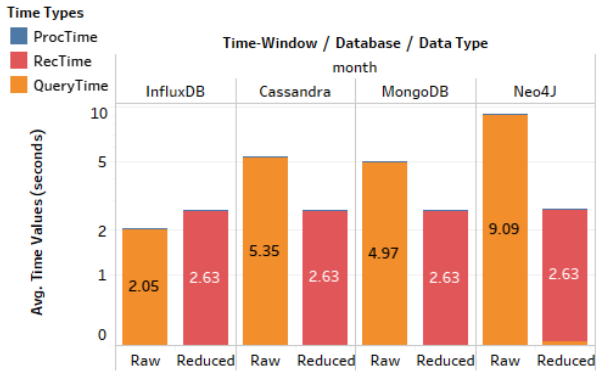


FIGURE 5.8: Query Type 1 execution times for one month's worth of data

it can be seen in figures 5.9 and 5.10, the introduced latency for the processing and reconstruction of the time series is assumable (lower than 30 seconds for a month's worth of data, for the second query type; which is in line with the times obtained by other works in the area of big data, handling similar amounts of data [BBHE18]), taking into account the potential savings on storage space. A similar behaviour has also been observed on the third query type, where the latency is lower than 22 seconds.

Figures 5.11 and 5.12 show the total query time (in logarithmic scale) for the fourth query type, defined as the detection of the timestamps when the given correlations between the values of different sensors are not held, for a day and a month respectively. As it can be observed, there is a slight time difference in favor of the reduced representations of the data when a high time consuming process is carried out, due to the data processing time, which is indifferent to the data storage technique (thus, it is the same for both representations), and it is significantly longer than the other times involved in the query (i.e., query execution time and data reconstruction time).

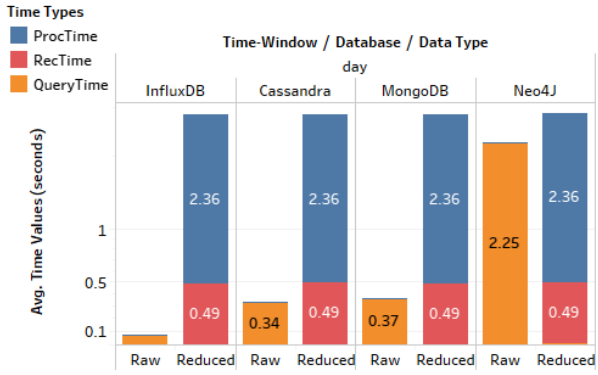


FIGURE 5.9: Query Type 2 execution times for one day's worth of data

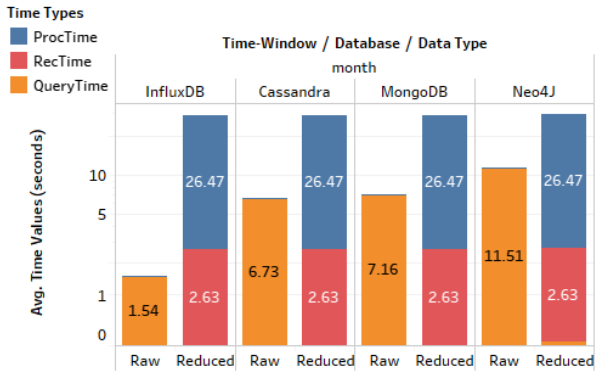


FIGURE 5.10: Query Type 2 execution times for one month's worth of data

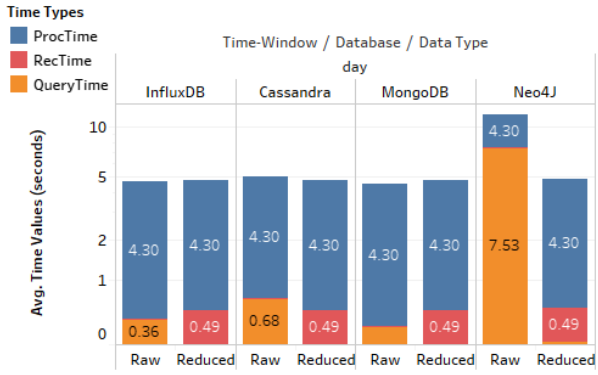


FIGURE 5.11: Query Type 4 execution times for one day's worth of data

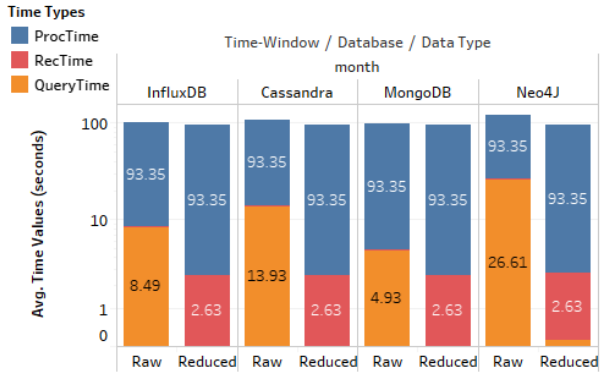


FIGURE 5.12: Query Type 4 execution times for one month's worth of data

5.6 Selection of a Suitable Architecture for a Smart Manufacturing Scenario

Previous sections have presented the proposed three-level architecture under two different perspectives: on the one hand, Section 5.4 has shown the potential data storage savings that could be achieved by introducing the third level of the architecture. On the other hand, Section 5.5 has shown the response times for different queries when accessing the data. The performed tests have shown that the proposed architecture allows to optimize cloud storage resources (and its associated costs) without hampering the suitability of the DBMSs for the management of industrial time-series data. However, the achieved reduction in data storage resources, varies depending on the level of implantation of the third level of the architecture (see Figure 5.2), while the data access times, varies depending on the location of the data (accessing raw data in the second level of the architecture or accessing a reduced representation of the data in the third level), the type of query and the time-window. Thus, for each specific scenario, with its own particular requirements, the most suitable configuration of the proposed architecture may vary. Therefore, in this section, a method for selecting the most suitable architecture for a Smart Manufacturing scenario is presented. That selection, can be seen as a multi-criteria decision problem for which the Design of Experiments methodology (DoE) [Ast12] is an effective statistical tool to understand and optimize processes and products, maximizing the information obtained from the data resulting from the experimentation.

In this work, the simplified methodology for applying the DoE in industrial environments, presented in [Tan08], has been used to design an experiment, as a proof of concept to test the potential of these techniques for helping in the selection of the most suitable configuration of the architecture. This methodology consists of seven consecutive steps. Figure 5.13 shows these steps and the activities that they entail, in addition to the different variables involved in each of these steps. Moreover, the following subsections describe the process carried out to complete the DoE.

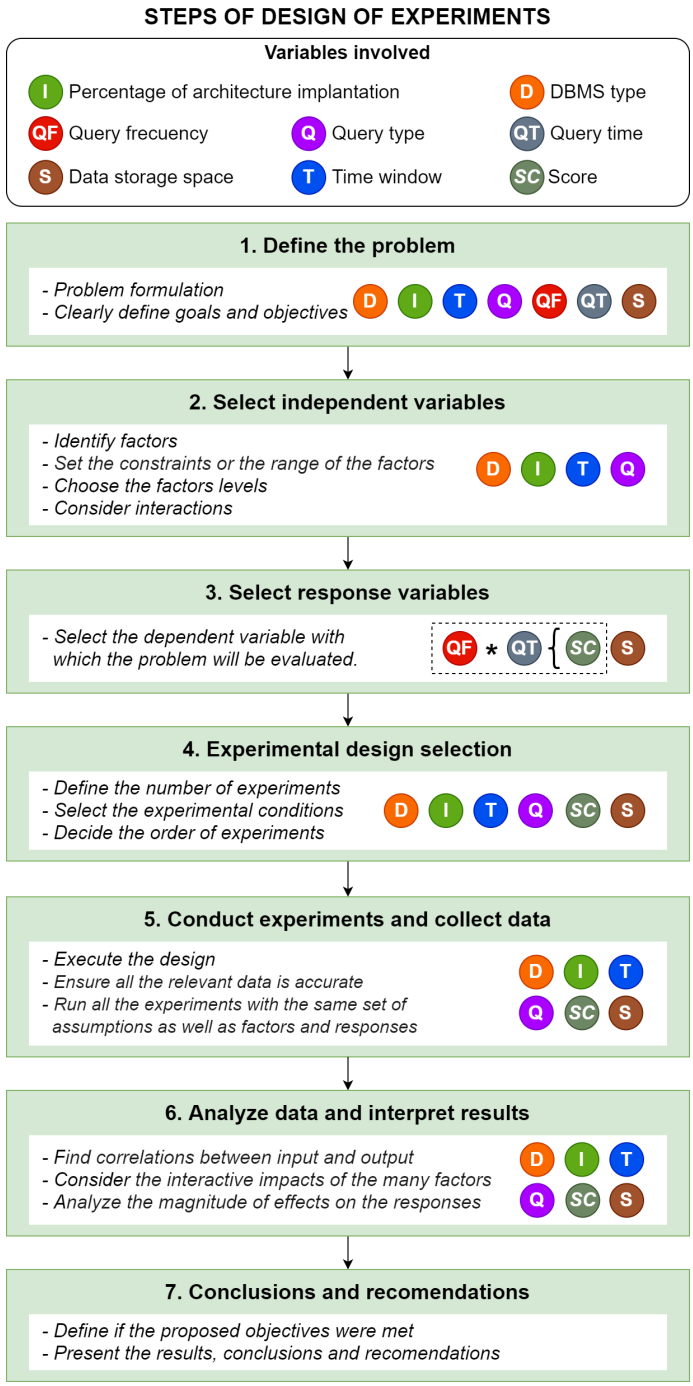


FIGURE 5.13: Steps of DoE including the experiment variables involved

5.6.1 Define the Problem

The first step consists in the recognition and *definition of the problem*; specifying the objectives and goals. For the particular scenario concerning this work (see Section 5.3.1), the problem consists in selecting an adequate implantation of the proposed architecture that allows to optimize the use of data storage space without drastically penalizing the query execution time. To conduct the analysis, the following variables have been considered: the percentage of implantation of the architecture, the type of DBMS, the types of queries, the time-window used in each type of query, the frequency with which each type of query is executed, the response time of each type of query, and the data storage space.

5.6.2 Select Independent Variables

Among those variables presented in the previous step, those that can influence on the objective of the problem described above must be taken into account for the selection of the independent variables or factors. The selected factors are: the percentage of implantation of the architecture, with five levels: {0%, 25%, 50%, 75%, 100%} (concerning the percentage of the data stored in a reduced representation on the third level of the architecture, respectively); the type of DBMS, with four levels: {InfluxDB, Cassandra, MongoDB and Neo4j}; the time-window of each type of query, with three levels: {day, week and month}; and the types of queries mentioned in Section 5.3.2, with four levels (one for each type of query).

5.6.3 Select Response Variables

Apart from those variables mentioned above, there are the response variables with which the problem will be evaluated. Considering that the problem defined in Section 5.6.1 is focused on the selection of an adequate implantation of the proposed architecture that allows to optimize the use of data storage space, without drastically penalizing the query execution times; there are two response variables to take into account: the *data storage space* and the *execution time of the queries*.

5.6.4 Experimental Design Selection

Once the independent and response variables have been selected, an experimental design must be chosen to conduct the experiment. Factorial designs are defined in [Tan08] as one of the most efficient tools for analyzing the effect produced by two or more factors with the aggravating factor that the number of experiments to be carried out increases exponentially with the number of factors (and their levels). However, taking into account that the number of factors in this experiment is reduced, the *full factorial design* has been selected. Once the design has been chosen, the number of experiments to be performed has been calculated taking into account that there were four factors with levels that vary between three and five. This

way, 240 experiments have been performed on which the *query execution time* and the *data storage space* have been measured.

$$\begin{aligned} Experiments &= L_{Implantation} \times L_{DBMS} \times L_{TimeWindow} \times L_{QueryTypes} \\ &= 5 \times 4 \times 3 \times 4 = 240 \end{aligned} \tag{5.3}$$

5.6.5 Conduct Experiments and Collect Data

The results of the conducted experiments, mentioned in the previous section, have been gathered and adapted according to the specifications of the chosen experimental design. In particular, the data from Table 5.6 were used to obtain the total query time for percentages of implantation of the architecture of 0% (Raw Series) and 100% (Reduced Series). In addition, further experiments were performed to obtain the total query time for percentages of implantation of the architecture of 25%, 50% and 75%. The data obtained as result from the performed experiments can be seen in the appendices Table B.1, Table B.2 and Table B.3 respectively. Moreover, the data storage space for the different percentages of implantation of the architecture were obtained from Table 5.4. The data obtained as result from the experiments was gathered in a table (See the Appendix C) with which the analysis presented in the following section were performed.

5.6.6 Analyze data and Interpret Results

The results of the conducted experiments have been analyzed and interpreted by using the statistical program *Minitab 19* [Min20]. In particular, two different types of analysis have been performed: on the one hand, in order to test the significance of the effects of different factors on the response variables, first, a *Pareto Diagram* has been done to visualize the magnitude of the effects; and then, an analysis of the variance (ANOVA) has been performed to formalize the test of significance of the effects for each response variable respectively. On the other hand, a response optimization has been performed in order to determine which is the recommended architecture for the considered scenario. Next subsections show the results of the performed analysis.

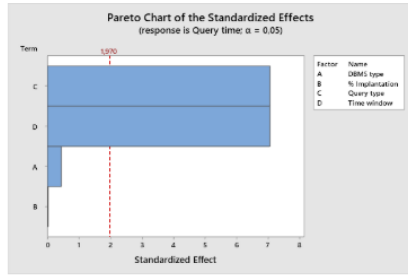
5.6.6.1 Analysis of the Significance of the Effects of the Different Factors

It can be observed that, both in the analysis of variance and in the Pareto chart (Figure 5.14), the *percentage of implantation of the architecture* and the *DBMS type* are not statistically significant (p=0.981 and p=0.661, respectively) with respect to the *query time*. This indicates that regardless of the percentage of implantation of the architecture and the type of DBMS that it is used, the query time will not be significantly penalized. Furthermore, the analysis of variance and the Pareto chart (Figure 5.15) for the *data storage space* show that the *DBMS type* and the *percentage*

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	12	1,14198E+11	9516475697	33,38	0,000
Linear	12	1,14198E+11	9516475697	33,38	0,000
DBMS type	3	455077588	151692529	0,53	0,661
% Implantation	4	118184966	29546241	0,10	0,981
Query type	3	67088037017	22362679006	78,44	0,000
Time window	2	46536408798	23268204399	81,62	0,000
Error	227	64715442628	285090056		
Total	239	1,78913E+11			

ANOVA



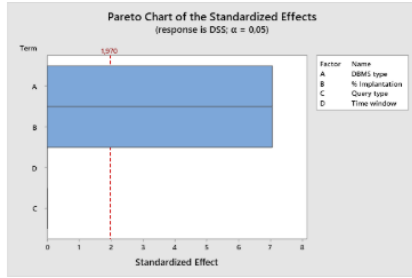
Pareto Chart

FIGURE 5.14: Analysis of the significance of the effects of the Query Time

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	12	48178161	4014847	48,88	0,000
Linear	12	48178161	4014847	48,88	0,000
DBMS type	3	37772758	12590919	153,28	0,000
% Implantation	4	10405403	2601351	31,67	0,000
Query type	3	0	0	0,00	1,000
Time window	2	0	0	0,00	1,000
Error	227	18646333	82142		
Total	239	66824494			

ANOVA



Pareto Chart

FIGURE 5.15: Analysis of the significance of the effects of the Data Storage Space

of implantation of the architecture are statistically significant ($p = 0.000$), greatly influencing storage space, while *Query type* and *Time window* are not ($p = 1{,}000$).

5.6.6.2 Response Optimization to Determine the Most Suitable Architecture for the Considered Scenario

After analyzing the influence of the different factors on the *query response time* and the *data storage space*, a response optimization has been performed in order to determine which is the architecture recommended for the considered scenario. However, it is worth mentioning, that in real world scenarios, not all the queries are executed with the same frequency, and thus, in order to select the most suitable architecture for the considered scenario, the frequency on which each query is executed should also be taken into account.

The frequency with which each type of query has been executed (in percentage of the total queries), is reflected in Table 5.8 (this table has been designed according to the experience provided by the domain experts from the real-world manufacturing scenario). Moreover, the distribution of those frequencies (in percentage of the total queries of each type) over the different time-windows considered in this work is presented. Lastly, the frequency with which each type of query is executed over each time-window has been computed. For example, *Query Type 1* is executed in

a 60% of the times; of which in a 70% of the times is executed over a time-window of one day. Therefore, *Query Type 1* is executed in a 42% of the times to retrieve a day's worth of data (60% × 70%).

In order to include the queries execution frequency on the response optimization, a new output variable called *score* has been created, where the frequency calculated for each type of query, executed over each time-window, is applied to the query execution times obtained in the conducted experiments. The application of those frequencies allows to weigh the response times of the queries, giving greater importance to those queries that are executed more frequently and lower importance to those executed more rarely.

TABLE 5.8: Query execution frequency (percentages)

Q. Type (%)		**T. Window** (%)		**Frequency (%)**
1	(60)	Day	(70)	42
		Week	(20)	12
		Month	(10)	6
2	(15)	Day	(70)	10.5
		Week	(20)	3
		Month	(10)	1.5
3	(12.5)	Day	(70)	8.75
		Week	(20)	2.5
		Month	(10)	1.25
4	(12.5)	Day	(70)	8.75
		Week	(20)	2.5
		Month	(10)	1.25
				Total: 100

Note: *Q. Type* and *T. Window* refers to *Query type* and *Time-window* respectively.

Finally, in order to select the most suitable architecture for the considered scenario, the response optimization option available in *Minitab 19* has been used, setting the goal to minimize the *score* and *data storage space*. However, it is worth mentioning that for the response optimization, the type of query and time-window factors have been leaved aside, in order to obtain a recommendation of a percentage of implantation of the architecture with a concrete DBMS but not for a particular type of query and time-window (otherwise these factors will be included in the response, leading to an specific optimization for a type of query executed over a particular time-window). As a result, the use of InfluxDB and a 75% of architecture implantation (25% of the data in raw format on the second level of the architecture and 75% of the data in a reduced representation on the third level) is suggested (see Figure 5.16), which seems a reasonable response considering: on the one hand, that the proposed architecture allows to decrease the required data storage resources without hampering the suitability of the DBMSs for the management of industrial time-series data; and on the other hand, that InfluxDB is a database optimized for time series management.

Solution

Solution	DBMS type	% Implantation	DSS Fit	SCORE Fit	Composite Desirability
1	InfluxDB	75	-116,998	278,968	0,902639

FIGURE 5.16: Solution from Minitab 19 for the response optimizer

5.6.7 Conclusions and Recommendations

Once the data analysis obtained from the DoE methodology is concluded, it can be verified that a greater implantation of the proposed architecture significantly reduces the data storage space, minimally affecting the query times, thus demonstrating the effectiveness of the architecture maintaining consistency with the results obtained in the previous sections.

5.7 Conclusions

This chapter presents an architecture that allows to relieve the problem of the considerable costs associated with the cloud storage resources required to store the high volume of time-series data generated during the manufacturing processes in Industry 4.0 scenarios. The architecture follows a *multi-temperature* data management paradigm on which the temperature tiers hot, warm, and cold are considered; and thereby, it is materialized as a three-level hierarchical architecture. The main novelty of the proposed architecture relies on the third level of the architecture, where a reduced representation of the time series (obtained by applying time series reduction techniques) is stored to reduce the required data storage resources. The architecture has been implemented by using four different types of database engines (InfluxDB, Cassandra, MongoDB and Neo4J), and its performance has been tested and contrasted on each database engine under the perspective of two main different dimensions: used storage space (and its associated costs) and total query time.

Regarding storage space, the performance results also show that the proposed architecture allows to optimize cloud storage resources (and its associated costs) without hampering the suitability of the DBMSs for the management of industrial time-series data. The achieved reduction in terms of data storage space is remarkable, at least a 78% even when dealing with a specialized time series database engine. Conversely, the compromised information due to the use of lossy reduction techniques is lower than a 5% in terms of RMSE (acceptable in the considered scenario).

With respect to the data access time, from the performed tests it can be observed that it varies depending on the type of query and the time-window. For simple queries that require very simple processing, the access to the reduced representation of the data and its reconstruction is faster than the access to the raw data as the time-window increases, while for queries requiring some processing over the data (e.g., aggregates or searching values, etc.) the access to the raw data is faster due to the required

time to reconstruct and process the reduced representation of the data. Nevertheless, the introduced latency for the first three types of considered queries are assumable (a small latency of 30 seconds when dealing with a month's worth of data).

From the performed tests it can be observed that, although all the database engines have been successfully used for the management of big volumes of time-series data, InfluxDB outperforms the other systems (as was expected for being a specific database engine for time series management). Moreover, it is worth mentioning that the performance results obtained from the tests and the conducted experiment applying the DoE methodology for industrial environments could give clues, during the execution of the task of selecting and developing the adequate architecture for a considered scenario, to those who are in charge of managing big volumes of manufacturing data.

Finally, it should be mentioned that part of this chapter has already been published in the journal paper entitled *"A Three-Level Hierarchical Architecture for an Efficient Storage of Industry 4.0 Data"* in the *Computers in Industry Journal* [VRD+].

Chapter 6

A Flexible Alarm Prediction System Following a Forecaster-Analyzer Approach

There is an increasing interest among manufacturers in exploiting the potential of the captured data during the manufacturing process to boost data-driven applications for different purposes, such as the control of product quality, the predictive maintenance of equipment, fault detection, etc. Among the different applications that can be implemented within the context of Smart Manufacturing, one of the most promising ones is the predictive maintenance of equipment as it as directly affects the service life of equipment and its production efficiency [WTL+17]. Consequently, several analysis methods are appearing to address a proactive maintenance of the equipment; among which, many of them manage different types of alarm systems to control the production process, and warn the operators in the plant about situations that could hamper the machine operation or incur in stops in the production process.

Overall, those alarm systems play a prominent role in maintaining plant safety and operation efficiency of modern industrial plants, by keeping the processes with normal operating ranges [WYCS16]. However, sometimes, in those systems the activation of the alarms is so close to the issue that there is no action-margin for the operators to manage the situation [LQPH13]. But, if the activation of the alarms is predicted early enough, the settings of the machine could be reconfigured in order to avoid stops in the production process or hampering the machine [WYCS16], [LABTS14]. In fact, the design of mechanisms to generate predictive alarms in order to forecast upcoming critical abnormal events has been stated as one of the open research problems in alarm systems [WYCS16], as it affects directly to the Overall Equipment Efficiency (OEE) = Availability (A) * Performance (P) * Quality (Q)). For example, in the real-world context of this

work, presented in Chapter 3, Section 3.1, an early prediction of the *Plastic Temperature not Reached in the Die Entry Alarm* could lead to avoid bad quality (Q) products, by increasing the resistors' temperature, and a *Molten Resistor or Broken Thermocouple Cable in Die Zone 2 Alarm* could allow the operators in the plant to perform a proactive maintenance of the equipment to avoid possible damages in the machine or its components, by turning on the fans that cool down the resistors (A & P).

In this regard, this chapter presents the third contribution of this research work, that resides in the design and development of *a flexible alarm prediction system for Smart Manufacturing scenarios following a forecaster-analyzer approach.* The system follows a two-stage approach, on which first, a LSTM neural networks-based forecaster predicts the future sensor's measurements and then, distinct analyzers based on Residual Neural Networks determine whether the predicted measurements will trigger an alarm or not. The system supports some features that make it particularly suitable for Smart Manufacturing scenarios: on the one hand, the forecaster is able to predict the future measurements of different types of time-series data captured by various sensors in non-stationary environments with dynamically changing processes. On the other hand, the analyzers are able to detect alarms that can be modeled with simple rules based on the activation condition, and also more complex alarms on which it is unknown when the activation condition will be fulfilled. Moreover, the followed approach for building the system makes it flexible and extensible for further predictive analysis tasks.

The chapter follows with an analysis of the state of the art in alarm prediction systems and an overview of the use of neural networks for predictive tasks in manufacturing. Then, continues describing the approach followed in this research work to design and build the alarm prediction system and a detailed description of the main components involved in it (i.e., the forecaster and the analyzers). Afterwards, it presents how these components are combined for predicting alarms, and finally, the conclusions of the realized work are presented.

6.1 Analysis of Related Work

The increasing interest among manufacturers in exploiting the potential of large volumes of manufacturing data for diverse purposes has led to the introduction of artificial intelligence techniques in these scenarios, to conduct different types of analysis over the captured data. In this regard, the automatic feature learning and high-volume modelling capabilities of deep learning, provide advanced analytics tools [WMZ+18], for this kind of scenarios managing huge volumes of data (which are essential for approaches that use artificial intelligence techniques [LOD18]). This, together with the increasing popularity of deep learning, has promoted the application of deep learning techniques to manufacturing data and many researchers are advocating for their use to boost data-driven applications in Smart Manufacturing scenarios [WMZ+18].

One of the most interesting applications of deep learning techniques to manufacturing data is the predictive maintenance of equipment, since it directly affects the service life of equipment and its production efficiency [WTL+17]. Thus, different methods are appearing to address a proactive maintenance of the equipment; for example, in [ZZL19], a data-driven bearing performance degradation assessment method based on LSTM neural networks is proposed; in [WDH18], an approach for fault prognosis with the degradation sequence of equipment based on LSTM neural networks is proposed; and in [MTR+16], a LSTM Encoder-Decoder model is used for multi-sensor prognostics using an unsupervised health index.

Besides those methods, different types of alarm-systems [WYCS16] have been also used in order to conduct a predictive maintenance of equipment. These systems control the production process and warn the operators in the plant about situations that could hamper the machine operation or incur in stops in the production process [WTL+17]. However, sometimes, in these systems the activation of the alarms is so close to the issue that there is no action-margin for the operators to perform a proactive maintenance of the equipment [LQPH13]. In such cases, an early prediction of the alarms' activation, will grant an extra time for the reconfiguration of the settings, controlling the production process in order to avoid production stops or hampering the machine. Therefore, the design of early alarm prediction systems has been stated as one of the open research problems in alarm systems [WYCS16].

Different proposals have attempted to predict alarm activations in Smart Manufacturing scenarios with different approaches. For example, in [ZWL+16], a dynamic alarm prediction algorithm is applied to an industrial case study to predict critical alarms by using a probabilistic model based on a *n-gram* model and sequences of previous alarm activations. In [LABTS14], an alarm prediction system has been built by using autoregressive Least Squares Support Vector Machines (LS-SVM) models, to predict the activation of a temperature alarm associated to the bearings of a steel production machine, and in [LQPH13], a customized SVM model has been built for alarm prediction in a large-scale railroad network. Finally, in [CPZH19], an alarm prediction method based on word embedding and LSTM neural networks is presented to predict the next alarm in a process setting. The system presented in this work follows a *forecaster-analyzer* approach that combines LSTM neural networks to forecast the future measurements of various sensors, with Residual Neural Networks to analyze (or classify) the alarms in the predicted values.

Regarding the followed approach; in [LABTS14], the captured data by the sensors are forecasted and used to predict alarm activations following a similar approach to the *forecaster-analyzer* proposed in this work. Nevertheless, there are significant differences between both works: on the one hand, in [LABTS14], an autoregressive LS-SVM model is used to predict a unique sensor's future measurements, while in this work, a LSTM neural networks based model has been used to predict multivariate time series captured by multiple sensors. The use of a multivariate time series forecaster avoids having a specific model for each sensor, and also allows

to capture interdependencies between different time series for predicting alarms on which various sensors could be involved. On the other hand, in [LABTS14], the used analyzer is based on a rule on which an alarm is predicted if in the forecasted temperature values, the maximum temperature is reached at least once, while the system proposed in this work uses Residual Neural Networks-based (ResNet) classifiers. The use of these classifiers leads to more general purpose analyzers that are able to detect different alarms, including alarms which can be detected thanks to a rule based on the activation condition but also more complex alarms that cannot be detected by this kind of rules.

Two main aspects distinguish the proposed system from those mentioned before. Firstly, the use of a LSTM based forecaster to predict multivariate IIoT devices time-series data in a real Smart Manufacturing scenario with dynamically changing processes (the system presented in [CPZH19] also uses LSTM neural networks for alarm prediction; however, that system predicts the activation of the alarms by using previous alarm activation data instead of the time-series data captured by the sensors); and secondly, the use of deep learning-based analyzers that are able to predict those kind of alarms on which the activation condition could be modeled by a rule (as the system presented in [LABTS14] does), but also more complex alarms that cannot be modeled with rules based on the activation condition. Moreover, it has shown the possibility of adapting the used analyzers to deal with unseen situations which can emerge unexpectedly.

Concerning the neural networks used to build the alarm prediction system, it can be seen in the literature that, on the one hand, LSTM neural networks have been already successfully used for forecasting time series of sensor data. For example, in [HLA+15], LSTM neural networks are used for forecasting time-series data coming from different sensors monitoring environment variables in a farm-monitoring context, and in [ZGL+18], LSTM neural networks are used for forecasting the time-series data from 33 sensors of a cooling pump in a power station. On the other hand, neural networks have also been already used for time series classification purposes. For example, in [WYO17], 44 different time series databases of different nature are used to compare the performance of nine time series classifiers including three deep learning classifiers based on neural networks. Furthermore, [IFFW+19] extends the benchmark to 85 time series databases including also multivariate time series databases to compare nine deep learning classifiers based on neural networks, on which the ResNet classifier (Residual Neural Networks), achieves the best performance. However, although the mentioned deep learning-based models have been independently used in Smart Manufacturing scenarios, to the best of the authors' of this work knowledge, these systems have not been already combined for predictive maintenance tasks in these scenarios.

6.2 Overview of the Alarm Prediction System

Withing the real-world context of this research work, among the different smart services that the CEM aims to provide to its customers as part of its data-driven servitization strategy, one of the most prominent ones is the predictive maintenance of equipment. The production process of this CEM's customers is mainly based on a blown-extrusion process (see Section 3.1.2 in Chapter 3); a continuous process in which the machines uptime is critical, since long periods of downtime considerably decrease production. In this regard, a predictive maintenance of equipment could play a crucial role in promoting machine uptime, as well as it could help decreasing the operating costs of equipment [OLBO15a].

With the aim of proposing useful contributions that help to conduct a predictive maintenance of equipment, in this work an alarm prediction system is presented. This section provides details about the main elements involved in the setting of the system; in particular, details about the main features of the captured time series and alarm data, the accomplished tasks for pre-processing the data, and the followed approach for implementing the system are given.

6.2.1 Time-Series and Alarm Data

As part of its data-driven servitization strategy, the CEM has installed several sensors in the machines that it manufactures. Figure 6.1 shows the scheme of an extruder machine on which the CEM has implanted several sensors. Those sensors register time-series data with a continuous measurement at 1Hz frequency (i.e., one measurement per second) of a variety of equipment setting parameters and physical magnitudes (temperatures, pressures, etc.) related to the raw materials, production processes and industrial equipment from a plastic bottles production plant based on an extrusion process. Moreover, with the aim of providing an enhanced equipment maintenance, the CEM has implanted an alarm system to control the production process, on which it has defined some alarms that are triggered under different conditions established over the measurements taken by the sensors. Those alarms can allow the operators in the plant to conduct a proactive management of the different controls in the machine for a predictive maintenance of the equipment. Next, the considered alarms when building the alarm prediction system presented in this work are described:

- *Plastic Temperature not Reached in the Die Entry:* This alarm is associated to the thermal sensor implanted in the entry of the die, to measure the *melting temperature* of the plastic. This temperature directly affects the viscosity of the melted plastic that at the same time affects the quality of the final product. Thus, in order to avoid bad quality products and stops in the production process, a specific alarm has been defined to ensure the correct temperatures. This alarm is triggered if the *melting temperature* is lower than 170°C.

- *Incorrect Temperature:* This alarm is triggered if in any of the extruder zones the measured temperature is not correct. The correct temperature is established in the production plant and it is bounded between the established values ± an error margin.
- *Molten Resistor or Broken Thermocouple Cable in Die Zone 2:*[1] This alarm is triggered if in the second zone of the die, the heat resistor has been molten or if the thermocouple cable has been broken.

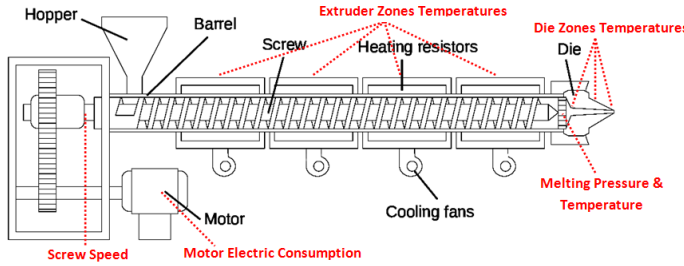


FIGURE 6.1: Different sensors (in red) implanted on an extruder machine

The data from the sensors implanted in an extruder machine of a real production plant and their associated alarm data (i.e., time series of alarm activation events registered in a log mode with a register per event) have been captured by using a REST API provided by the CEM. Table 6.1 shows the type of captured time series, the type of sensor and their associated alarms with their activation condition given by the domain experts from the CEM. When building the alarm prediction system, the captured data from the 01-12-2018 to the 28-02-2019 have been used to train and test the models, and the captured data from the 01-03-2019 to the 31-03-2019 have been used to evaluate the models. Figure 6.2 shows as an example a four-hour sub-sequence of the *Melting Temperature* time series on which an alarm (vertical red line) has been triggered.

TABLE 6.1: Properties of captured time-series data and the associated alarms

Time series	Sensor type	Associated Alarm	Activation Condition
Melting Temperature	Thermal (°C)	Plastic temperature not reached in the die entry	Melting Temperature < 170 °C
Extruder Temperatures Zones [1-4] (Extruder) Zone 5 (Union) Zone 6 (Filter) Zones [1-4] (Die)	Thermal (°C) (×10)	Incorrect temperature	Temperature > (set-temperature + error-margin) or Temperature < (set-temperature - error-margin) in any of the zones
Extruder Temperatures Zone 2 (Die)	Thermal (°C)	Molten resistor or broken thermocouple cable in die zone 2	The heat resistor in the second zone of the die is molten, or the thermocouple cable is broken

[1] Although this alarm type has been associated to all the resistors or thermocouple cables from the different zones of the extruder, only the one associated to the second zone of the die has been considered, because in the selected period of time is the only one that has been triggered.

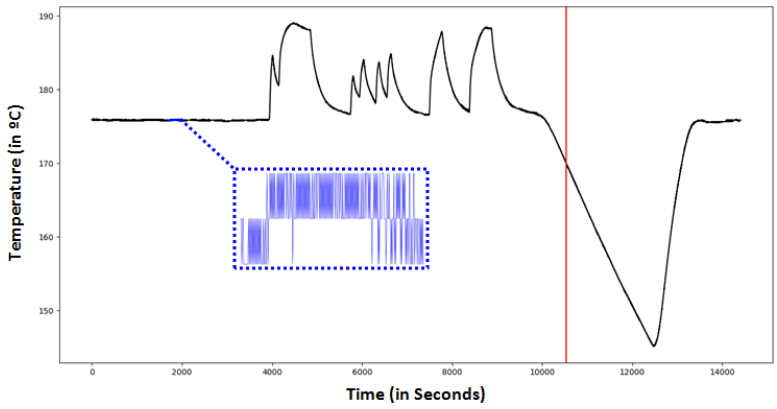


FIGURE 6.2: Sub-sequence of melting temperature time series on which an alarm has been activated

However, as already mentioned in Chapter 3, Section 3.2.1, before building the models, data must be correctly pre-processed in order to ensure data quality for accurate data analysis. In this sense, high-frequency sensors capturing data during long periods of time, leads to large-scale raw time-series data that hampers the performance of machine learning models, which usually scale poorly to high dimensional data [LKLC03]. Thus, in order to reduce the dimensionality of the data, the time series have been aggregated by minute, using the Piecewise Aggregate Approximation technique available in the system presented in Chapter 4. This aggregation also allows to reduce the complexity of the time series forecasting problem, as it reduces the number of steps to predict, and as can be seen in [LABTS14], the performance of the models for predicting future values decreases as the number of steps to predict increases. For example, without any aggregation, in order to build a model that predicts the measurements of the following 5 minutes, the model would need to predict 300 steps-ahead, while aggregating the data by minute it will only need to predict 5 steps-ahead.

Furthermore, the measurements of the implanted sensors present some inaccuracies (i.e., noise) due to the precision of the sensors which introduces an additional complexity into the ability of the models to predict the under-laying behaviour of the time series. Thus, in order to remove the noise that hampers the performance of the models, data has been filtered by using the Discrete Fourier Transform [AFS93]. Figure 6.3 shows the same time series presented in Figure 6.2 after aggregating the data by minute and removing the noise. Missing values and outliers have also been removed from the raw data by using the system presented in Chapter 4.

Finally, the pre-processed time-series data have been integrated in a dataset on which each timestamp is associated to the measurements of all the sensors (i.e., a dataset with a structure of ($timestamps \times num\_sensors$)). This dataset has been the one used to generate the input

data for the models. However, these data do not meet the specific necessities of the selected deep learning models, and thus, before building and training the models, first, data have been normalized in the range [-1, 1], and then, some transformations have been applied in order to meet the requirements of the models (see sections 6.3.1 and 6.4.1 respectively).

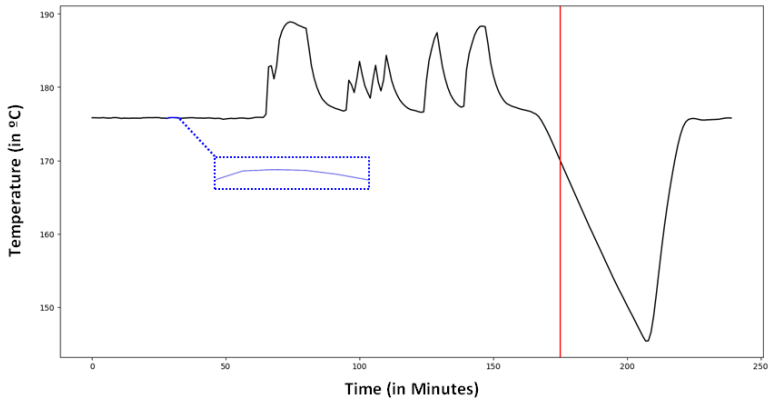


FIGURE 6.3: Sub-sequence of melting temperature time series on which an alarm has been activated (after pre-processing)

6.2.2 An Alarm Prediction System Following a Forecaster-Analyzer Approach

As mentioned before the alarm prediction system follows a two-stage *forecaster-analyzer* approach on which first, the future measurements of the sensors are forecasted; and then, different types of alarms are predicted by using three different analyzers (i.e., classifiers trained to detect interesting patterns matching alarm activations) over the forecasted data. Figure 6.4 shows an example of this approach by using the built forecaster for predicting the future values of the *Melting Temperatures* time series and an analyzer that tries to detect if the *Plastic temperature not reached in the die entry* alarm will be activated or not in the predicted values. In the *training phase*; first, the forecaster is built and trained by using time-series sub-sequences to predict the following values of the time series, and then, an analyzer is built and trained using those sub-sequences to determine if in a given sub-sequence an alarm will be activated or not. In the *deployment phase*; the future measurements of the sensor (time-series sub-sequences) are predicted by using the built forecaster and introduced into the corresponding analyzer that determines if an alarm will be triggered or not.

The prediction of those alarms, could allow the operators in the plant to reconfigure the settings of the machines in a proactive way in order to avoid bad quality products or hampering the machine. Furthermore, the followed approach is easily extensible, since the predicted data by the forecaster, could also be used to anticipate other kind of events by building

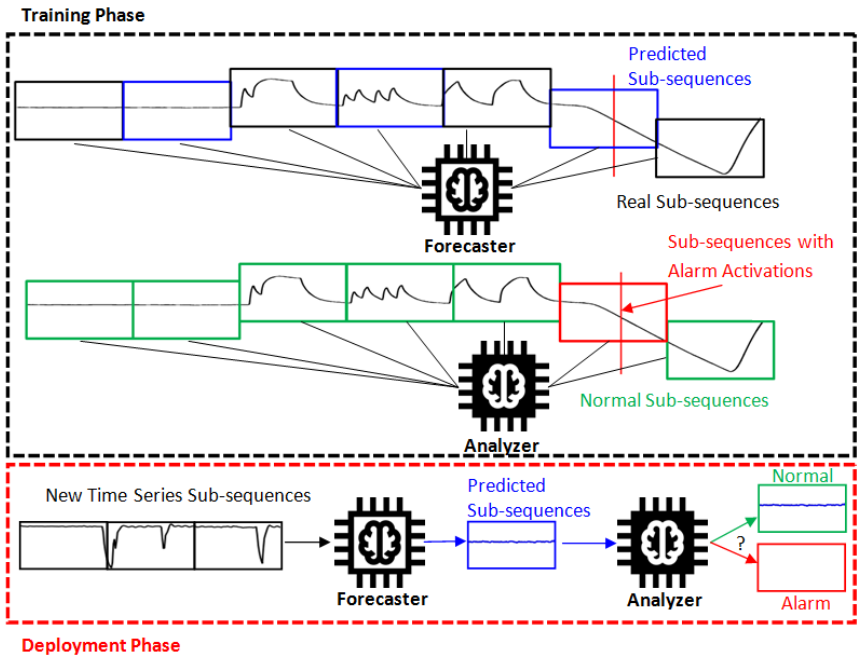


FIGURE 6.4: Alarm prediction system following a forecaster-analyzer approach

new analyzers (e.g., abnormal behaviours or faults in the machine or its components). Next sections, describe in detail the main elements involved in the alarm prediction system.

Finally, with regard to the models involved in the alarm prediction system, they have been built using Tensorflow [ABC+16] and Keras [C+15] libraries and they have been deployed using the Google AI Platform [Goo19a]. In particular the training and prediction jobs have been executed on a *n1-highcpu-16*[2] machine (as master node) with a *standard_p100*[3] GPU accelerator. Moreover, the Google AI Platform allows to build and train the models over different clusters of GPU workers, which can result particularly interesting for Smart Manufacturing scenarios dealing with big volumes of data.

6.3 Industrial Sensors Time-Series Data Forecasting

Time series forecasting is an important research topic in the domain of science and engineering, in which past observations of the data are collected and analyzed to develop a model that can predict future observations [KAV15]. Over the years, various forecasting models have been developed

[2]Machine types in Google Compute Engine: `https://cloud.google.com/compute/docs/machine-types`.

[3]GPU types in Google AI Platform: `https://cloud.google.com/ml-engine/docs/using-gpus`.

in the literature. In particular, for time series forecasting, Autoregressive Integrated Moving Average (ARIMA) models have been widely used, and more recently, Artificial Neural Networks (ANNs) [ZPH98].

In the literature, both approaches have been compared in different application domains with mixed results [ZPH98] (in some cases, ANN perform better than classic time series forecasting models, whereas in other cases, classical time series models make more accurate predictions or both show a similar behaviour), mainly due to the complex nature of real-world problems [Zha03]. However, in the particular context of sensor and IIoT devices time-series data forecasting, different works (such as [HLA+15] and [ZGL+18]) have shown that the inefficiency of classical time series models to capture long-term multivariate dependencies of the data coming from multiple devices of different nature [WMW+19], makes ANN-based models more suitable than classical models. In particular, Deep Neural Networks (DNN) of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have been widely used for time series forecasting tasks [WMW+19], [SVG+17], [WLZ+19].

In order to evaluate the behaviour of different types of models with data coming from a real manufacturing scenario, in this work, three different types of models have been built to predict the future measurements of the sensors mentioned in Table 6.1, in three different time horizons that are relevant for the considered scenario: a CNN model, a LSTM-RNN model and an ARIMA model. Firstly, in the following subsections, the steps followed to prepare the data for the prediction models are presented, and then, the built models together with their performance evaluation.

6.3.1 Forecasting Data Preparation

As mentioned before, the pre-processed data (see Section 6.2.1) did not meet the specific necessities of the selected deep learning models and thus, before data could be used in those models, they should be prepared according to the input/output specifications of the models. To prepare the data, a sliding-window approach [Sad18] has been followed to transform the pre-processed dataset into a dataset composed of time-series sub-sequences with the measurements of all the sensors (i.e., a dataset with a structure of (*num_sub-sequences* × *window-length* × *num_sensors*), where *window-length* = *input_sequence_length* + *output_sequence_length* (see sections 6.3.2.2 and 6.3.2.3)).

Moreover, the deep learning-based time series forecasting models use a time-series sub-sequence as input, to learn how to predict the future sub-sequences of measurements with a given time horizon (i.e., *output_sequence_length*). Thus, the first *input_sequence_length* steps (i.e., minutes) will serve as input for the forecasting models, and the following *output_sequence_length* steps as output (the target values to predict). Therefore, the dataset mentioned above has been split into two datasets, an input dataset with a structure of (*num_sub-sequences* × *input_sequence_length* × *num_sensors*), and an output dataset

with a structure of (*num_sub-sequences* × *output_sequence_length* × *num_sensors*).

The selection of the time horizons has been determined by two constraints: on the one hand, it is known from the R&D director of the CEM from the real-world scenario that the effects of adjusting some of the settings of the production process may not be noticed until a few minutes (up to 15 minutes) have elapsed. On the other hand, the performance of the models for predicting future values decreases as the number of steps to predict increases (as can be seen in [LABTS14]). Therefore, at each prediction, 5 steps (i.e., 5 minutes) are forecasted and then, those predictions are used to predict further time horizons (10 and 15 minutes) following the approach described in [TBAS12], that uses the predicted sub-sequence together with the input data to predict the next sub-sequence.

6.3.2 Time Series Forecasting Models

Different models have been built in order to test their performance in both univariate and multivariate time series forecasting. For each type of model, a univariate time series forecaster has been built by using the *Melting Temperatures Sensor Data* and a multivariate time series forecaster by using the data of all the sensors shown in Table 6.1. The built models have been trained and evaluated following the *rolling strategy* described in [SNN18] on which the model predicts the future measurements of the sensors using the last available measurements. This strategy has been applied over the prepared training and evaluation datasets. Next the built models are presented.

6.3.2.1 ARIMA Model

ARIMA is a linear regression-based forecasting approach that captures temporal structures in time-series data. The acronym ARIMA stands for *Autoregressive*[4] (AR) *Integrated*[5] (I) *Moving Average*[6] (MA) [SNN18] and captures the key components of the model. These three components are specified as parameters when building an ARIMA(p,d,q) model, where p is the lag order (i.e., the number of lag observations used in model training); d is the degree of differencing (i.e., the number of differencing items applied); and q is the order of moving average (i.e., the size of the moving average window). ARIMA models were initially conceived for univariate time series forecasting; however, some generalizations of these models have been developed to allow them involving multiple variables. Such is the case of Vector Autoregresive (VAR) [Lüt11] models that capture the linear inter-dependencies among multiple time series introduced as variables. In these models, each variable has a linear function explaining its evolution based on its own lagged values, the lagged values of the other

[4] A model that uses the dependent relationship between an observation and some number of lagged observations [Bro17].

[5] The differencing of observations in order to make the time series stationary [Bro17].

[6] A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations [Bro17].

variables in the model, and an error term. When building a VAR(p) model, although usually the only required parameter is the lag-order (p), the model requires all the variables to have the same order of integration; thus, before building the model the data has been differenced with a degree of one (d=*1*).

In this work, an ARIMA(4,1,0) model has been built for univariate time series forecasting, and a VAR(4) model has been built for multivariate time series forecasting. The selection of the parameters has been done with a grid search (considering the following parameter ranges: p=[1-10], d=[1-5], q=[0-10]), using the *auto_arima* function and the *RollingForecastCV* model selection function of the *pmdarima* package [Smi17]. For building the models, the approach followed in [SNN18] has been used, on which the model performs multi-step *out-of-sample* forecasting with *re-estimation* (i.e., each time the model is re-fitted to build the best estimation model).

6.3.2.2 CNN Model

Convolutional Neural Networks (CNN) [Kou16] are a specialized type of neural networks for processing data that has a known, grid-like topology (including time-series data) [SVG+17]. These networks, employ a mathematical operation called *convolution* between the input data and a filter or a kernel, usually alternated with pooling operations to generate a *feature map* that is finally connected to a fully-connected neural network that analyzes the features for classification and prediction tasks [ZLC+17] (see Figure 6.5). The impressive success arisen by CNNs in the domain of computer vision (powering tasks like image classification, object recognition, etc.) has led researches and practitioners to apply them in other domains such as time series classification [ZLC+17] and time series forecasting [WLZ+19].

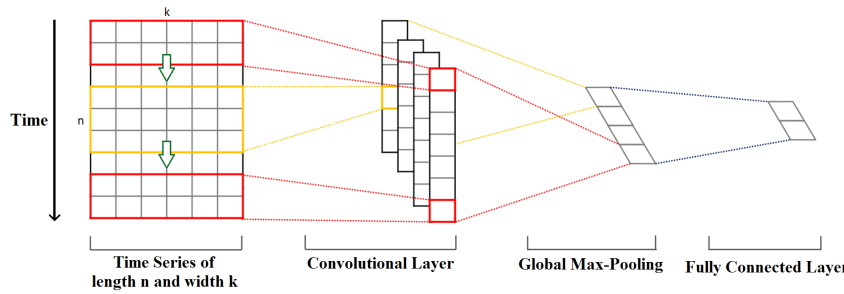


FIGURE 6.5: How 1D Convolutional Neural Networks work (extracted from [Kim14])

In this work, different CNN-based models with different parameter configurations have been built in order to select the most appropriate one for the considered scenario. These models are composed by blocks (up to

three) of a 1D convolutional layer and a *max-pooling* layer (takes the highest value from each area scanned by the CNN) followed by a flatten layer to reduce the feature maps to a one-dimensional vector and a fully-connected (dense) layer that interprets the features extracted by the convolutional part of the model to predict the future measurements of the sensors. For selecting the best parameter configuration, a grid search has been done by using the GPyOpt library[7] [GPy16] considering the parameter values shown in Table 6.2. Two constraints have been defined for the grid search: the first one, to ensure that the number of filters of a convolutional layer (in models with more than one layer) is the half of the precedent layer's number of filters; and the second one, to ensure that the kernel size in the subsequent layer is equal or lower than the precedent layer.

The built models with the different parameter configurations have been trained and evaluated five times and the best model, based on the obtained RMSE on the evaluation dataset, has been selected. Taking into account the results of the parameter optimization process, a CNN model has been built for univariate time series forecasting that uses a single convolutional block with 32 filters with a kernel size of 2 and the *ReLU* activation function, and a pool-size of 2 in the pooling layer. For multivariate time series forecasting, the model that achieved the best performance was a model with a single convolutional block with 64 filters with a kernel size of 8 and the *ReLU* activation function, and a pool-size of 2 in the pooling layer. The univariate and multivariate time series forecasting models have been trained using the *Adam* Optimizer with a learning rate of 0.002 and 0.001 (respectively), and the *mse* loss function during 400 and 300 epoch (respectively), with a *batch size* of 256 and an input sequence length 100 and 300 steps (respectively).

Table 6.2: Deep learning models' parameters[8]

Parameter Description	**CNN**	**LSTM**
Blocks of convolutional and max-pooling layers	1, 2, 3	-
Activation function of the convolutional layers	ReLU	-
N° of filters on each convolutional layer	64, 128, 256	-
Kernel size on each convolutional layer	2, 4, 6, 8	-
Pool size on each max-pooling layer	2, 3, 4	-
N° of hidden layers	-	1, 2, 3
N° of units (neurons) on each hidden layer	-	64, 128, 256
Input sequence length	100, 200, 300	100, 200, 300
Output sequence length	5	5
Loss function	mse	mse
Learning rate	0.001, 0.002, 0.005	0.001, 0.002, 0.005
N° of training epoch	100, 200, 300, 400	100, 200, 300, 400
Optimizer	adam, nadam	adam, nadam
Batch size	64, 128, 256	64, 128, 256

[7] A Bayesian Optimization tool for black-box functions that allows tuning automatically machine learning models' parameters.

[8] A hyphen (-) means that the parameter is not applicable for the model or that has not been considered.

6.3.2.3 LSTM Model

Long Short-Term Memory (LSTM) [HS97] is a special kind of Recurrent Neural Network (RNN) capable of learning order dependence in sequence prediction problems. A RNN is a type of Artificial Neural Network (ANN) that contains feedback loops allowing information to be stored within the network. RNNs can be seen as an extension of feedforward ANN that add recurrent connections feeding the hidden layers of the neural network back into themselves. The recurrent connections provide a RNN with information of not just the current data sample it has been provided, but also with information from the *hidden state* which remembers some information about previous data samples. A RNN with a feedback loop can be seen as multiple copies of an ANN, with the output of one serving as an input to the next (see Figure 6.6).

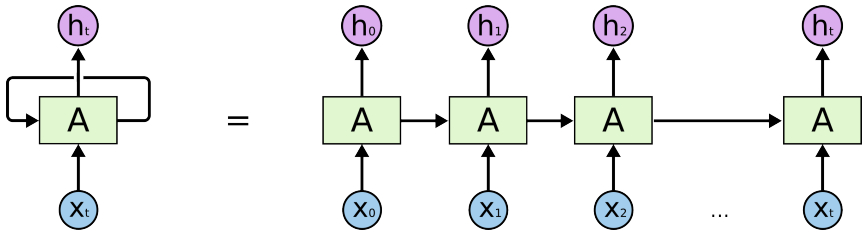


FIGURE 6.6: An unrolled Recurrent Neural Network (extracted from [Ola15])

RNNs unlike other ANNs, use their understanding from previous events to process an upcoming event rather than starting from scratch every time (e.g., using previous video frames might inform the understanding of the present frame). As a consequence, RNNs are particularly useful for sequence data processing in classification or regression tasks. Indeed, RNN are typically used to solve tasks related to time-series data [Nvi20]. Different architectures, such as *Bidirectional RNNs* or *Gated Recurrent Units* (GRUs) have been proposed for RNNs. However, Long Short Term Memory (LSTM) RNNs provide better performance compared to other RNN architectures by alleviating what it is known as the vanishing gradient problem [Hoc98]; and thus, they are the most widely used ones [Nvi20].

LSTM neural networks have the chain like structure composed by a set of cells, typical of RNNs, on which each cell contains a cell state that allows the information to be kept for a long period of time [YDJY17]. In LSTM neural networks the information added or removed from the cell state is carefully regulated by structures called *gates* (composed out of a *sigmoid* neural network layer and a point-wise multiplication operation). A LSTM neural network has three of these gates controlling the cell state (see Figure 6.7): A forget gate and an input gate that control which part of the information should be removed/reserved in the network; and an output gate that uses the processed information to generate the correct output [Ola15]. LSTM neural networks have been explicitly designed to avoid the long-term dependency problem present in other recurrent neural networks. Their ability to remember information for longer periods of

time allows them to perform well in diverse time series forecasting tasks for both, one-step-ahead forecasting [HLA+15], and multi-step-ahead forecasting [YDJY17].

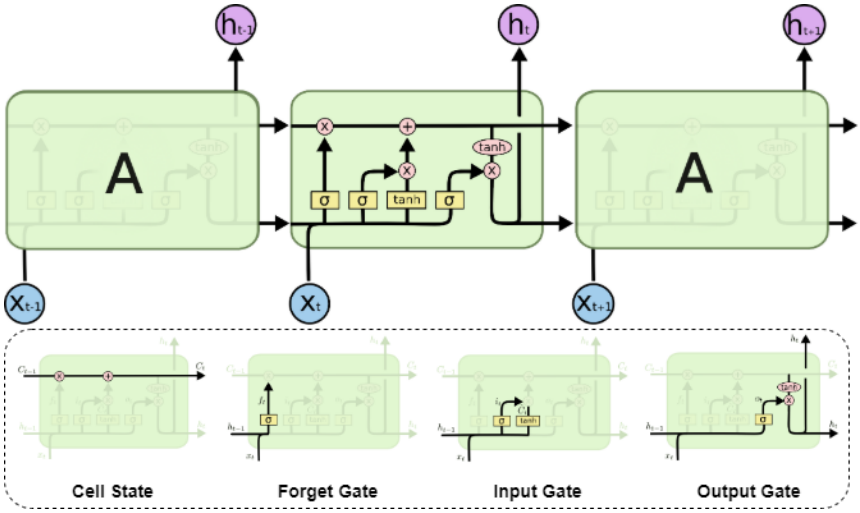


FIGURE 6.7: LSTM RNN Cell architecture (adapted from [Ola15])

In this work, different LSTM neural networks-based models with different parameter configurations have been built in order to select the most appropriate one for the considered scenario, following the same approach described in Section 6.3.2.2, Table 6.2 shows the considered parameter values for the optimization process. A constraint has been defined to ensure that the number of neurons of the subsequent layer (in models with more than one hidden layer) is the half of a precedent layer's number of neurons. Taking into account the results of the parameter optimization process, a *Vanilla LSTM* model has been built with a single layer and 128 neurons for both, univariate and multivariate time series forecasting. Both models have been trained by using the *Adam* optimizer with a learning rate of 0.001 and the *mse* loss function. Models have been trained during 300 and 400 epochs (respectively) with a *batch size* of 128 and an input sequence length of 300 steps.

6.3.3 Forecasting Models Evaluation

In order to select a suitable time series forecasting model, an instance of each of the models mentioned above (after selecting the best parameter configuration) has been built, and its performance has been evaluated. In general, to evaluate the performance of that type of models, a metric is often defined in terms of the forecasting error, which is the difference between the actual (desired) and the predicted values. Different metrics have been used in the literature to measure the performance of the predictions (a review of them can be found in [SBS+13], together with their formula). However, each of them presents different advantages and limitations, and

thus, there is not a universally accepted one by the forecasting academicians and practitioners [ZPH98]. Therefore, in this work three different error metrics have been selected to evaluate the performance of the forecasters: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE).

Table 6.3 summarizes the performance results of the different models considered for forecasting the whole time series for the three different time horizons and over the two available datasets (train and evaluation). Among all these results, those related with the evaluation dataset (unseen data for the model) have been considered in order to select the most suitable forecasting model. The performance results show that when considering the RMSE metric, the LSTM-based model outperforms the ARIMA and CNN-based models in both univariate and multivariate time series forecasting. When considering MAE and MAPE metrics, on the one hand, for univariate time series forecasting, ARIMA-based models outperform LSTM and CNN-based models. On the other hand, for multivariate time series forecasting (which is the most relevant case for the considered scenario) ARIMA and LSTM-based models show a similar performance and both outperform CNN-based models. However, for near time horizons, ARIMA-based models show a better performance, while as the time window to predict increases, their performance is degraded and LSTM-based models show a better performance.

Table 6.3: Time series forecasting evaluation results

			Dataset - Steps - ahead					
			Train			**Evaluation**		
	Metric	**Forecaster**	**5 min**	**10 min**	**15 min**	**5 min**	**10 min**	**15 min**
Univariate	RMSE	ARIMA	0.00028	0.00281	0.01048	0.01426	0.02399	0.03049
		CNN	0.01313	0.02648	0.06689	0.00503	0.01291	0.02346
		LSTM	0.00249	0.01132	0.02875	0.00137	0.00577	0.01560
	MAE	ARIMA	0.00005	0.00047	0.00167	0.00019	0.00064	0.00165
		CNN	0.00402	0.00860	0.01551	0.00196	0.00451	0.00764
		LSTM	0.00068	0.00202	0.00403	0.00037	0.00131	0.00291
	MAPE	ARIMA	0.02696	0.25799	0.78174	0.02350	0.11597	0.73543
		CNN	0.88629	1.94116	3.58219	0.42255	1.33615	1.76896
		LSTM	0.36965	0.80791	1.59330	0.07121	0.72974	1.74319
Multivariate	RMSE	VAR	0.00017	0.00195	0.00807	0.01444	0.02436	0.03180
		CNN	0.02581	0.03723	0.04972	0.01716	0.02119	0.02588
		LSTM	0.00757	0.01219	0.02036	0.00852	0.01215	0.01737
	MAE	VAR	0.00005	0.00051	0.00203	0.00300	0.00458	0.00628
		CNN	0.01297	0.01633	0.01936	0.00882	0.01096	0.01290
		LSTM	0.00444	0.00547	0.00683	0.00409	0.00498	0.00582
	MAPE	VAR	0.03581	0.27815	1.41278	0.54049	1.36444	2.14099
		CNN	4.61300	5.81008	6.53746	1.62916	2.11659	2.43444
		LSTM	1.30136	1.74324	2.59245	0.63228	0.87252	1.18475

In addition to the achieved performance results, regarding the applicability of the built forecasters in real Smart Manufacturing scenarios; on the one hand, the proposed system should be flexible enough to take into account the non-stationary nature of this environments with dynamically

changing industrial processes, that could hamper the performance of the built forecaster (e.g., changes in the machine operation mode, changes in the type of product to produce, etc.); on the other hand, the system should be suitable for making real time predictions in industrial contexts with big volumes of data produced by multiple sensors of different nature.

In this sense, it is worth mentioning that in order to make accurate predictions, the ARIMA models require to be re-estimated with the latest data before each prediction step. However, although this fact helps the model to make more accurate predictions (since it is always up to date with the newest data), it restricts the feasibility of its application to real-world problems in the context of Smart Manufacturing, where the latest data is not always available (e.g., due to stops in the production process), and where constantly re-estimating models for real time predictions could be computationally expensive. Conversely, LSTM and CNN-based models are not re-estimated before each prediction step, a property that could result unfavorable if the environment conditions change. Nevertheless, these models can be updated with new data due to a specific requirement of certain circumstances (e.g., one of the raw materials has been changed) or they could be periodically updated (e.g., daily) to keep the models up to date. Moreover as it is stated in [Ola15], LSTM neural networks have been explicitly designed to avoid the long-term dependency problem by remembering information for long periods of time, an interesting behaviour, especially when the model has been trained with large time series (since they could capture and remember different operation modes of the machine under different circumstances). Thus, taking into account the results of the performed tests as well as the system applicability in Smart Manufacturing scenarios, LSTM neural networks have been selected to build the forecaster of the proposed system.

6.3.4 LSTM Forecaster Performance Results

A time series forecaster has been built to predict the future measurements of the sensors, by using the selected LSTM-based model. The built forecaster takes sub-sequences of the time-series data captured by 11 sensors implanted on an extruder machine as input (see Table 6.1), and it predicts a 5-step-ahead sub-sequence for each sensor as output (i.e., 11 sub-sequences of 5 sensor measurements corresponding with the following 5 minutes). These predictions will serve as the output for the first time horizon (5 minutes), and also as the input to predict recursively the next two time horizons (10 and 15 minutes) (see Section 6.3.1).

Table 6.4 shows the performance results of the selected model when predicting the future measurements of each sensor individually. The performance results are shown with the RMSE, MAE and MAPE metrics. However, in the following, the RMSE metric is used for presenting the performance results of the selected forecaster, for being the one corresponding with the loss function used to build the model (RMSE = $\sqrt{MSE}$). Although there is some variation in the RMSE obtained when predicting the different sensors' data, the built forecaster achieves a great performance

with an average RMSE of 0.00852, 0.01215 and 0.01737 (respectively for each time horizon on the evaluation dataset).

Table 6.4: Forecasting performance results of each sensor for the evaluation dataset

		Steps - ahead		
Sensor	**Metric**	**5 min**	**10 min**	**15 min**
Melting Temperatures	RMSE	0.00564	0.01011	0.01556
	MAE	0.00412	0.00578	0.00617
	MAPE	0.56375	0.99342	2.40473
Extruder Temperature Zone 1	RMSE	0.01766	0.01819	0.02145
	MAE	0.00547	0.00579	0.00724
	MAPE	0.68267	0.72025	0.89809
Extruder Temperature Zone 2	RMSE	0.00955	0.01367	0.01809
	MAE	0.00221	0.00312	0.00456
	MAPE	0.40620	0.56167	0.84430
Extruder Temperature Zone 3	RMSE	0.01456	0.01866	0.02426
	MAE	0.00583	0.00665	0.00870
	MAPE	1.00995	1.12361	1.51917
Extruder Temperature Zone 4	RMSE	0.00831	0.01185	0.01788
	MAE	0.00213	0.00282	0.00415
	MAPE	0.40764	0.53567	0.72905
Extruder Temperature Zone 5 (Union)	RMSE	0.00550	0.00968	0.01525
	MAE	0.00388	0.00516	0.00552
	MAPE	0.49775	0.67376	0.81368
Extruder Temperature Zone 6 (Filter)	RMSE	0.00520	0.00950	0.01573
	MAE	0.00230	0.00262	0.00333
	MAPE	0.32786	0.41065	0.70757
Extruder Temperature Zone 1 (Die)	RMSE	0.00470	0.00844	0.01453
	MAE	0.00265	0.00296	0.00342
	MAPE	0.35323	0.49375	0.66072
Extruder Temperature Zone 2 (Die)	RMSE	0.00569	0.00967	0.01458
	MAE	0.00369	0.00505	0.00549
	MAPE	0.48412	0.67080	0.76792
Extruder Temperature Zone 3 (Die)	RMSE	0.00453	0.00831	0.01447
	MAE	0.00246	0.00308	0.00342
	MAPE	0.91350	1.85027	2.00073
Extruder Temperature Zone 4 (Die)	RMSE	0.01238	0.01559	0.01931
	MAE	0.01028	0.01178	0.01204
	MAPE	1.30838	1.56381	1.68630
Average	RMSE	0.00852	0.01215	0.01737
	MAE	0.00409	0.00498	0.00582
	MAPE	0.63228	0.87252	1.18475

Furthermore, if due to special requirements of the application scenario more precision is required for a particular sensor, a specific forecaster could be built to predict only the future measurements of that sensor in a more accurately way. Table 6.5 shows a comparison between the performance results of a specific forecaster for the *Melting Temperature sensor* and the multi-sensor forecaster; and Figure 6.8 shows, as an example, the predicted time series for the *Melting Temperature* sensor in a sub-sequence of the evaluation dataset (3600 minutes), when using the multi-sensor forecaster

(in green), and when using the specific forecaster (in red). Although the specific forecaster is more precise than the multi-sensor forecaster, the difference does not hamper the ability of the built analyzers to predict the alarms (see Section 6.5.1). Moreover, the multi-sensor forecaster allows to predict the data of various sensors at the same time, instead of building a specific forecaster for each sensor for those alarms associated to multiple sensors.

TABLE 6.5: Multi-sensor vs specific forecaster prediction error for Melting Temperatures (evaluation data)

Forecaster	Metric	Steps - ahead 5 min	10 min	15 min
Multi-sensor	RMSE	0.00564	0.01011	0.01556
	MAE	0.00412	0.00578	0.00617
	MAPE	0.56375	0.99342	2.40473
Melting Temperatures	RMSE	0.00137	0.00577	0.01560
	MAE	0.00037	0.00131	0.00291
	MAPE	0.07121	0.72974	1.74319

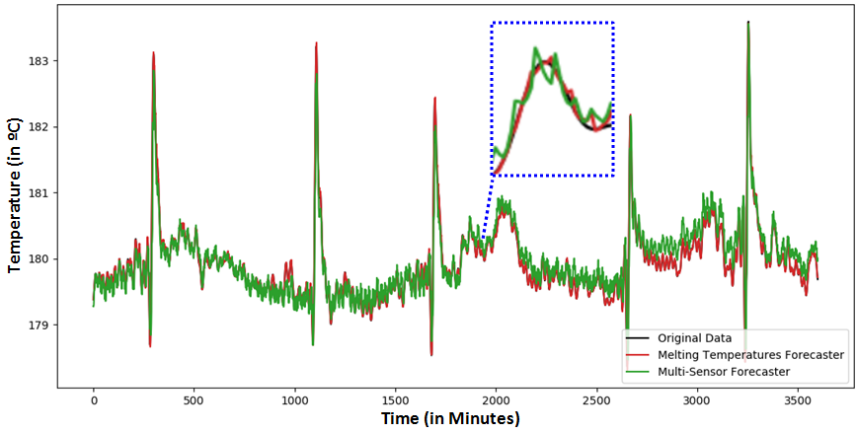


FIGURE 6.8: Melting Temperatures forecasting: predicted measurements vs original measurements (time horizon 5 minutes)

6.3.5 Dealing with Non-stationary Environments

As mentioned in Section 6.3.3, in non-stationary environments with dynamically changing processes, the data distribution can change over the time, yielding the phenomenon of *concept drift* [GvB+14]. In these environments, the common approach of training models in an offline manner using historical data could lead to models on which the performance decreases as time goes by and environment conditions change. Therefore, one desirable property of the models is their ability of incorporating new data. This ability to adapt to such *concept drift* can be seen as a natural

extension for incremental learning systems, such as the neural networks-based models used in this work, which learn predictive models example by example and thus, they can update the decision model as new examples arrive [GvB+14].

Two main approaches are used to adapt a decision model in order to address the concept drift [FGdCG16]: blind approaches that update the decision model periodically without verifying if changes really occurred; and informed approaches that modify the decision model when changes are detected. In this last approach, several methods have been proposed to detect the concept drift (such as the *Page-Hinkley* method or the *ADaptive sliding WINdow* (ADWIN)), a survey of them can be found in [GvB+14]. Most of those methods detect when the concept drift occurs and then the models can be updated independently. Another interesting approach is the one proposed in [SMT+18], where the model is slightly updated when new data is available using an anomaly-score computed between the predicted values and the original ones. This anomaly score ensures that a high anomalous score, during large periods of time, due to concept drift, leads to continuous changes in the network parameters, whereas short term anomalies or outliers producing a high anomaly score do not produce significant updates in the network parameters. However, as stated in Section 6.3.3, updating the model every time new data is available in Smart Manufacturing scenarios could be computationally inefficient, and although there exist some research works in Online Deep Learning Neural Networks that learns on the fly [SPLH17], using a traditional neural network-based model (like the ones used in this work) to change the model itself (every time new data is available) could leverage dangerous consequences (performance degradation) in long term. Thus, reaching a trade-off between both approaches could be more interesting.

In this work, a combination of both approaches is proposed to test the ability of the model to adapt to non-stationary environments. First, an anomaly-score has been used to detect when the model performance starts to degrade due to concept drift. For that, the RMSE between the predicted values and the real measurements is computed for every prediction made by the model as anomaly-score. If the mean of the obtained anomaly-score in a fixed period of time (i.e., the last hours) is higher than the obtained RMSE by the model when evaluated over a large period of normal operation mode (e.g., the evaluation dataset) plus an error margin, the model could be updated using a width enough period of time of the latest available data (e.g., the last day).

Figure 6.9 shows a comparison between the performance of the model to predict sensors' measurements after two different types of concept drift occurs (sudden concept drift in Figure 6.9-I, and incremental concept drift in Figure 6.9-II) and its performance after updating it with the latest available data (Figure 6.9-III and Figure 6.9-IV respectively). Moreover, the figure also shows a comparison between the obtained anomaly-scores when predicting the sensors' measurements, before and after updating the model, for each type of concept drift (sudden concept drift in Figure 6.9-V and incremental concept drift in Figure 6.9-VI). It can be observed that

at first, in both cases, the model performs well and its performance starts to degrade when the concept drift has occurred. However, once the concept drift has been detected and the model has been updated (the period compressed between the vertical gray dashed-lines), the performance of the model improves, whereas when the model has not been updated, it remains degraded.

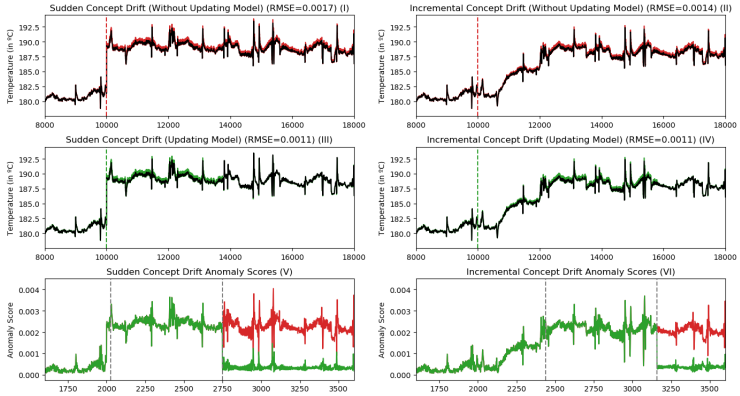


FIGURE 6.9: Time series forecasting with different types of concept drift and obtained anomaly-scores

6.4 Predictive Analysis of Industrial Sensor Time-Series Data

Smart Manufacturing leverages the tremendous advances in Big Data analytics to improve existing analysis capabilities and provide new ones, such as predictive analytics that analyze current and historical data to make predictions about future unknown events [MI17]. In this regard, the built forecaster allows to obtain future values of the sensors for a data-driven predictive analysis [MKLR19], by using models (i.e., analyzers) capable to detect events in the forecasted data. In this work, in particular, three different analyzers have been built for predicting three different types of alarms (see Table 6.1). Each analyzer is composed by a classifier that determines whether in a given time-series sub-sequence, an alarm will be triggered or not. More precisely, the analyzers are trained to detect negative and positive classes (regarding the activations of the alarms) in a binary classification problem. Thus, each analyzer is specifically used to detect a concrete type of alarm, which seems reasonable for the most critical alarms since specialized classifiers usually outperform general purpose ones.

However, these are not the unique alarms that can be produced in the considered scenario, and building a specific analyzer for each alarm that can be triggered could result tedious. Moreover, considering that all

the possible situations which could occur in non-stationary environments (such as Smart Manufacturing scenarios with dynamically changing industrial processes), are included in the training and testing datasets is too optimistic and could leverage to wrong predictions when dealing with unexpected circumstances. Taking into account that situation, the proposed system should be able to detect the alarms for which it has been trained, but also capable to deal properly with new conditions unseen when building it. Next sections show, first, the followed steps to prepare the data for the analyzers; then, the different analyzers built together with their performance evaluation, and finally, how the proposed system deals with new situations not considered *a priori* when building it.

6.4.1 Analyzers Data Preparation

For the classification data, the sliding window approach has been used to obtain different sub-sequences of the time series with a window-length of 100 (100 steps). In each sub-sequence, if an alarm has been triggered, the sub-sequence has been labelled as *True* and as *False* otherwise. This approach transforms the pre-processed dataset (see Section 6.2.1) into a dataset composed of time-series sub-sequences with measurements of each sensor and the sub-sequence label (i.e., a dataset with a structure of (*num_sub-sequences* $\times$ *window-length* $\times$ (*num_sensors* + *label*)). The sub-sequences of sensors measurements serve as input for the model (*num_sub-sequences* $\times$ *window-length* $\times$ *num_sensors*), whereas the label (one-hot encoded), serves as output for the model. Notice that while the number of steps predicted by the forecaster is between 5 and 15, a longer sub-sequence has been selected for the analyzers (100 steps), due to the requirements of the models. In order to make accurate predictions of the alarms, not only are necessary the last measurements of the sensors, but also historic information about the data. Thus, in the *deployment phase*, the predicted values by the forecaster for each time horizon are appended to the last observations of the data (e.g., for the time horizon of five minutes, the five predicted values by the forecaster are appended to the last 95 observations obtained by the sensor).

6.4.2 Analyzers

Three different analyzers have been built for predicting three different types of alarms (see Table 6.1). Each analyzer is composed by a classifier that determines if, in a given time-series sub-sequence, an alarm will be triggered or not. When building the analyzers, for simple alarms, such as the first two types of alarms presented in this work (*Plastic Temperature not Reached in the Die Entry Alarm* and *Incorrect Temperature Alarm*), a rule-based classifier based on the alarm activation condition (described in Table 1) could be used to predict correctly all the alarms. However, for more complex alarms, like the third alarm presented in this work (*Molten Resistor or Broken Thermocouple Cable in Die Zone 2 Alarm*), for which it is unknown when the activation condition will be fulfilled and therefore, cannot be modeled with rules, an automatic learning-based approach (such

as neural networks-based approaches) is necessary. In this work, three Residual Neural Networks-based classifiers have been built, one to predict this last type of alarms, and another two to predict the first two types of alarms (in order to test also the performance of these classifiers when predicting simple alarms).

Different neural networks-based classifiers have been widely used for different time series classification tasks. The selection of the most suitable classifier depends on the type of time-series data and the classification task itself. In order to select an appropriate classifier for an scenario with different types of time series, coming from multiple sensors of heterogeneous nature, and for different purposes (detect different types of alarms), the benchmarks presented in [WYO17] and [IFFW$^+$19] have been considered. Those benchmarks test up to nine distinct classifiers based on neural networks over 44 and 85 time-series databases (respectively) of different nature. In the benchmark presented in [WYO17] the classifier based on Fully Convolutional Networks (FCN classifier in [WYO17]) achieves the best performance for classifying the time series databases in the benchmark, followed by the Residual Networks [HZRS16] based classifier (ResNet classifier [WYO17]). Nevertheless, in the benchmark presented in [IFFW$^+$19] with an extended dataset, that includes a larger number of time series databases, the ResNet classifier achieves the best performance. For that reason the selected classifier for building the analyzers, has been the *ResNet* classifier proposed in [IFFW$^+$19]. Moreover, in those benchmarks, this classifier has been demonstrated to perform well on diverse time series datasets of different nature, an interesting property for Smart Manufacturing scenarios where multiple heterogeneous sensors produce different types of time series.

The architecture of the ResNet classifier (outlined in Figure 6.10) is composed by three residual blocks, followed by a global average pooling layer and a fully-connected layer with the *softmax* activation function. Each residual block is composed of three convolutions whose output is added to the residual block's input and then fed to the next layer. The number of filters for all convolutions in each block is fixed to 64, 128 and 128 respectively, with the *ReLU* activation function that is preceded by a batch normalization operation. In each residual block, the filter's length is set to 8, 5 and 3 respectively for the first, second and third convolution (see [IFFW$^+$19] for the concrete implementation).

In order to test the suitability of the ResNet classifier when it comes to alarm prediction, for each analyzer five different classifiers have been built by using different partitions of the training dataset (*5-fold cross-validation*). Each classifier has been trained with a batch size of 64 samples and during 400 epochs. The selected classifier has been the best one based on the loss function (*binary cross-entropy*) using the *Adam* optimizer with a learning rate of 0.001. To overcome the class imbalance, characteristic of alarm prediction and fault diagnosis scenarios, a balanced class weighting has been used during the training of the models. Then, the built *analyzers* are presented, and Section 6.4.3 shows their performance on predicting the three different types of alarms.

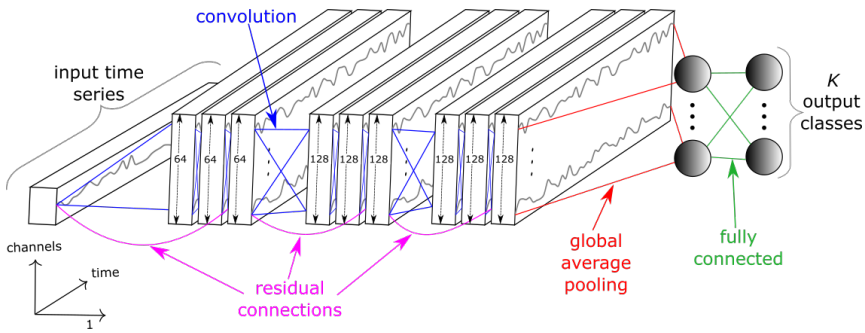


FIGURE 6.10: ResNet Residual Neural Network architecture (extracted from [IFFW+19])

6.4.2.1 Melting Temperatures Analyzer

The first analyzer has been built to predict the activation of the *Plastic Temperature not Reached in the Die Entry* alarm using the predicted measurements of the sensor that measures the plastics' *melting temperature*. This one is the simplest alarm to predict, since it can be modeled with a rule based on the activation condition which states that when the melting temperature is lower than 170 °C, the alarm is activated. In this case, it seems that the model manages to learn [Hin92] the activation condition of the alarm which can be reflected on the high performance achieved when predicting the alarms (an AUC-ROC value greater than 0.98%).

6.4.2.2 Incorrect Temperatures Analyzer

A second analyzer has been built to predict the *Incorrect Temperature* alarm. The prediction of this alarm is more complicated than the first one, because this alarm is triggered if in any of the extruder zones measuring the temperature of the plastic, the measured temperature is higher or lower than the established one, plus or minus an error margin (respectively). Nevertheless, the prediction of this alarm could be also modeled with a more complex rule that checks the activation condition for every temperature zone in the extruder. In this case, the analyzer also shows a good performance predicting the alarms, so that it seems to have learnt the under-laying condition of the activation of the alarm (an AUC-ROC value greater than 0.98%).

6.4.2.3 Die Zone 2 Analyzer

A third analyzer has been built to predict the *Molten Resistor or Broken Thermocouple Cable in Die Zone 2* alarm. The prediction of this alarm is even more complicated than the previous one, because although there is an activation condition described by the domain experts (see Table 6.1), it is unknown when the activation condition will be fulfilled and trigger the alarm, and thus, the detection of this alarm cannot be modeled using rules. Therefore, in this work a Residual Neural Networks-based classifier

has been used to build an analyzer that attempts to learn the underlying pattern of the sub-sequences on which the alarm has been triggered, so that if it sees a similar pattern in the predicted measurements, it will anticipate the activation of the alarm and the operators could stop the machine in a safe way or turn on the fans that cold down the resistor. This case reinforces the utility of using a more sophisticated classifier for predicting confidently the activation of this kind of alarms that cannot be modeled with simple rules (an AUC-ROC value greater than 0.93%).

6.4.3 Analyzers Performance Results

As mentioned in the previous section, for building the analyzers, Residual Neural Networks (ResNet) based classifiers have been used. In order to test the suitability of these classifiers for predicting the alarms, a 5-fold cross-validation process has been followed, on which the initial training dataset has been split into five new partitions of the data into train-test datasets. For each partition of the data, five classifiers have been trained in order to test their performance for predicting the different types of alarms (totaling 25 classifiers for each analyzer). The performance of each classifier, in order to discriminate normal operation sub-sequences from sub-sequences on which an alarm has been activated, has been evaluated by using the area under the Relative Operating Characteristic (ROC) curve (AUC-ROC) metric, a performance measurement for classification problems at various thresholds settings. This metric represents a degree or a measure of separability among the classes by telling how accurate is the model distinguishing among classes (the higher is the AUC-ROC, the better is the model at predicting normal sub-sequences as normal sub-sequences and sub-sequences with alarm activations as sub-sequences with alarm activations). Table 6.6 summarizes the AUC-ROC value obtained by each analyzer when predicting the alarms on the cross-validation process (see the average of the AUC-ROC value for all the built classifiers on the Cross-Validation column).

Once tested the suitability of the classifiers for predicting the alarms with the cross-validation process, five new classifiers have been built for each analyzer by using the whole training dataset. These classifiers have been evaluated using the evaluation dataset, and the average AUC-ROC of the five classifiers has been used to evaluate the performance of each analyzer (see the performance on the Validation column from Table 6.6). As Table 6.6 reflects, the more complicated is the prediction of an alarm, the lower is the performance of the analyzer. The first two analyzers manage to learn to discriminate normal operation sub-sequences from sub-sequences on which an alarm has been activated, and thus, they predict correctly almost all the samples. The third analyzer is the one that shows more difficulties to learn to difference between both types of sub-sequences. Nevertheless, it shows a great performance considering the difficulty of the type of alarm to predict.

Finally, among those classifiers, the one with the highest AUC-ROC (for each analyzer) has been the selected one for predicting the alarms in

the system. Figure 6.11 plots the ROC curve, showing the True Positive Rate (TPR) against the False Positive Rate (FPR) of the selected classifier for each analyzer, for the evaluation dataset (see also Table 6.8 in Section 6.5).

TABLE 6.6: Time series analyzers evaluation results

Analyzer	Cross-Validation		Validation	
	TRAIN	TEST	TRAIN	EVAL
Melting Temperatures	0.9978	0.9848	0.9998	0.9992
Incorrect Temperatures	0.9980	0.9817	1	0.9904
Die Zone 2	0.9849	0.9530	0.9464	0.9375

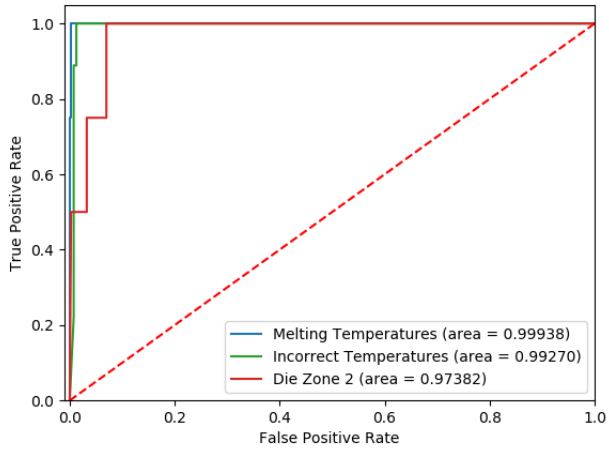


FIGURE 6.11: Area under ROC curve for each analyzer for the evaluation dataset

6.4.4 Dealing with Unknown Situations in Smart Manufacturing Scenarios

In the same way as traditional recognition and classification algorithms, the deep learning-based classifiers used in previous sections usually work under a common closed set (or static environment), where the training and testing data are drawn from the same label and feature spaces. However, this assumption is rather restrictive, given that in real world recognition and classification tasks it is usually hard to collect training samples representing all the possible situations. Therefore, a more realistic scenario is usually open and non-stationary, due to the fact that unseen situations can emerge unexpectedly (as occurs in Smart Manufacturing scenarios with dynamically changing processes), which could drastically weaken the robustness of these traditional methods [GHC18]. Taking into account

this aspect, in recent years, several approaches have appeared to deal with these situations including among others zero/one-shot learning and open set recognition. A categorization of them can be found in [GHC18], where the authors put particular interest in open set recognition for being able to deal with unknown situations, like those mentioned before.

Open set recognition [SdRRSB13] describes such a scenario where new classes (unseen during the training) could appear in the testing and requires the classifiers not only to classify the known classes accurately, but also, to deal effectively with unknown classes [GHC18]. Therefore, in open set recognition problems, classifiers usually consider a reject option that allows them to refuse to recognize an input sample due to its low confidence, avoiding to classify unknown samples as other classes. For example, in the context of this work, the analyzers manage to detect confidently the alarms for which they have been trained on, however, if the raw materials in the production process change requiring a higher/lower temperature to be melted, the *Melting Temperatures Analyzer* could detect an incorrect temperature in the new samples collected and trigger an alarm continuously. Detecting these situations would allow to update the models so that they can deal with the new situations while also avoiding incorrect classifications.

Open set recognition is a recent research topic to which researchers are devoting many efforts and therefore, some open set recognition algorithms have been developed to extend both traditional machine learning methods and deep neural networks-based models (a review of them can be found in [GHC18]). Regarding the deep neural networks-based models considered in this work, although they were not initially conceived for open set recognition problems, some works have already attempted to extend this models for open set recognition tasks [HC18], [BB16]. One of the most popular one is the method proposed in [BB16] where a new model layer, *OpenMax*, is introduced to estimate the probability of an input sample being from an unknown class.

OpenMax has already been used by other authors in the domain of computer vision [BB16] and natural language processing [SXL17] for image and text classification/recognition tasks (respectively). However, no reference has been found, where it is used in the context of industrial time-series data classification. In this work, a first attempt has been done to adapt a neural network for open set time series classification purposes by changing the *softmax* activation layer of the used *ResNet* classifiers to use the *openmax* layer as proposed in [BB16].[9] Nevertheless, it is worth mentioning that open set recognition is an open research topic that still faces serious challenges on which there is a lack of well-known frameworks and algorithms [GHC18].

As a proof of concept, the *softmax* activation layer of the *Melting Temperatures Analyzer* has been changed to use the *openmax* activation layer. The performance of the classifier with both activation layers has been

[9] Code and data for the research paper "Towards Open Set Deep Networks" [BB16] `https://github.com/abhijitbendale/OSDN` and an example of its implementation with Keras `https://github.com/aadeshnpn/OSDN`.

tested with sub-sequences of three different time series to predict whether an alarm will be activated or not in those sub-sequences. Concretely, the *Melting Temperatures* time series (those with which the classifier has been trained to detect the *Melting Temperature not Reached in the Die Entry Alarm* activations); the *Die Zone 4 Temperature* time series (which is a highly correlated time series to the *Melting Temperatures* time series since the sensors measuring the temperatures are placed close to each other in the extruder machine); and the *Melting Pressure* time series (which has been captured by a different nature sensor) have been considered for the test. Figure 6.12 shows a segment of the considered time series and two example sub-sequences: a normal operation mode sub-sequence (*s1*) and a sub-sequence on which an alarm has been activated (*s2*).

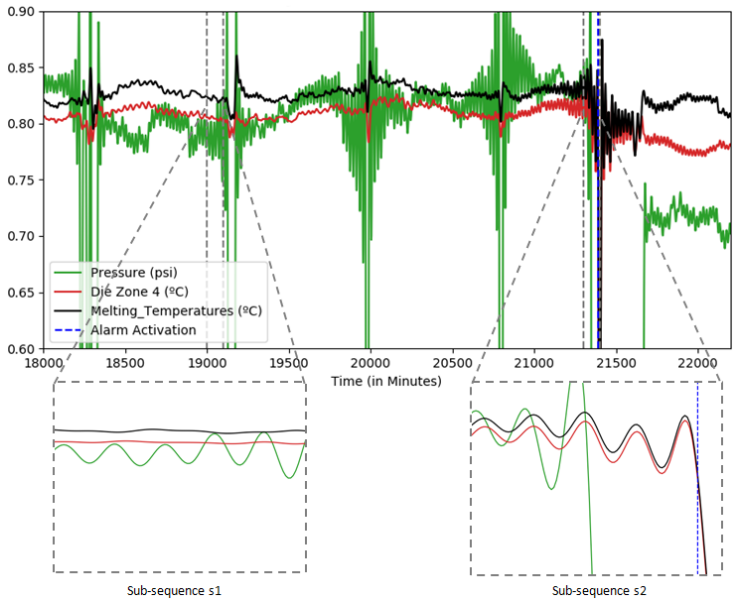


FIGURE 6.12: Example of open set recognition with different time series

The predictions made by the classifier with each activation layer for those sub-sequences are shown in Table 6.7. It can be noticed that both layers classify properly the sub-sequences of the *Melting Temperatures* time series, predicting that an alarm will be triggered in the sub-sequence *s2* and the contrary case for the sub-sequence *s1*. The same predictions are obtained for the *Die Zone 4 Temperatures* time series which seems reasonable since the sensor capturing these series is close to the sensor measuring the melting temperatures and therefore, both time series are really similar and highly correlated. Finally, when dealing with a different nature time series, the *softmax* layer predicts that a *Plastic Temperature not Reached in the Die Entry Alarm* will be triggered in both sub-sequences which is not correct since it is unknown whether in those sub-sequences an alarm would be activated or not. In this case, the *openmax* layer is able to reject to classify the sample sub-sequence and thus, it predicts it as *unknown*.

TABLE 6.7: Open set recognition example results

		Activation Layer					
			Softmax			Openmax	
Time Series	Label	Class	Probability	Predicted	Class	Probability	Predicted
Melting Temperatures S1	No-Alarm	No-Alarm Alarm Unknown	9.9994e-01 5.0694e-05 -	No-Alarm	No-Alarm Alarm Unknown	9.4576e-01 9.3959e-04 5.3294e-02	No-Alarm
Die Zone 4 Temperatures S1	-	No-Alarm Alarm Unknown	9.9985e-01 1.4477e-04 -	No-Alarm	No-Alarm Alarm Unknown	0.9285 0.00189 0.06953	No-Alarm
Melting Pressure S1	-	No-Alarm Alarm Unknown	1.0282e-20 ≈ 1.000e+00 -	Alarm	No-Alarm Alarm Unknown	3.3574e-13 2.7945e-01 7.2054e-01	Unknown
Melting Temperatures S2	Alarm	No-Alarm Alarm Unknown	0.0065 0.9934 -	Alarm	No-Alarm Alarm Unknown	0.0230 0.5123 0.4646	Alarm
Die Zone 4 Temperatures S2	-	No-Alarm Alarm Unknown	0.0032 0.9967 -	Alarm	No-Alarm Alarm Unknown	0.0144 0.5249 0.4606	Alarm
Melting Pressure S2	-	No-Alarm Alarm Unknown	2.3415e-09 ≈ 1.000e+00 -	Alarm	No-Alarm Alarm Unknown	2.1567e-06 4.9905e-01 5.0094e-01	Unknown

6.5 Alarm Prediction System Performance Evaluation

Previous sections have presented the different components (models) involved in the alarm prediction system individually. However, in order to predict alarms, these models have to be combined in a single system, following the approach described in Section 6.2.2. This section shows first, the performance of the proposed alarm prediction system as a whole, to predict the three different types of alarms by using the built analyzers over the forecasted measurements by the built forecaster; and then, an example of the prediction of an alarm on a real use case.

6.5.1 System Performance

With the objective of testing a real use case in a real scenario, the evaluation dataset (unseen data for the built models) has been used to evaluate the system performance. First, the measurements of each sensor in the evaluation dataset have been predicted by using the built forecaster. Iteratively, each sub-sequence of five measurements in the evaluation dataset (10 and 15 respectively for the time horizons of 10 and 15 minutes) has been replaced with the predicted values by the forecaster for the previous 300 observations. Then, the predicted values have been transformed by using the same approach described in Section 6.4.1 for the *classification data* (the sub-sequences of predicted values have been labelled with the corresponding true labels from the evaluation dataset). Finally, those values have been used to evaluate the analyzers.

The performance of the analyzers, to discriminate normal operation sub-sequences from sub-sequences on which an alarm has been activated, has been evaluated by using the area under Relative Operating Characteristic (ROC) curve (AUC-ROC) metric. Table 6.8 summarizes the AUC-ROC value obtained by each analyzer when predicting the alarms on the forecasted values for each time horizon for the evaluation dataset compared to the AUC-ROC value obtained when predicting alarms over the original evaluation dataset.

TABLE 6.8: Time Series analyzers evaluation results over the forecasted data (AUC ROC)

Analyzer	Raw Data EVAL	Forecasted (steps-ahead) 5	10	15
Melting Temperatures	0.99937	0.99937	0.99937	0.99812
Incorrect Temperatures	0.99270	0.99270	0.99270	0.99270
Die Zone 2	0.97381	0.97381	0.97381	0.97381

6.5.2 Alarm Prediction Example

Figure 6.13 shows the materialization of the proposed system in a real use case. In the figure, it can be seen on the one hand, a 3D model of an extruder machine on which an alarm activation has been predicted with a time horizon of 15 minutes. Taking into account that the alarm has been predicted 15 minutes ahead (the less critical time horizon), the extruder zones corresponding to the sensors associated to that alarm are colored in yellow, indicating that something could be wrong (see the warning icon and the extruder part highlighted in yellow). Furthermore, in the chart displayed under the extruder it can be seen in black the last measurements of the extruder (real measurements), whereas in red, orange and yellow the predicted measurements of the extruder with a time horizon of 5, 10 and 15 minutes respectively (the vertical red line indicates the activation of the real alarm).

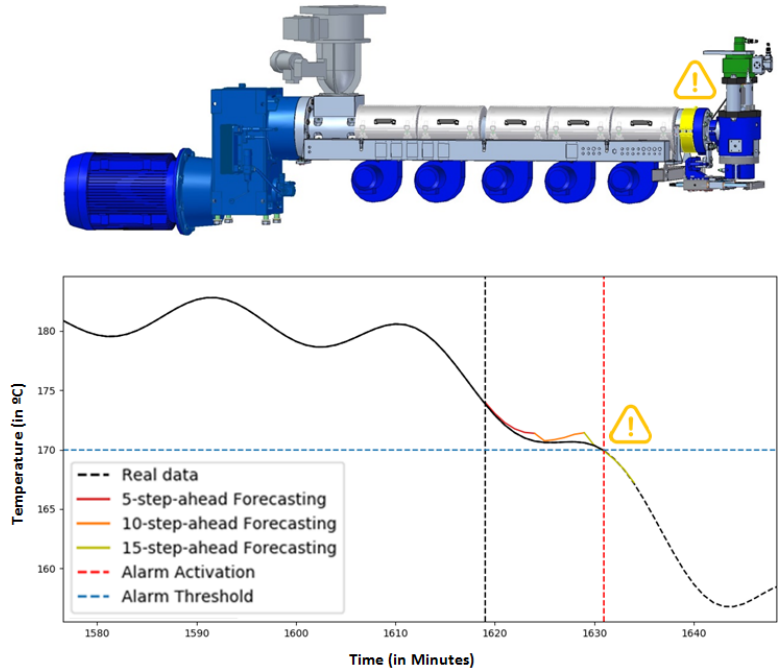


FIGURE 6.13: Example of an alarm prediction using the proposed system

6.6 Conclusions

This chapter presents an alarm prediction system that applies deep learning techniques to predict the activation of diverse type of alarms that could serve for different purposes, such as the predictive maintenance of the equipment or production and product quality optimization. The system follows a two-stage *forecaster-analyzer* approach on which first a LSTM neural networks based forecaster predicts the future measurements of the sensors and then, distinct analyzers based on Residual Neural Networks determine whether the predicted measurements will trigger an alarm or not.

Regarding the forecaster, different models have been tested and compared for the selection of an adequate forecasting model. Based on the obtained results, a time series forecaster has been built by using LSTM neural networks, that achieves a great performance when predicting the future measurements of 11 different sensors implanted in an extruder machine. Moreover, the suitability of the built model for Smart Manufacturing scenarios with dynamically changing processes, and its ability to adapt to non-stationary environments on which concept drift could occur has also been tested, under the perspective of system applicability in Smart Manufacturing scenarios.

With respect to the analyzers, three different analyzers have been built using Residual Neural Networks to detect if in a time series sub-sequence an alarm will be triggered or not. The built analyzers have demonstrated that Residual Neural Networks are able to detect efficiently, not only alarms that can be modeled with simple rules based on the activation condition, but also more complex alarms on which it is unknown when the activation condition will be fulfilled, and thus, cannot be modeled by rules. That behaviour reinforces the interest of using pattern matching-based analyzers. Moreover, as shown in this work, the ability of these models to deal also with unknown situations not seen before could be really interesting in Smart Manufacturing scenarios.

Lastly, concerning the followed two-stage approach, it requires building and training two different models instead of using a single model to predict the alarms in a straightforward way. However, using different models for forecasting the future measurements of the sensors and for detecting alarms in the predicted measurements gives the system more modularity and makes it more flexible to changes and extensible to different predictive analysis tasks, since the predicted measurements of the sensors could be used for other analysis processes.

Chapter 7

Conclusions

The fourth industrial revolution has given rise to what is called Smart Manufacturing, addressing the use of modern Information Technologies to transform the acquired data across the product life-cycle, into manufacturing intelligence, in order to achieve meaningful improvements in all aspects of manufacturing. Driven by the data, Smart Manufacturing opens the door to new possibilities to transform the production process and the business model of the whole manufacturing industry. On the one hand, the adoption of these data-driven approaches and the value extracted from data insights is expected to produce significant gains in the production systems performance, the quality of produced goods and profit in general. On the other hand, it enables shifting the business model of companies towards data-driven servitization strategies on which not only equipment and goods are provided to their customers, but also value-added services.

The rise of Smart Manufacturing, together with the strategic initiatives carried out worldwide, have promoted its adoption among manufacturers, who are increasingly interested in boosting data-driven applications for different purposes, such as product quality control, fault diagnosis, predictive maintenance of equipment, etc. However, the adoption of these data-driven approaches faces diverse technological challenges with regard to the key-enabling technologies supporting the whole life-cycle of the manufacturing data. In this regard, the collaboration with different industrial agents involved in the deployment of Smart Manufacturing approaches, has facilitated a first-hand identification and understanding of some of these challenges, for which the three main contributions presented in this dissertation aim at providing valuable solutions. The following sections presents these contributions, together with the publications supporting them and some research lines that have been identified for further work.

7.1 Contributions

Among the different opportunities that arise for relevant contributions aligned with the achievement of the goals established for Smart Manufacturing approaches and the challenges related to them, this research work has been focused on three of them; and therefore, three main contributions are presented. These contributions are supported by the identified opportunities in the relevant research areas related to the different steps that

compound the *manufacturing data life-cycle*. With regard to the early stages of the manufacturing data life-cycle, the first two contributions propose solutions that enable an efficient storage of the time series and an automatization of their pre-processing for further analysis in the later stages. Regarding these analysis, the last contribution propose a solution that leverages advanced data analysis techniques to conduct a predictive maintenance of equipment.

The main differential value of the contributions presented in this dissertation is that they integrate innovative proposals in the state of the art of the related research fields, with some of the challenges and requirements that arise from real-world problems, in order to develop new solution proposals. Moreover, the utility and applicability of the proposed contributions has been contrasted and validated in a real-world context representing a relevant instance of the Smart Manufacturing scenarios where these contributions are targeted at. In the following subsections each of these contributions is presented.

7.1.1 A System that Efficiently Guides a Data Engineer in Time Series Pre-processing

There is an increasing interest among manufacturers in exploiting the potential of the captured data during the manufacturing process for different data analysis purposes. However, in order to extract knowledge and useful information from those data, they must be first pre-processed. The pre-processing of massive amounts of data generated by multiple heterogeneous IIoT devices poses an important challenge, mainly due to the multiple existing alternatives to treat specific problems, the diversity of treatments that each data may require, and the lack of automatic approaches to determine which techniques to apply in each situation.

In this regard, this research work contributes with the design and development of a system that facilitates the pre-processing task of the captured time-series data through an automatized approach that helps in the selection of the most adequate pre-processing techniques to apply to each data type. The proposed system is available as a visual-interactive web system that provides a wide range of pre-processing techniques for the different tasks related to time series cleaning and dimensionality reduction. Furthermore, it provides some recommendations on which techniques are more suitable and have more potential to work for each type of time series.

With respect to the time series cleaning techniques, the system provides a set of representative techniques that cover a broad spectrum of type of series. In particular, it provides five techniques for the imputation of missing values, two techniques for removing noise, and two techniques for detecting and handling outliers. Moreover, the system also provides some general information about the characteristics of a time series for which the application of a concrete technique is more suitable, and an appropriate range of parameters values that the technique may require.

With regard to the reduction techniques, the system provides nine different techniques from the most representative families analyzed and discussed in the literature. The selection of the most suitable techniques to apply to each series is supported by a novel machine learning-based model that given a time series, recommends the most appropriate reduction techniques that allow obtaining an adequate reduced syntactic representation of raw time-series data, while preserving their main characteristics.

7.1.2 A Three-Level Hierarchical Architecture for an Efficient Storage of Time-Series Data

The deployment of IIoT devices in Smart Manufacturing scenarios allows to capture big amounts of data, regarding different magnitudes or indicators of interest that are usually stored for further analysis processes. However, the accumulation of massive amounts of data generates a problem related to the considerable costs associated with the storage resources in Cloud Computing environments. In this regard, this research work contributes with the design of an architecture that helps to manage and reduce the required data storage resources and, consequently, its associated costs.

The proposed architecture follows a multi-temperature data management paradigm on which the temperature tiers hot, warm, and cold are considered; and thereby, it is materialized as a three-level hierarchical architecture. The main novelty of the proposed architecture relies on the third level of the architecture, where a reduced representation of the time series (obtained by applying time series reduction techniques) is stored to reduce the required data storage resources. It has been implemented by using four different types of database engines (InfluxDB, Cassandra, MongoDB and Neo4J), and its performance has been tested and contrasted on each database engine under the perspective of two main different dimensions: used storage space (and its associated costs), and total query time for answering four different types of queries (proposed by domain experts from a real manufacturing environment).

From the performance results it has been observed that the proposed architecture allows to optimize cloud storage resources (and its associated costs) without hampering the suitability of the database engines for the management of industrial time-series data. The achieved reduction in terms of data storage space is remarkable, at least a 78% even when dealing with a specialized time series database engine. Conversely, the compromised information due to the use of lossy reduction techniques is lower than a 5% in terms of RMSE (acceptable in the considered scenario) and the introduced latency when accessing the data for some types of queries is assumable (a small latency of 30 seconds when dealing with a month's worth of data). Moreover, the performance results have been analyzed applying the Design of Experiments methodology, in order to recommend the most suitable configuration of the proposed architecture, for the real-world scenario where the data used to test the architecture comes from.

7.1.3 A Flexible Alarm Prediction System Following a Forecaster-Analyzer Approach

The increasing interest among manufacturers in exploiting the potential of large volumes of manufacturing data for diverse purposes has led to the introduction of artificial intelligence techniques in these scenarios, to conduct different types of analysis over the captured data. Among the different applications of these techniques in the Smart Manufacturing context, one of the most prominent ones is the predictive maintenance of equipment as it directly affects the service life of equipment and its production efficiency. In this regard, this research work contributes with the design and development of an alarm prediction system that applies deep learning techniques to predict the activation of diverse types of alarms that could serve for different purposes, such as the predictive maintenance of the equipment or production optimization.

The proposed system follows a two-stage forecaster-analyzer approach on which first, a time series forecaster predicts the future measurements of various sensors implanted on an extruder machine, and then, distinct analyzers (i.e., classifiers) determine whether the predicted measurements will trigger an alarm or not. It has been tested with time-series data coming from a real production plant, on which it has shown a great performance to predict diverse types of alarms in three different time horizons (5, 10 and 15 minutes). Moreover, the proposed system supports some features that make it particularly suitable for Smart Manufacturing scenarios. On the one hand, the built system is suitable for multi-sensor time-series data forecasting in non-stationary environments such as Smart Manufacturing scenarios with dynamically changing processes. On the other hand, it is able to detect alarms that can be modeled with simple rules based on the activation condition, and also more complex alarms on which it is unknown when the activation condition will be fulfilled, and it has shown the possibility of dealing with unseen situations that can emerge unexpectedly. Furthermore, the followed approach to build the system makes it easily extensible to other predictive analysis tasks, since the predicted measurements of the sensors could be used for other processes, such as anomaly detection, prediction of other types of alarms, etc.

7.2 Publications

Part of the work realized in the contributions presented in this thesis has already been presented and discussed in distinct peer-reviewed forums. The publications that endorse this thesis are listed below.

Selected Publications

1. The conference paper entitled *A multi-services architecture for smart manufacturing scenarios* [VBD+18] presented in the *International Conference on Industrial Internet of Things and Smart Manufacturing (IoTSM*, held in London (United Kingdom) in 2018. It presents

the design of a global framework to provide multiple interconnected services in Smart Manufacturing scenarios, that frames the time-series data *Pre-processing* and *Reduction* services presented in Chapter 4, Section 4.5.

2. The conference paper entitled *I4TSRS: A system to assist a data engineer in time-series dimensionality reduction in industry 4.0 scenarios* [VDI+18] published in the proceedings of the *27th ACM International Conference on Information and Knowledge Management (CIKM)*, held in Torino (Italy) in 2018. It presents the I4TSRS Web App, that allows to obtain an adequate reduced representation of time-series data. The I4TSRS Web App is one of the two main applications that compound the *Pre-processing System* presented in Chapter 4, Section 4.5.

3. The conference paper entitled *I4TSPS: a visual-interactive web system for industrial time-series pre-processing* [VVD+18] published in the proceedings of the *2018 IEEE International Conference on Big Data (Big Data)*, held in Seattle (Washington) in 2018. It presents the visual-interactive Web systems (I4TSPS Web App and the I4TSRS Web App) that provide a wide range of pre-processing techniques for the different tasks related to time series cleaning and dimensionality reduction (respectively), and help in the selection of the most adequate techniques to apply to each data type. Those systems, compound the *Pre-processing System* that efficiently guides a data engineer in the task of pre-processing raw time-series data presented in Chapter 4.

4. The conference paper entitled *A Hierarchical Storage System for Industrial Time-Series Data* [VRD+19] published in the proceedings of the *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, held in Helsinki-Espoo (Finland) in 2019. It presents the three-level hierarchical architecture for time-series data storage on cloud environments, that helps to manage and reduce the required data storage resources (and consequently its associated costs).

5. The journal article entitled *A Three Level Hierarchical Architecture for an Efficient Storage of Industry 4.0 Data* [VRD+] accepted for its publication in the *Computers in Industry* journal. It formalizes and extends the performance tests of the three-level hierarchical architecture already presented in [VRD+19]. The proposed architecture, together with the performance results are presented in Chapter 5.

6. The journal article entitled *A Flexible Alarm Prediction System for Smart Manufacturing Scenarios Following a Forecaster-Analyzer Approach*, which remains in a minor revision process in the *Journal of Intelligent Manufacturing*. It presents the alarm prediction system that allows to anticipate the activation of different types of alarms that can be produced on a real Smart Manufacturing scenario. The proposed system is presented in Chapter 6.

7.3 Future Work

Considering distinct aspects of the research fields and knowledge areas related to the proposed contributions in this dissertation, different research directions have been identified for future work. In this section, some of the most prominent ones for each contribution are presented.

With respect to the proposed pre-processing system, although the constructed model fits properly to the data with which it has been built and tested, the followed approach for building the model, makes it highly dependant on the application scenario. The different steps required to discover the time series families, extract the features that compose their syntactic characterization and classify them into the discovered families; result into a complex model building process with limited adaptability to deal with new time series and low replicability in different scenarios. In this regard, the automatic feature learning and high-volume modelling capabilities of deep learning-based models (as those considered in the alarm prediction system), provide advanced analytics tools with lower complexity and higher flexibility, that present as a really suitable option for this kind of problems. Moreover, these models have also demonstrated the possibility not only to deal with unseen situations (e.g., open-set recognition), but also to detect new classes (i.e., novelty detection), which could be really interesting for a system like the one proposed in this research work, where new types of time series with new pre-processing requirements could emerge unexpectedly. Therefore, additional research in this line would involve leveraging the experience and knowledge acquired during the construction of the deep learning-based models in the alarm prediction system, to adopt the use of these type of models for the recommendation of the most suitable techniques for both, cleaning and reduction.

With regard to the future research directions for the proposed architecture, one concern and open research problem on which the other researchers in the field of Time Series Management Systems (TSMSs) put the focus, is the scalability of the systems for the storage and processing of ever-increasing large-scale time-series data. With respect to the scalability of the storage, the proposed architecture has already shown the potential resource that time series approximation techniques could represent to reduce data storage resources and its associated costs. Therefore, the future work would be focused in the efficient processing of the data, where Approximate Query Processing (AQP) has been stated as one of the open research problems proposed by the researchers in the field. In this regard, further research would involve testing the capabilities of the considered techniques for AQP and considering new techniques (e.g., mathematical models) that allow to reduce data storage resources and also efficient AQP.

Finally, concerning the built alarm prediction system and the models involved in it, the ability of models to deal with unknown situations and their capability to adapt to new conditions has been stated as one of the recent research topics to which researchers are devoting many efforts. This properties result particularly interesting for non-stationary environments,

such as Smart Manufacturing scenarios with dynamically changing processes, that could hamper the performance of the built models when the environment conditions change. In this regard, the proposed alarm prediction system has shown the capability of the considered forecasting model to deal with the concept drift phenomenon and the possibility to deal with unseen situations in the considered classifiers. Future work would involve further research in this lines, as well as in online learning models for streaming data analytics. Moreover, the followed *forecaster-analyzer* approach for building the system makes it flexible and extensible for other predictive analysis tasks; therefore, further research would also involve building more analyzers for different purposes, such as anomaly detection or product quality control.

Appendix A

Volumes of Data Generated Across Time for Different Scenarios

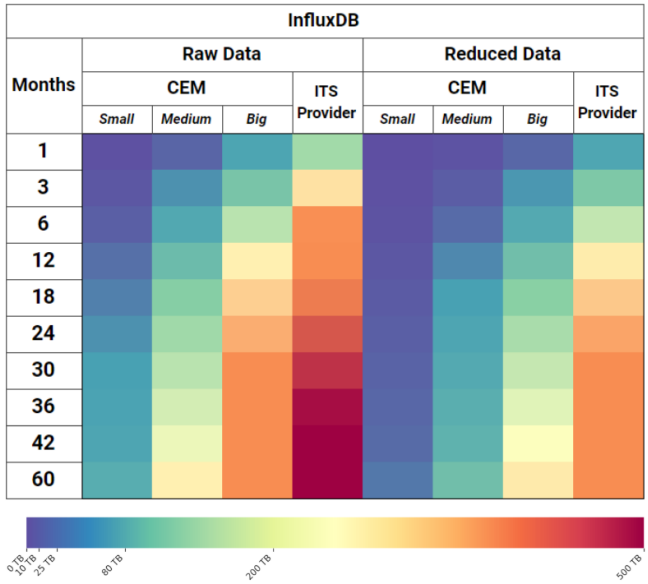


FIGURE A.1: Volumes of data generated (in TB) across time for different scenarios in InfluxDB

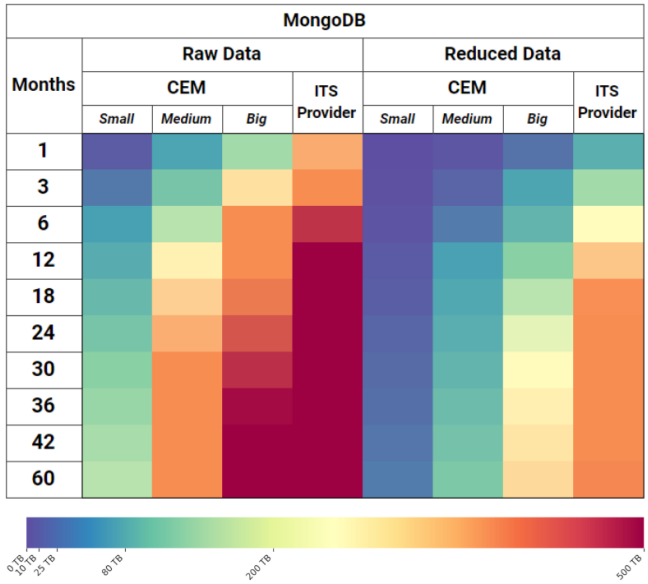


FIGURE A.2: Volumes of data generated (in TB) across time for different scenarios in MongoDB

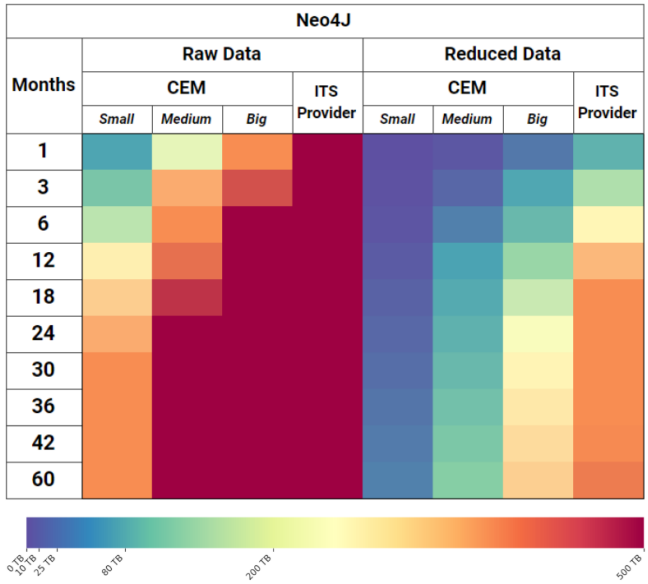


FIGURE A.3: Volumes of data generated (in TB) across time for different scenarios in Neo4J

Appendix B

Query Times for Different Levels of Implantation of the Architecture

TABLE B.1: Total Query times summary (in ms) with 25% of implantation

Q.	DBMS	75% Raw Series & 25% Reduced Series Total Time = (QueryTime+RecTime) or (QueryTime+RecTime+ProcTime) Day	Week	Month
1	InfluxDB	181.60 (59.43+122.17) 290.41	627.69 (476.28+151.41) 1244.24	2190.95 (1533.87+657.08) 4668.61
	Cassandra	(168.24+122.17) 317.30	(1092.83+151.41) 1355.93	4011.54+657.08) 4387.53
	MongoDB	(195.13+122.17) 1926.17	(1204.52+151.41) 4728.95	(3730.46+657.08) 7490.76
	Neo4J	(1804.00+122.17)	(4577.54+151.41)	(6833.68+657.08)
2	InfluxDB	768.21 (55.84+122.17+590.21) 966.22	2926.97 (363.64+151.41+2411.92) 4165.37	8433.63 (1158.03+657.08+6618.53) 12323.49
	Cassandra	(253.84+122.17+590.21) 986.26	(1602.05+151.41+2411.92) 4281.56	(5047.89+657.08+6618.53) 12642.67
	MongoDB	(273.89+122.17+590.21) 2401.59	(1718.23+151.41+2411.92) 7415.87	(5367.07+657.08+6618.53) 15921.18
	Neo4J	(1689.22+122.17+590.21)	(4852.54+151.41+2411.92)	(8645.58+657.08+6618.53)
3	InfluxDB	248.38 (53.71+32.17+162.50) 960.70	1036.70 (43.77+54.79+938.14) 5515.05	5863.20 (498.72+756.13+4608.35) 22277.34
	Cassandra	(766.03+32.17+162.50) 299.29	(4522.12+54.79+938.14) 1650.82	(16912.86+756.13+4608.35) 7403.51
	MongoDB	(104.62+32.17+162.50) 2011.48	(657.89+54.79+938.14) 3316.73	(2039.03+756.13+4608.35) 10960.95
	Neo4J	(1816.81+32.17+162.50)	(2323.80+54.79+938.14)	(5596.47+756.13+4608.35)
4	InfluxDB	4690.26 (268.09+122.17+4300.00) 4932.87	23632.02 (2030.61+151.41+21450.00) 24511.13	100372.65 (6365.57+657.08+93350.00) 104458.10
	Cassandra	(510.70+122.17+4300.00) 4614.33	(2909.72+151.41+21450.00) 22783.72	(10451.02+657.08+93350.00) 97703.41
	MongoDB	(192.16+122.17+4300.00) 10078.40	(1182.31+151.41+21450.00) 33133.81	(3696.33+657.08+93350.00) 114004.59
	Neo4J	(5656.23+122.17+4300.00)	(11532.40+151.41+21450.00)	(19997.52+657.08+93350.00)

TABLE B.2: Total Query times summary (in ms) with 50% of implantation

Q.	DBMS	50% Raw Series & 50% Reduced Series Total Time = (QueryTime+RecTime) or (QueryTime+RecTime+ProcTime) Day	Week	Month
1	InfluxDB	283.96 (39.63+244.34) 357.40	620.36 (317.54+302.82) 1032.16	2336.76 (1022.61+1314.15) 3990.61
	Cassandra	(113.07+244.34) 374.45	(729.34+302.82) 1105.87	(2676.46+1314.15) 3801.20
	MongoDB	(130.11+244.34) 1451.08	(803.05+302.82) 3362.51	(2487.05+1314.15) 5887.10
	Neo4J	(1206.74+244.34)	(3059.69+302.82)	(4572.95+1314.15)
2	InfluxDB	1461.98 (37.23+244.34+1180.42) 1594.89	5369.10 (242.44+302.82+4823.84) 6195.48	15323.26 (772.05+1314.15+13237.06) 17918.56
	Cassandra	(170.14+244.34+1180.42) 1607.38	(1068.82+302.82+4823.84) 6272.21	(3367.36+1314.15+13237.06) 18129.39
	MongoDB	(182.63+244.34+1180.42) 2554.78	(1145.55+302.82+4823.84) 8369.46	(3578.19+1314.15+13237.06) 20334.59
	Neo4J	(1130.03+244.34+1180.42)	(3242.80+302.82+4823.84)	(5783.38+1314.15+13237.06)
3	InfluxDB	425.17 (35.83+64.35+325.00) 902.12	2015.06 (29.21+109.57+1876.29) 5002.94	11061.51 (332.55+1512.26+9216.71) 22010.34
	Cassandra	(512.78+64.35+325.00) 459.12	(3017.08+109.57+1876.29) 2424.56	(11281.38+1512.26+9216.71) 12088.59
	MongoDB	(69.78+64.35+325.00) 1605.91	(438.71+109.57+1876.29) 3553.90	(1359.63+1512.26+9216.71) 14511.20
	Neo4J	(1216.57+64.35+325.00)	(1568.05+109.57+1876.29)	(3782.24+1512.26+9216.71)
4	InfluxDB	4723.08 (178.74+244.34+4300.00) 4885.37	23106.59 (1353.77+302.82+21450.00) 23693.20	98907.94 (4243.79+1314.15+93350.00) 101632.27
	Cassandra	(341.04+244.34+4300.00) 4672.45	(1940.38+302.82+21450.00) 22541.05	(6968.12+1314.15+93350.00) 97128.41
	MongoDB	(128.12+244.34+4300.00) 8327.93	(788.23+302.82+21450.00) 29461.22	(2464.26+1314.15+93350.00) 108046.04
	Neo4J	(3783.59+244.34+4300.00)	(7708.40+302.82+21450.00)	(13381.89+1314.15+93350.00)

TABLE B.3: Total Query times summary (in ms) with 75% of implantation

Q.	DBMS	25% Raw Series & 75% Reduced Series Total Time = (QueryTime+RecTime) or (QueryTime+RecTime+ProcTime) Day	Week	Month
1	InfluxDB	386.33 (19.82+366.50) 424.40	613.02 (158.79+454.23) 820.09	2482.58 (511.35+1971.23) 3312.60
	Cassandra	(57.89+366.50) 431.59	(365.86+454.23) 855.80	(1341.37+1971.23) 3214.86
	MongoDB	(65.09+366.50) 975.98	(401.57+454.23) 1996.06	(1243.63+1971.23) 4283.44
	Neo4J	(609.48+366.50)	(1541.83+454.23)	(2312.21+1971.23)
2	InfluxDB	2155.75 (18.63+366.50+1770.62) 2223.55	7811.22 (121.24+454.23+7235.75) 8225.58	22212.88 (386.07+1971.23+19855.58) 23513.63
	Cassandra	(86.43+366.50+1770.62) 2228.49	(535.60+454.23+7235.75) 8262.85	(1686.82+1971.23+19855.58) 23616.11
	MongoDB	(91.36+366.50+1770.62) 2707.96	(572.87+454.23+7235.75) 9323.04	(1789.30+1971.23+19855.58) 24747.99
	Neo4J	(570.83+366.50+1770.62)	(1633.06+454.23+7235.75)	(2921.19+1971.23+19855.58)
3	InfluxDB	601.95 (17.94+96.52+487.49) 843.53	2993.43 (14.64+164.36+2814.43) 4490.82	16259.81 (166.37+2268.38+13825.06) 21743.33
	Cassandra	(259.52+96.52+487.49) 618.95	(1512.04+164.36+2814.43) 3198.30	(5649.89+2268.38+13825.06) 16773.66
	MongoDB	(34.94+96.52+487.49) 1200.33	(219.52+164.36+2814.43) 3791.07	(680.22+2268.38+13825.06) 18061.45
	Neo4J	(616.32+96.52+487.49)	(812.29+164.36+2814.43)	(1968.01+2268.38+13825.06)
4	InfluxDB	4755.89 (89.39+366.50+4300.00) 4837.88	22581.16 (676.93+454.23+21450.00) 22875.26	97443.22 (2122.00+1971.23+93350.00) 98806.45
	Cassandra	(171.37+366.50+4300.00) 4730.58	(971.03+454.23+21450.00) 22298.37	(3485.22+1971.23+93350.00) 96553.42
	MongoDB	(64.07+366.50+4300.00) 6577.45	(394.14+454.23+21450.00) 25788.64	(1232.19+1971.23+93350.00) 102087.48
	Neo4J	(1910.95+366.50+4300.00)	(3884.41+454.23+21450.00)	(6766.25+1971.23+93350.00)

Appendix C

Conducted Experiments in the DOE Methodology

Table C.1: Conducted experiments in the DOE methodology

Run	DBMS Type	% Implantation	Query Type	Time Window	Query Time	Score	DSS
1	InfluxDB	0	1	day	79.23	33.2766	49.39
2	InfluxDB	0	1	week	635.03	76.2036	49.39
3	InfluxDB	0	1	month	2045.13	122.7078	49.39
4	InfluxDB	0	2	day	74.44	7.8162	49.39
5	InfluxDB	0	2	week	484.84	14.5452	49.39
6	InfluxDB	0	2	month	1544.01	23.16015	49.39
7	InfluxDB	0	3	day	71.6	6.265	49.39
8	InfluxDB	0	3	week	58.33	1.45825	49.39
9	InfluxDB	0	3	month	664.89	8.311125	49.39
10	InfluxDB	0	4	day	4657.44	407.526	49.39
11	InfluxDB	0	4	week	24157.45	603.93625	49.39
12	InfluxDB	0	4	month	101837.36	1272.967	49.39
13	InfluxDB	25	1	day	181.595	76.2699	39.82
14	InfluxDB	25	1	week	627.6925	75.3231	39.82
15	InfluxDB	25	1	month	2190.945	131.4567	39.82
16	InfluxDB	25	2	day	768.21	80.66205	39.82
17	InfluxDB	25	2	week	2926.9675	87.809025	39.82
18	InfluxDB	25	2	month	8433.6325	126.5044875	39.82
19	InfluxDB	25	3	day	248.3825	21.73346875	39.82
20	InfluxDB	25	3	week	1036.695	25.917375	39.82
21	InfluxDB	25	3	month	5863.1975	73.28996875	39.82
22	InfluxDB	25	4	day	4690.2575	410.3975313	39.82
23	InfluxDB	25	4	week	23632.02	590.8005	39.82
24	InfluxDB	25	4	month	100372.6475	1254.658094	39.82
25	InfluxDB	50	1	day	283.96	119.2632	30.25
26	InfluxDB	50	1	week	620.355	74.4426	30.25
27	InfluxDB	50	1	month	2336.76	140.2056	30.25
28	InfluxDB	50	2	day	1461.98	153.5079	30.25
29	InfluxDB	50	2	week	5369.095	161.07285	30.25
30	InfluxDB	50	2	month	15323.255	229.848825	30.25
31	InfluxDB	50	3	day	425.165	37.2019375	30.25
32	InfluxDB	50	3	week	2015.06	50.3765	30.25
33	InfluxDB	50	3	month	11061.505	138.2688125	30.25
34	InfluxDB	50	4	day	4723.075	413.2690625	30.25
35	InfluxDB	50	4	week	23106.59	577.66475	30.25
36	InfluxDB	50	4	month	98907.935	1236.349188	30.25
37	InfluxDB	75	1	day	386.325	162.2565	20.67

38	InfluxDB	75	1	week	613.0175	73.5621	20.67
39	InfluxDB	75	1	month	2482.575	148.9545	20.67
40	InfluxDB	75	2	day	2155.75	226.35375	20.67
41	InfluxDB	75	2	week	7811.2225	234.336675	20.67
42	InfluxDB	75	2	month	22212.8775	333.1931625	20.67
43	InfluxDB	75	3	day	601.9475	52.67040625	20.67
44	InfluxDB	75	3	week	2993.425	74.835625	20.67
45	InfluxDB	75	3	month	16259.8125	203.2476563	20.67
46	InfluxDB	75	4	day	4755.8925	416.1405938	20.67
47	InfluxDB	75	4	week	22581.16	564.529	20.67
48	InfluxDB	75	4	month	97443.2225	1218.040281	20.67
49	InfluxDB	100	1	day	488.69	205.2498	11.04
50	InfluxDB	100	1	week	605.68	72.6816	11.04
51	InfluxDB	100	1	month	2628.39	157.7034	11.04
52	InfluxDB	100	2	day	2849.52	299.1996	11.04
53	InfluxDB	100	2	week	10253.35	307.6005	11.04
54	InfluxDB	100	2	month	29102.5	436.5375	11.04
55	InfluxDB	100	3	day	778.73	68.138875	11.04
56	InfluxDB	100	3	week	3971.79	99.29475	11.04
57	InfluxDB	100	3	month	21458.12	268.2265	11.04
58	InfluxDB	100	4	day	4788.71	419.012125	11.04
59	InfluxDB	100	4	week	22055.73	551.39325	11.04
60	InfluxDB	100	4	month	95978.51	1199.731375	11.04
61	MongoDB	0	1	day	260.15	109.263	240.13
62	MongoDB	0	1	week	1605.99	192.7188	240.13
63	MongoDB	0	1	month	4973.87	298.4322	240.13
64	MongoDB	0	2	day	365.15	38.34075	240.13
65	MongoDB	0	2	week	2290.91	68.7273	240.13
66	MongoDB	0	2	month	7155.95	107.33925	240.13
67	MongoDB	0	3	day	139.46	12.20275	240.13
68	MongoDB	0	3	week	877.08	21.927	240.13
69	MongoDB	0	3	month	2718.43	33.980375	240.13
70	MongoDB	0	4	day	4556.2	398.6675	240.13
71	MongoDB	0	4	week	23026.4	575.66	240.13
72	MongoDB	0	4	month	98278.4	1228.48	240.13
73	MongoDB	25	1	day	317.2975	133.26495	184.23
74	MongoDB	25	1	week	1355.9275	162.7113	184.23
75	MongoDB	25	1	month	4387.5325	263.25195	184.23
76	MongoDB	25	2	day	986.2625	103.5575625	184.23
77	MongoDB	25	2	week	4281.5575	128.446725	184.23
78	MongoDB	25	2	month	12642.67	189.64005	184.23
79	MongoDB	25	3	day	299.29	26.187875	184.23
80	MongoDB	25	3	week	1650.82	41.2705	184.23
81	MongoDB	25	3	month	7403.5075	92.54384375	184.23
82	MongoDB	25	4	day	4614.325	403.7534375	184.23
83	MongoDB	25	4	week	22783.7225	569.5930625	184.23
84	MongoDB	25	4	month	97703.405	1221.292563	184.23
85	MongoDB	50	1	day	374.445	157.2669	128.33
86	MongoDB	50	1	week	1105.865	132.7038	128.33
87	MongoDB	50	1	month	3801.195	228.0717	128.33
88	MongoDB	50	2	day	1607.375	168.774375	128.33
89	MongoDB	50	2	week	6272.205	188.16615	128.33
90	MongoDB	50	2	month	18129.39	271.94085	128.33
91	MongoDB	50	3	day	459.12	40.173	128.33
92	MongoDB	50	3	week	2424.56	60.614	128.33
93	MongoDB	50	3	month	12088.585	151.1073125	128.33
94	MongoDB	50	4	day	4672.45	408.839375	128.33
95	MongoDB	50	4	week	22541.045	563.526125	128.33
96	MongoDB	50	4	month	97128.41	1214.105125	128.33
97	MongoDB	75	1	day	431.5925	181.26885	72.42
98	MongoDB	75	1	week	855.8025	102.6963	72.42
99	MongoDB	75	1	month	3214.8575	192.89145	72.42
100	MongoDB	75	2	day	2228.4875	233.9911875	72.42
101	MongoDB	75	2	week	8262.8525	247.885575	72.42

102	MongoDB	75	2	month	23616.11	354.24165	72.42
103	MongoDB	75	3	day	618.95	54.158125	72.42
104	MongoDB	75	3	week	3198.3	79.9575	72.42
105	MongoDB	75	3	month	16773.6625	209.6707813	72.42
106	MongoDB	75	4	day	4730.575	413.9253125	72.42
107	MongoDB	75	4	week	22298.3675	557.4591875	72.42
108	MongoDB	75	4	month	96553.415	1206.917688	72.42
109	MongoDB	100	1	day	488.74	205.2708	16.53
110	MongoDB	100	1	week	605.74	72.6888	16.53
111	MongoDB	100	1	month	2628.52	157.7112	16.53
112	MongoDB	100	2	day	2849.6	299.208	16.53
113	MongoDB	100	2	week	10253.5	307.605	16.53
114	MongoDB	100	2	month	29102.83	436.54245	16.53
115	MongoDB	100	3	day	778.78	68.14325	16.53
116	MongoDB	100	3	week	3972.04	99.301	16.53
117	MongoDB	100	3	month	21458.74	268.23425	16.53
118	MongoDB	100	4	day	4788.7	419.01125	16.53
119	MongoDB	100	4	week	22055.69	551.39225	16.53
120	MongoDB	100	4	month	95978.42	1199.73025	16.53
121	Cassandra	0	1	day	223.41	93.8322	155.83
122	Cassandra	0	1	week	1456.31	174.7572	155.83
123	Cassandra	0	1	month	5346.62	320.7972	155.83
124	Cassandra	0	2	day	337.55	35.44275	155.83
125	Cassandra	0	2	week	2135.27	64.0581	155.83
126	Cassandra	0	2	month	6728.42	100.9263	155.83
127	Cassandra	0	3	day	1019.29	89.187875	155.83
128	Cassandra	0	3	week	6027.16	150.679	155.83
129	Cassandra	0	3	month	22544.35	281.804375	155.83
130	Cassandra	0	4	day	4980.36	435.7815	155.83
131	Cassandra	0	4	week	25329.06	633.2265	155.83
132	Cassandra	0	4	month	107283.92	1341.049	155.83
133	Cassandra	25	1	day	290.405	121.9701	119.81
134	Cassandra	25	1	week	1244.235	149.3082	119.81
135	Cassandra	25	1	month	4668.6125	280.11675	119.81
136	Cassandra	25	2	day	966.2175	101.4528375	119.81
137	Cassandra	25	2	week	4165.3725	124.961175	119.81
138	Cassandra	25	2	month	12323.49	184.85235	119.81
139	Cassandra	25	3	day	960.7025	84.06146875	119.81
140	Cassandra	25	3	week	5515.0475	137.8761875	119.81
141	Cassandra	25	3	month	22277.3425	278.4667813	119.81
142	Cassandra	25	4	day	4932.865	431.6256875	119.81
143	Cassandra	25	4	week	24511.1275	612.7781875	119.81
144	Cassandra	25	4	month	104458.095	1305.726188	119.81
145	Cassandra	50	1	day	357.4	150.108	83.78
146	Cassandra	50	1	week	1032.16	123.8592	83.78
147	Cassandra	50	1	month	3990.605	239.4363	83.78
148	Cassandra	50	2	day	1594.885	167.462925	83.78
149	Cassandra	50	2	week	6195.475	185.86425	83.78
150	Cassandra	50	2	month	17918.56	268.7784	83.78
151	Cassandra	50	3	day	902.115	78.9350625	83.78
152	Cassandra	50	3	week	5002.935	125.073375	83.78
153	Cassandra	50	3	month	22010.335	275.1291875	83.78
154	Cassandra	50	4	day	4885.37	427.469875	83.78
155	Cassandra	50	4	week	23693.195	592.329875	83.78
156	Cassandra	50	4	month	101632.27	1270.403375	83.78
157	Cassandra	75	1	day	424.395	178.2459	47.75
158	Cassandra	75	1	week	820.085	98.4102	47.75
159	Cassandra	75	1	month	3312.5975	198.75585	47.75
160	Cassandra	75	2	day	2223.5525	233.4730125	47.75
161	Cassandra	75	2	week	8225.5775	246.767325	47.75
162	Cassandra	75	2	month	23513.63	352.70445	47.75
163	Cassandra	75	3	day	843.5275	73.80865625	47.75
164	Cassandra	75	3	week	4490.8225	112.2705625	47.75
165	Cassandra	75	3	month	21743.3275	271.7915938	47.75

166	Cassandra	75	4	day	4837.875	423.3140625	47.75
167	Cassandra	75	4	week	22875.2625	571.8815625	47.75
168	Cassandra	75	4	month	98806.445	1235.080563	47.75
169	Cassandra	100	1	day	491.39	206.3838	11.73
170	Cassandra	100	1	week	608.01	72.9612	11.73
171	Cassandra	100	1	month	2634.59	158.0754	11.73
172	Cassandra	100	2	day	2852.22	299.4831	11.73
173	Cassandra	100	2	week	10255.68	307.6704	11.73
174	Cassandra	100	2	month	29108.7	436.6305	11.73
175	Cassandra	100	3	day	784.94	68.68225	11.73
176	Cassandra	100	3	week	3978.71	99.46775	11.73
177	Cassandra	100	3	month	21476.32	268.454	11.73
178	Cassandra	100	4	day	4790.38	419.15825	11.73
179	Cassandra	100	4	week	22057.33	551.43325	11.73
180	Cassandra	100	4	month	95980.62	1199.75775	11.73
181	Neo4j	0	1	day	2401.26	1008.5292	1968.3
182	Neo4j	0	1	week	6095.39	731.4468	1968.3
183	Neo4j	0	1	month	9094.42	545.6652	1968.3
184	Neo4j	0	2	day	2248.41	236.08305	1968.3
185	Neo4j	0	2	week	6462.28	193.8684	1968.3
186	Neo4j	0	2	month	11507.77	172.61655	1968.3
187	Neo4j	0	3	day	2417.06	211.49275	1968.3
188	Neo4j	0	3	week	3079.56	76.989	1968.3
189	Neo4j	0	3	month	7410.76	92.6345	1968.3
190	Neo4j	0	4	day	11828.87	1035.026125	1968.3
191	Neo4j	0	4	week	36806.39	920.15975	1968.3
192	Neo4j	0	4	month	119963.15	1499.539375	1968.3
193	Neo4j	25	1	day	1926.1675	808.99035	1480.87
194	Neo4j	25	1	week	4728.9475	567.4737	1480.87
195	Neo4j	25	1	month	7490.7575	449.44545	1480.87
196	Neo4j	25	2	day	2401.5925	252.1672125	1480.87
197	Neo4j	25	2	week	7415.8675	222.476025	1480.87
198	Neo4j	25	2	month	15921.1775	238.8176625	1480.87
199	Neo4j	25	3	day	2011.4825	176.0047188	1480.87
200	Neo4j	25	3	week	3316.73	82.91825	1480.87
201	Neo4j	25	3	month	10960.995	137.0124375	1480.87
202	Neo4j	25	4	day	10078.3975	881.8597813	1480.87
203	Neo4j	25	4	week	33133.805	828.345125	1480.87
204	Neo4j	25	4	month	114004.5925	1425.057406	1480.87
205	Neo4j	50	1	day	1451.075	609.4515	993.45
206	Neo4j	50	1	week	3362.505	403.5006	993.45
207	Neo4j	50	1	month	5887.095	353.2257	993.45
208	Neo4j	50	2	day	2554.775	268.251375	993.45
209	Neo4j	50	2	week	8369.455	251.08365	993.45
210	Neo4j	50	2	month	20334.585	305.018775	993.45
211	Neo4j	50	3	day	1605.905	140.5166875	993.45
212	Neo4j	50	3	week	3553.9	88.8475	993.45
213	Neo4j	50	3	month	14511.23	181.390375	993.45
214	Neo4j	50	4	day	8327.925	728.6934375	993.45
215	Neo4j	50	4	week	29461.22	736.5305	993.45
216	Neo4j	50	4	month	108046.035	1350.575438	993.45
217	Neo4j	75	1	day	975.9825	409.91265	506.02
218	Neo4j	75	1	week	1996.0625	239.5275	506.02
219	Neo4j	75	1	month	4283.4325	257.00595	506.02
220	Neo4j	75	2	day	2707.9575	284.3355375	506.02
221	Neo4j	75	2	week	9323.0425	279.691275	506.02
222	Neo4j	75	2	month	24747.9925	371.2198875	506.02
223	Neo4j	75	3	day	1200.3275	105.0286563	506.02
224	Neo4j	75	3	week	3791.07	94.77675	506.02
225	Neo4j	75	3	month	18061.465	225.7683125	506.02
226	Neo4j	75	4	day	6577.4525	575.5270938	506.02
227	Neo4j	75	4	week	25788.635	644.715875	506.02
228	Neo4j	75	4	month	102087.4775	1276.093469	506.02
229	Neo4j	100	1	day	500.89	210.3738	18.6

230	Neo4j	100	1	week	629.62	75.5544	18.6
231	Neo4j	100	1	month	2679.77	160.7862	18.6
232	Neo4j	100	2	day	2861.14	300.4197	18.6
233	Neo4j	100	2	week	10276.63	308.2989	18.6
234	Neo4j	100	2	month	29161.4	437.421	18.6
235	Neo4j	100	3	day	794.75	69.540625	18.6
236	Neo4j	100	3	week	4028.24	100.706	18.6
237	Neo4j	100	3	month	21611.7	270.14625	18.6
238	Neo4j	100	4	day	4826.98	422.36075	18.6
239	Neo4j	100	4	week	22116.05	552.90125	18.6
240	Neo4j	100	4	month	96128.92	1201.6115	18.6

Bibliography

[A+09] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[AAAS15] Federico Adrodegari, Andrea Alghisi, Marco Ardolino, and Nicola Saccani. From Ownership to Service-oriented Business Models: A Survey in Capital Goods Companies and a PSS Typology. *Procedia CIRP*, 30:245 – 250, 2015. 7th Industrial Product-Service Systems Conference - PSS, industry transformation for sustainability and business.

[AAF02] Rumaih M. Alrumaih and Mohammad A. Al-Fawzan. Time Series Forecasting Using Wavelet Denoising an Application to Saudi Stock Index. *Journal of King Saud University - Engineering Sciences*, 14(2):221 – 233, 2002.

[ABC+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, and et al. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA, 2016. USENIX Association.

[Add17] Paul S Addison. *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. CRC press, 2017.

[AFGM+15] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, Fourthquarter 2015.

[AFS93] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *Foundations of Data Organization and Algorithms*, pages 69–84, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[AJB97] Henrik André-Jönsson and Dushan Z. Badal. Using signature files for querying time-series data. In *Principles of Data Mining and Knowledge Discovery*, pages 211–220, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[AM19] Omar Saeed Al-Mushayt. Automating E-Government Services With Artificial Intelligence. *IEEE Access*, 7:146821–146829, 2019.

[Ama20] Amazon. AWS resource & custom metrics monitoring, February 2020. Accessed: 2020-04-28.

[AMCD20] Joseph Azar, Abdallah Makhoul, Raphaël Couturier, and Jacques Demerjian. Robust IoT time series classification with data compression and deep learning. *Neurocomputing*, 2020.

[AN98] Mehdi Azzouzi and Ian T Nabney. Analysing time series structure with Hidden Markov Models. In *Neural Networks for Signal Processing VIII. Proceedings of the 1998 IEEE Signal Processing Society Workshop (Cat. No. 98TH8378)*, pages 402–408. IEEE, 1998.

[ANM] ANMPO (Advanced Manufacturing National Program Office). MANUFACTURING.GOV: A national advanced manufacturing portal. `https://www.manufacturing.gov/reports`. Accessed: 2020-04-02.

[Apa19] Apache Hadoop. Apache Hadoop. `https://hadoop.apache.org/`, 2019. Accessed: 2020-01-31.

[AR04] Edgar Acuña and Caroline Rodriguez. The Treatment of Missing Values and its Effect on Classifier Accuracy. In *Classification, Clustering, and Data Mining Applications*, pages 639–647, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Ast12] Viktor P. Astakhov. *Design of Experiment Methods in Manufacturing: Basics and Practical Applications", bookTitle="Statistical and Computational Techniques in Manufacturing*, pages 1–54. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[ASW15] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering – A decade review. *Information Systems*, 53:16 – 38, 2015.

[Bao08] Depei Bao. A generalized model for financial time series representation and prediction. *Applied Intelligence*, 29(1):1–11, Aug 2008.

[BB16] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[BBHE18] Marouane Birjali, Abderrahim Beni-Hssane, and Mohammed Erritali. Evaluation of high-level query languages based on MapReduce in Big Data. *Journal of Big Data*, 5(1):36, 2018.

[BC94] Donald J. Berndt and James Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, pages 359–370. AAAI Press, 1994.

[BCSMAB13] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. A review of feature selection methods on synthetic data. *Knowledge and Information Systems*, 34(3):483–519, Mar 2013.

[BCW+11] Ergin Bayrak, John P Conley, Simon Wilkie, et al. The economics of cloud computing. *The Korean Economic Review*, 27(2):203–230, 2011.

[BGCML20] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. A review on outlier/anomaly detection in time series data. *arXiv preprint arXiv:2002.04236*, 2020.

[BH95] William L Briggs and Van Emden Henson. *The DFT: an owners' manual for the discrete Fourier transform*, volume 45. Siam, 1995.

[BKF17] Andreas Bader, Oliver Kopp, and Michael Falkenthal. Survey and comparison of open source time series databases. *Datenbanksysteme für Business, Technologie und Web (BTW 2017)-Workshopband*, 2017.

[BKNS00] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, page 93–104, New York, NY, USA, 2000. Association for Computing Machinery.

[BLB+17] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, May 2017.

[BLJZ00] Bin Li, Lixiang Tan, Jinsong Zhang, and Zhenquan Zhuang. Using fuzzy neural network clustering algorithm in the symbolization of time series. In *IEEE APCCAS 2000. 2000 IEEE Asia-Pacific Conference on Circuits and Systems. Electronic Communication Systems. (Cat. No.00EX394)*, pages 379–382, 2000.

[BMN15] Hans-Jörg Bullinger, Thomas Meiren, and Rainer Nägele. Smart services in manufacturing companies. In *23rd International Conference on Production Research*, pages 7–13, 2015.

[BRG+12] Jürgen Bernard, Tobias Ruppert, Oliver Goroll, Thorsten May, and Jörn Kohlhammer. Visual-interactive preprocessing of time series data. In *Proceedings of SIGRAD 2012; Interactive Visual Analysis of Data; November 29-30; 2012; Växjö; Sweden*, number 081, pages 39–48. Linköping University Electronic Press, 2012.

[BRK+06] Anthony Bagnall, Chotirat Ann Ratanamahatana, Eamonn Keogh, Stefano Lonardi, and Gareth Janacek. A Bit Level Representation for Time Series Data Mining with Shape Based Similarity. *Data Mining and Knowledge Discovery*, 13(1):11–40, Jul 2006.

[Bro17] Jason Brownlee. *Introduction to time series forecasting with python: how to prepare data and develop models to predict the future*. Machine Learning Mastery, 2017.

[BWSR13] Maxim Buevich, Anne Wright, Randy Sargent, and Anthony Rowe. Respawn: A Distributed Multi-resolution Time-Series Datastore. In *2013 IEEE 34th Real-Time Systems Symposium*, pages 288–297, Dec 2013.

[C+15] François Chollet et al. Keras. `https://keras.io/`, 2015. Accessed: 2020-04-24.

[Cas19] Apache Cassandra. The Cassandra Query Language (CQL). `http://cassandra.apache.org/doc/latest/cql/`, 2019. Accessed: 2020-03-03.

[CCW16] José M Cavanillas, Edward Curry, and Wolfgang Wahlster. *New horizons for a data-driven economy: a roadmap for usage and exploitation of big data in Europe*. Springer, 2016.

[CDG+08] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.*, 26(2), June 2008.

[CFM+04] Alfredo Cuzzocrea, Filippo Furfaro, Elio Masciari, Domenico Saccà, and Cristina Sirangelo. Approximate query answering on sensor network data streams. *GeoSensor Networks*, 49, 2004.

[CFMS04] Alfredo Cuzzocrea, Filippo Furfaro, Giuseppe M. Mazzeo, and Domenico Saccà. A Grid Framework for Approximate Aggregate Query Answering on Summarized Sensor Network Readings. In *On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops*, pages 144–153, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[CFR13] Alfredo Cuzzocrea, Giancarlo Fortino, and Omer Rana. Managing Data and Processes in Cloud-Enabled Large-Scale Sensor Networks: State-of-the-Art and Future Research Directions. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 583–588, May 2013.

[CH18] Alem Colaković and Mesud Hadzialić. Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. *Computer Networks*, 144:17 – 39, 2018.

[Che14] Jehanzeb R Cheema. A review of missing data handling methods in education research. *Review of Educational Research*, 84(4):487–508, 2014.

[CHT09] Alok Kumar Choudhary, Jenny A Harding, and Manoj Kumar Tiwari. Data mining in manufacturing: a review based on the kind of knowledge. *Journal of Intelligent Manufacturing*, 20(5):501, 2009.

[Cis20] Cisco. Cisco Annual Internet Report (2018–2023) White Paper. `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`, 2020. Accessed: 2020-03-22.

[Ciu18] Dan Ciuriak. The economics of data: implications for the data-driven economy. In *Data Governance in the Digital Age*, chapter 8. Centre for International Governance Innovation, 2018.

[CKH+15] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The UCR Time Series Classification Archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

[CKLF16] Maximilian Christ, Andreas W Kempa-Liehr, and Michael Feindt. Distributed and parallel time series feature extraction for industrial big data applications. *arXiv preprint arXiv:1610.07717*, 2016.

[CKMP02] Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. *ACM Trans. Database Syst.*, 27(2):188–228, June 2002.

[CMJB18] Stöhr Carsten, Janssen Monika, Niemann Jörg, and Reich Benedikt. Smart Services. *Procedia - Social and Behavioral Sciences*, 238:192 – 198, 2018. Challenges and Innovation in Management and Entrepreneurship.

[CML14] Min Chen, Shiwen Mao, and Yunhao Liu. Big Data: A Survey. *Mobile Networks and Applications*, 19(2):171–209, Apr 2014.

[CN04] Yuhan Cai and Raymond Ng. Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, page 599–610, New York, NY, USA, 2004. Association for Computing Machinery.

[COO05] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and Fast Similarity Search for Moving Object Trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 491–502, New York, NY, USA, 2005. ACM.

[CPZH19] Shuang Cai, Ahmet Palazoglu, Laibin Zhang, and Jinqiu Hu. Process alarm prediction using deep learning and word embedding methods. *ISA Transactions*, 85:274 – 283, 2019.

[Cut14] Thomas R Cutler. The internet of manufacturing things. *Ind. Eng*, 46(8):37–41, 2014.

[Cuz15] Alfredo Cuzzocrea. Temporal Aspects of Big Data Management: State-of-the-Art Analysis and Future Research Directions. In *2015 22nd International Symposium on Temporal Representation and Reasoning (TIME)*, pages 180–185, 2015.

[Dau92] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, USA, 1992.

[DCL18] Ali Davoudian, Liu Chen, and Mengchi Liu. A Survey on NoSQL Stores. *ACM Comput. Surv.*, 51(2), April 2018.

[Den14] Houtao Deng. Interpreting tree ensembles with intrees. *arXiv preprint arXiv:1408.5456*, 2014.

[DEP+12] Jim Davis, Thomas Edgar, James Porter, John Bernaden, and Michael Sarli. Smart manufacturing, manufacturing intelligence and demand-dynamic performance. *Computers & Chemical Engineering*, 47:145 – 156, 2012. FOCAPO 2012.

[DF14] Ted Dunning and Ellen Friedman. *Time Series Databases: New Ways to Store and Access Data*. O'Reilly Media, Incorporated, 2014.

[DG08] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[DG13] Xuewu Dai and Zhiwei Gao. From Model, Signal to Knowledge: A Data-Driven Perspective of Fault Detection and Diagnosis. *IEEE Transactions on Industrial Informatics*, 9(4):2226–2238, 2013.

[Dha12] Subhankar Dhar. From outsourcing to Cloud computing: evolution of IT services. *Management Research Review*, 2012.

[DLM+98] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule Discovery from Time Series. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, KDD'98, page 16–22. AAAI Press, 1998.

[DRTB14] Houtao Deng, George Runger, Eugene Tuv, and Wade Bannister. CBC: An Associative Classifier with a Small Number of Rules. *Decis. Support Syst.*, 59:163–170, March 2014.

[Dru20] Apache Druid. Apache Druid is a high performance real-time analytics database. `https://druid.apache.org/`, 2020. Accessed: 2020-04-27.

[DWC10] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud Computing: Issues and Challenges. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 27–33, 2010.

[DWS+19] Xiaoou Ding, Hongzhi Wang, Jiaxuan Su, Zijue Li, Jianzhong Li, and Hong Gao. Cleanits: A Data Cleaning System for Industrial Time Series. *Proc. VLDB Endow.*, 12(12):1786–1789, August 2019.

[EA12a] Philippe Esling and Carlos Agon. Time-Series Data Mining. *ACM Comput. Surv.*, 45(1), December 2012.

[EA12b] Peter C. Evans and Marco Annunziata. Industrial Internet: Pushing the Boundaries of Minds and Machines. Technical report, General Electrics, November 2012.

[EEC+09] Hazem Elmeleegy, Ahmed K. Elmagarmid, Emmanuel Cecchet, Walid G. Aref, and Willy Zwaenepoel. Online Piece-Wise Linear Approximation of Numerical Streams with Precision Guarantees. *Proc. VLDB Endow.*, 2(1):145–156, August 2009.

[EFF16] EFFRA (European Factories of the Future Research Association). Factories 4.0 and Beyond. Technical report, 2016.

[Eur14] European Commission. Towards a thriving data-driven economy. Technical report, 2014.

[Eur19a] European Commission. IICT Innovation for Manufacturing SMEs. Technical report, 2019.

[Eur19b] European Commission. Smart Manufacturing. Technical report, 2019.

[Eva11] D Evans. The Internet of Things: How the Next Evolution of the Internet is Changing Everything. *Cisco Internet Business Solutions Group (IBSG)*, 1:1–11, 01 2011.

[Fay96] Usama Fayyad. Data mining and knowledge discovery: making sense out of data. *IEEE Expert*, 11(5):20–25, 1996.

[FGdCG16] Elaine R. Faria, Isabel J. C. R. Gonçalves, André C. P. L. F. de Carvalho, and João Gama. Novelty detection in data streams. *Artificial Intelligence Review*, 45(2):235–269, Feb 2016.

[FJ14] Ben D. Fulcher and Nick S. Jones. Highly Comparative Feature-Based Time-Series Classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, Dec 2014.

[FPG03] E. Fink, K. B. Pratt, and H. S. Gandhi. Indexing of time series by major minima and maxima. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, volume 3, pages 2332–2335 vol.3, 2003.

[FPSS96] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37, Mar. 1996.

[FSBR17] P. Ferrari, E. Sisinni, D. Brandão, and M. Rocha. Evaluation of communication latency in industrial IoT applications. In *2017 IEEE International Workshop on Measurement and Networking (M N)*, pages 1–6, Sep. 2017.

[FTLN02] Fu-lai Chung, Tak-chung Fu, R. Luk, and V. Ng. Evolutionary time series segmentation for stock data mining. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 83–90, Dec 2002.

[FU95] Usama Fayyad and Ramasamy Uthurusamy, editors. *KDD'95: Proceedings of the First International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1995.

[Fu11] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164 – 181, 2011.

[Ful17] Ben Fulcher. Feature-based time-series analysis. *CoRR: Computing Research Repository*, 09 2017.

[Gar19] Gartner. Gartner Forecasts Worldwide Public Cloud Revenue. `https://www.gartner.com/en/newsroom/press-releases/2019-11-13-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2020`, 2019. Accessed: 2020-03-22.

[GBB18] Mouzhi Ge, Hind Bangui, and Barbora Buhnova. Big Data for Internet of Things: A Survey. *Future Generation Computer Systems*, 87:601 – 614, 2018.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GBMP13] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[GGB12] Juozas Gordevičius, Johann Gamper, and Michael Böhlen. Parsimonious temporal aggregation. *The VLDB Journal*, 21(3):309–332, Jun 2012.

[GGL03] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, page 29–43, New York, NY, USA, 2003. Association for Computing Machinery.

[GHC18] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent Advances in Open Set Recognition: A Survey. *arXiv preprint arXiv:1811.08581*, 2018.

[GHMP09] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. The Cost of a Cloud: Research Problems in Data Center Networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, December 2009.

[GJZ08] Chonghui Guo, Hongfeng Jia, and Na Zhang. Time Series Clustering Based on ICA for Stock Data Analysis. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4, 2008.

[GLH15] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2015.

[GLH16] Salvador García, Julián Luengo, and Francisco Herrera. Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. *Knowledge-Based Systems*, 98:1 – 29, 2016.

[GLT01] C. Lee Giles, Steve Lawrence, and Ah Chung Tsoi. Noisy Time Series Prediction using Recurrent Neural Networks and Grammatical Inference. *Machine Learning*, 44(1):161–183, 2001.

[GMM18] Federico Giannoni, Marco Mancini, and Federico Marinelli. Anomaly Detection Models for IoT Time Series Data. *arXiv preprint arXiv:1812.00890*, 2018.

[Goo19a] Google. Google Artificial Intelligence Platform. `https://cloud.google.com/ai-platform/`, 2019. Accessed: 2020-01-21.

[Goo19b] Google. Google Cloud Platform. `https://cloud.google.com/compute/pricing?hl=es-419#disk`, 2019. Accessed: 2019-10-30.

[GPC] Esther Gómez Pérez-Cuadrado. Plan Made in China 2025. `https://www.icex.es/icex/es/navegacion-principal/todos-nuestros-servicios/informacion-de-mercados/paises/navegacion-principal/el-mercado/estudios-informes/DOC2016671546.html?idPais=CN`. Accessed: 2020-04-02.

[GPy16] GPyOpt authors. GPyOpt: A Bayesian Optimization framework in Python. `http://github.com/SheffieldML/GPyOpt`, 2016.

[Gro95] Peter P. Groumpos. The challenge of Intelligent Manufacturing Systems (IMS): the European IMS information event. *Journal of Intelligent Manufacturing*, 6(1):67–77, 1995.

[GSRP+18] Vicente García, J. Salvador Sánchez, Luis Alberto Rodríguez-Picón, Luis Carlos Méndez-González, and Humberto de Jesús Ochoa-Domínguez. Using regression models for predicting the product quality in a tubing extrusion process. *Journal of Intelligent Manufacturing*, Mar 2018.

[GT19] Adrian Iustin Georgevici and Marius Terblanche. Neural networks and deep learning: a brief introduction. *Intensive Care Medicine*, 45(5):712–714, May 2019.

[GvB+14] João Gama, Indrundefined Žliobaitundefined, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.*, 46(4), March 2014.

[Hal00] Mark A. Hall. Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 359–366, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[Har90] Andrew C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1990.

[Has11] Hash M. Hashemian. State-of-the-Art Predictive Maintenance Techniques. *IEEE Transactions on Instrumentation and Measurement*, 60(1):226–236, 2011.

[HBM01] Bernard Hugueney and Bernadette Bouchon-Meunier. Time-Series Segmentation and Symbolic Representation, from Process-Monitoring to Data-Mining. In *Computational Intelligence. Theory and Applications*, pages 118–123, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[HC18] Mehadi Hassen and Philip K Chan. Learning a neural-network-based representation for open set recognition. *arXiv preprint arXiv:1802.04365*, 2018.

[HH01] Georges Hebrail and Bernard Hugueney. Symbolic representation of long time-series. In *Proceedings of the Applied Stochastic Models and Data Analysis Conference*, pages 537–542, 2001.

[Hin92] Geoffrey E. Hinton. How Neural Networks Learn from Experience. *Scientific American*, 267(3):144–151, 1992.

[HLA+15] Adriana Horelu, Catalin Leordeanu, Elena Apostol, Dan Huru, Mariana Mocanu, and Valentin Cristea. Forecasting Techniques for Time Series from Sensor Data. In *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 261–264, Sep. 2015.

[HNF08] Ville Hautamaki, Pekka Nykanen, and Pasi Franti. Time-series clustering by approximate prototypes. In *2008 19th International Conference on Pattern Recognition*, pages 1–4, Dec 2008.

[Hoc98] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[HPK11] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 3rd edition edition, 2011.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[HSSK05] J. A. Harding, M. Shahbaz, Srinivas, and A. Kusiak. Data Mining in Manufacturing: A Review. *Journal of Manufacturing Science and Engineering*, 128(4):969–976, 12 2005.

[Hsu17] Daniel Hsu. Time series compression based on adaptive piecewise recurrent autoencoder. *arXiv preprint arXiv:1707.07961*, 2017.

[HV08] M. Hubert and E. Vandervieren. An adjusted boxplot for skewed distributions. *Computational Statistics & Data Analysis*, 52(12):5186 – 5201, 2008.

[HWJ03] Dennis E Hinkle, William Wiersma, and Stephen G Jurs. *Applied statistics for the behavioral sciences*, volume 663. Houghton Mifflin College Division, 2003.

[HXD03] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641 – 1650, 2003.

[HY99] Yun-Wu Huang and Philip S. Yu. Adaptive Query Processing for Time-Series Data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, page 282–286, New York, NY, USA, 1999. Association for Computing Machinery.

[HYZ+17] Li Huang, Yongbiao Yang, Honglei Zhao, Xudong Wang, and Hongjuan Zheng. Time series modeling and filtering method of electric power load stochastic noise. *Protection and Control of Modern Power Systems*, 2(1):25, 2017.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[IDC12] IDC. The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. Technical report, 2012.

[IDC19] IDC and The Lisbon Council. Data as the Engine of Europe's Digital Future: Second Report on Policy Conclusions. Technical report, European Commission, 2019.

[IFFW+19] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, Jul 2019.

[IIC] IIC (Industrial Internet Consortium). Industrial Internet Consortium: Technical Papers, Publications and White Papers. `https://www.iiconsortium.org/white-papers.htm`. Accessed: 2020-04-02.

[IMDK19] Rahat Iqbal, Tomasz Maniak, Faiyaz Doctor, and Charalampos Karyotis. Fault Detection and Isolation in Industrial Processes Using Deep Learning Approaches. *IEEE Transactions on Industrial Informatics*, 15(5):3077–3084, May 2019.

[IMMR16] Rizaldy G Ignacio, Ken Maycock, Gary F Murtagh, and Shay Roe. Increased database performance via migration of data to faster storage, November 15 2016. US Patent 9,495,396.

[Inf19a] InfluxData Inc. Influx Query Language (InfluxQL) reference. `https://docs.influxdata.com/influxdb/v1.7/query_language/spec/`, 2019. Accessed: 2020-03-03.

[Inf19b] InfluxData Inc. InfluxDB. `https://www.influxdata.com/`, 2019. Accessed: 2020-03-21.

[ISD] ISDP (Institute for Security & Development Policy). Made in China 2025 Backgrounder. `https://isdp.eu/publication/made-china-2025/`. Accessed: 2020-04-02.

[JAF+06] S. R. Jeffery, G. Alonso, M. J. Franklin, Wei Hong, and J. Widom. A Pipelined Framework for Online Cleaning of Sensor Data Streams. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 140–140, April 2006.

[JB12] Dieter William Joenssen and Udo Bankhofer. Hot deck methods for imputing missing data. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 63–75. Springer, 2012.

[JHGJ11] Jing Han, Haihong E, Guan Le, and Jian Du. Survey on NoSQL database. In *2011 6th International Conference on Pervasive Computing and Applications*, pages 363–366, 2011.

[JPT17] S. K. Jensen, T. B. Pedersen, and C. Thomsen. Time Series Management Systems: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2581–2600, Nov 2017.

[KA99] Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337)*, pages 126–133, March 1999.

[Kaf86] Karen Kafadar. Gaussian white-noise generation for digital signal synthesis. *IEEE Transactions on Instrumentation and Measurement*, IM-35(4):492–495, Dec 1986.

[Kai15] KairosDB. KairosDB Fast Time Series Database on Cassandra. `https://kairosdb.github.io/`, 2015. Accessed: 2020-04-27.

[KAV15] Ina Khandelwal, Ratnadip Adhikari, and Ghanshyam Verma. Time Series Forecasting Using Hybrid ARIMA and ANN Models Based on DWT Decomposition. *Procedia Computer Science*, 48:173 – 179, 2015. International Conference on Computer, Communication and Convergence (ICCC 2015).

[KBM15] Mourad Khayati, Michael H. Böhlen, and Philippe Cudré Mauroux. Using Lowly Correlated Time Series to Recover Missing Values in Time Series: A Comparison Between SVD and CD. In *Advances in Spatial and Temporal Databases*, pages 237–254, Cham, 2015. Springer International Publishing.

[KBSM14] Yanfei Kang, Danijel Belušić, and Kate Smith-Miles. Detecting and Classifying Events in Noisy Time Series. *Journal of the Atmospheric Sciences*, 71(3):1090–1104, 2014.

[KBT11] Gulser Koksal, Inci Batmaz, and Murat Caner Testik. A review of data mining applications for quality improvement in manufacturing industry. *Expert Systems with Applications*, 38(10):13448 – 13467, 2011.

[KCH+03] Won Kim, Byoung-Ju Choi, Eui-Kyeong Hong, Soo-Kyung Kim, and Doheon Lee. A Taxonomy of Dirty Data. *Data Mining and Knowledge Discovery*, 7(1):81–99, Jan 2003.

[KCPM01] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, 3(3):263–286, Aug 2001.

[Keo97a] Eamonn Keogh. A fast and robust method for pattern matching in time series databases. *Proceedings of WUSS*, 97(1):99, 1997.

[Keo97b] Eamonn Keogh. Fast similarity search in the presence of longitudinal scaling in time series databases. In *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*, pages 578–584, Nov 1997.

[KFP15] Yannis Katsis, Yoav Freund, and Yannis Papakonstantinou. Combining Databases and Signal Processing in Plato. In *CIDR*, 2015.

[KGHS14] István Kiss, Béla Genge, Piroska Haller, and Gheorghe Sebestyén. Data clustering-based anomaly detection in industrial control systems. In *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 275–281, 2014.

[KHDK18] E. Kučera, O. Haffner, P. Drahoš, and Š. Kozák. Emerging technologies for Industry 4.0: OPC unified architecture and virtual/mixed reality. In *AIFICT 2018: 1st International conference on applied informatics in future ICT*, pages 57–62, May 2018.

[Kim14] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[KJF97] Flip Korn, H. V. Jagadish, and Christos Faloutsos. Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, SIGMOD '97, page 289–300, New York, NY, USA, 1997. Association for Computing Machinery.

[KKZK12] Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan. Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In *2012 10th International Conference on Frontiers of Information Technology*, pages 257–260, Dec 2012.

[KL05a] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and Information Systems*, 8(2):154–177, Aug 2005.

[KL05b] T Kohler and Dirk Lorenz. A comparison of denoising methods for one dimensional time series. *Bremen, Germany, University of Bremen*, 131, 2005.

[KLM96] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[KLTCM20] Mourad Khayati, Alberto Lerner, Zakhar Tymchenko, and Philippe Cudré-Mauroux. Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series. *Proc. VLDB Endow.*, 13(5):768–782, January 2020.

[KM06] Lukasz A. Kurgan and Petr Musilek. A survey of Knowledge Discovery and Data Mining process models. *The Knowledge Engineering Review*, 21(1):1–24, 2006.

[KMB+18] Moritz Kulessa, Alejandro Molina, Carsten Binnig, Benjamin Hilprecht, and Kristian Kersting. Model-based Approximate Query Processing. *CoRR*, abs/1811.06224, 2018.

[Kou16] Jayanth Koushik. Understanding convolutional neural networks. *arXiv preprint arXiv:1605.09081*, 2016.

[KPC14] Srdjan Krčo, Boris Pokrić, and Francois Carrez. Designing IoT architecture(s): A European perspective. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 79–84, March 2014.

[KRH+14] Henning Kagermann, Frank Riemensperger, Dirk Hoke, Johannes Helbig, Dirk Stocksmeier, Wolfgang Wahlster, August-Wilhelm Scheer, and Dieter Schweer. Smart Service Welt: Recommendations for the Strategic Initiative Web-based Services for Businesses. `https://www.acatech.de/wp-content/uploads/2014/03/BerichtSmartService_final_barrierefrei_ENG.pdf`, 2014.

[KRZ+20] W.Z. Khan, M.H. Rehman, H.M. Zangoti, M.K. Afzal, N. Armi, and K. Salah. Industrial internet of things: Recent advances, enabling technologies and open challenges. *Computers & Electrical Engineering*, 81:106522, 2020.

[KS14] Maciej Krawczak and Grażyna Szkatuła. An approach to dimensionality reduction in time series. *Information Sciences*, 260:15 – 36, 2014.

[KU02] K. Kawagoe and T. Ueda. A similarity search method of time series data with combination of Fourier and wavelet transforms. In *Proceedings Ninth International Symposium on Temporal Representation and Reasoning*, pages 86–92, 2002.

[Kus] Andrew Kusiak. *Intelligent Manufacturing Systems*. 10. Prentice Hall Press, 200 Old Tappan Road, Old Tappan, NJ 07675, USA, 1.

[Kus90] Andrew Kusiak. Editorial. *Journal of Intelligent Manufacturing*, 1(1):i–i, 1990.

[Kus17] Andrew Kusiak. Smart Manufacturing Must Embrace Big Data. *Nature News*, 544:23–25, 04 2017.

[Kus18] Andrew Kusiak. Smart manufacturing. *International Journal of Production Research*, 56(1-2):508–517, 2018.

[KWH13] Henning Kagermann, Wolfgang Wahlster, and Johannes Helbig. Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group. Technical report, 2013.

[LABTS14] Rocco Langone, Carlos Alzate, Abdellatif Bey-Temsamani, and Johan A. K. Suykens. Alarm prediction in industrial machines using autoregressive LS-SVM models. In *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 359–364, Dec 2014.

[LAL19] Minseok Lee, Jihoon An, and Younghee Lee. Missing-value imputation of continuous missing based on deep imputation network using correlations among multiple iot data streams in a smart space. *IEICE TRANSACTIONS on Information and Systems*, 102(2):289–298, 2019.

[Lan18] Doug Laney. 3-D Data Management: Controlling Data Volume, Velocity and Variety. `https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf`, February 2018. Accessed: 2020-01-30.

[LBP20] Chunbin Lin, Etienne Boursier, and Yannis Papakonstantinou. Plato: Approximate Analytics over Compressed Time Series with Tight Deterministic Error Guarantees. *Proc. VLDB Endow.*, 13(7):1105–1118, March 2020.

[LCL99] R. H. Lesch, Y. Caille, and D. Lowe. Component analysis in financial time series. In *Proceedings of the IEEE/IAFE 1999 Conference on Computational Intelligence for Financial Engineering (CIFEr) (IEEE Cat. No.99TH8408)*, pages 183–190, 1999.

[LCST09] Chien-Cheng Lee, Yu-Chun Chiang, Cheng-Yuan Shih, and Chun-Li Tsai. Noisy time series prediction using M-estimator based robust radial basis function neural networks with growing and pruning techniques. *Expert Systems with Applications*, 36(3, Part 1):4717 – 4724, 2009.

[Lia05] T. Warren Liao. Clustering of time series data a survey. *Pattern Recognition*, 38(11):1857 – 1874, 2005.

[LKL03] Sangjun Lee, Dongseop Kwon, and Sukho Lee. Dimensionality Reduction for Indexing Time Series Based on the Minimum Distance. *J. Inf. Sci. Eng.*, 19:697–711, 2003.

[LKLC03] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, page 2–11, New York, NY, USA, 2003. Association for Computing Machinery.

[LL16] Peter Laurinec and Mária Lucká. Comparison of representations of time series for clustering smart meter data. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, 2016.

[LLTX09] Mingwei Leng, Xinsheng Lai, Guolv Tan, and Xiaohui Xu. Time series representation for anomaly detection. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 628–632, 2009.

[LMPF09] Lei Li, James McCann, Nancy S. Pollard, and Christos Faloutsos. DynaMMo: Mining and Summarization of Coevolving Sequences with Missing Values. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 507–516, New York, NY, USA, 2009. Association for Computing Machinery.

[LOD18] Liangzhi Li, Kaoru Ota, and Mianxiong Dong. Deep Learning for Smart Industry: Efficient Manufacture Inspection System With Fog Computing. *IEEE Transactions on Industrial Informatics*, 14(10):4665–4673, Oct 2018.

[LQPH13] Hongfei Li, Buyue Qian, Dhaivat Parikh, and Arun Hampapur. Alarm prediction in large-scale sensor networks — A case study in railroad. In *2013 IEEE International Conference on Big Data*, pages 7–14, Oct 2013.

[LRU19] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive data sets*. Cambridge university press, 2019.

[LSN19] Jinwei Liu, Haiying Shen, and Husnu S. Narman. Popularity-Aware Multi-Failure Resilient and Cost-Effective Replication for High Data Durability in Cloud Storage. *IEEE Transactions on Parallel and Distributed Systems*, 30(10):2355–2369, Oct 2019.

[Lüt11] Helmut Lütkepohl. *Vector Autoregressive Models*, pages 1645–1647. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[LW02] Andy Liaw and Matthew Wiener. Classification and Regression by randomForest. *R News*, 2(3):18–22, 2002.

[LY19] Rui Liu and Jun Yuan. Benchmarking Time Series Databases with IoTDB-Benchmark for IoT Scenarios, 2019. Pre-print availiable at https://arxiv.org/abs/1901.08304v3.

[LYC98] Chung-Sheng Li, Philip S. Yu, and Vittorio Castelli. MALM: A Framework for Mining Sequence Database at Multiple Abstraction Levels. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, page 267–272, New York, NY, USA, 1998. Association for Computing Machinery.

[Man14] Manish Gupta and Jing Gao and Charu C. Aggarwal and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.

[MCB+11] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. Big Data: The Next Frontier for Innovation, Competition, and Productivity, June 2011.

[MFP+19] Sergio Di Martino, Luca Fiadone, Adriano Peron, Alberto Riccabone, and Vincenzo Norman Vitale. Industrial Internet of Things: Persistence for Time Series with NoSQL Databases. In *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 340–345, June 2019.

[MG11] Peter Mell and Tim Grance. The NIST definition of cloud computing. 2011.

[MG15] Morteza Mashayekhi and Robin Gras. Rule extraction from random forest: the RF+ HC methods. In *Canadian Conference on Artificial Intelligence*, pages 223–237. Springer, 2015.

[MI17] James Moyne and Jimmy Iskandar. Big data analytics for smart manufacturing: Case studies in semiconductor manufacturing. *Processes*, 5(3):39, 2017.

[Min20] LLC Minitab. Minitab. `https://www.minitab.com/es-mx/`, 2020. Accessed: 2020-03-10.

[Mit10] Theophano Mitsa. *Temporal Data Mining*. Chapman & Hall/CRC, 1st edition, 2010.

[MKLR19] Brenno C. Menezes, Jeffrey D. Kelly, Adriano G. Leal, and Galo C. Le Roux. Predictive, Prescriptive and Detective Analytics for Smart Manufacturing in the Information Age. *IFAC-PapersOnLine*, 52(1):568 – 573, 2019. 12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems DYCOPS 2019.

[MLW04] Vasileios Megalooikonomou, Guo Li, and Qiang Wang. A Dimensionality Reduction Technique for Efficient Similarity Analysis of Time Series Databases. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, page 160–161, New York, NY, USA, 2004. Association for Computing Machinery.

[MMF10] Gonzalo Mariscal, Óscar Marbán, and Covadonga Fernández. A survey of data mining and knowledge discovery process models and methodologies. *The Knowledge Engineering Review*, 25(2):137–166, 2010.

[MML16] Usue Mori, Alexander Mendiburu, and Jose A. Lozano. Similarity Measure Selection for Clustering Time Series Databases. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):181–195, Jan 2016.

[Mon08] MonetDB. MonetDB An Open-Source Database System. `https://www.monetdb.org/`, 2008. Accessed: 2020-04-27.

[Mon19a] MongoDB. MongoDB. `https://www.mongodb.com/`, 2019. Accessed: 2020-03-21.

[Mon19b] MongoDB. MongoDB Documentation Query Documents. `https://docs.mongodb.com/manual/tutorial/query-documents/`, 2019. Accessed: 2020-03-03.

[MSBB+15] Steffen Moritz, Alexis Sardá, Thomas Bartz-Beielstein, Martin Zaefferer, and Jörg Stork. Comparison of different methods for univariate time series imputation in R. *arXiv preprint arXiv:1510.03924*, 2015.

[MSDA19] Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access*, 7:1991–2005, 2019.

[MTR+16] Pankaj Malhotra, Vishnu TV, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder, 2016.

[MW01] Polly Wan Po Man and Man Hon Wong. Efficient and Robust Feature Extraction and Pattern Matching of Time Series by a Lattice Structure. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, page 271–278, New York, NY, USA, 2001. Association for Computing Machinery.

[MW15] Nathan Marz and James Warren. *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.

[MY12] Supavit Muangjaroen and Thaweesak Yingthawornsuk. A Study of Noise Reduction in Speech Signal using FIR Filtering. *International Conference on Advances in Electrical and Electronics Engineering (ICAEEE'2012)*, pages 51–54, 2012.

[NAM01] Alex Nanopoulos, Rob Alcock, and Yannis Manolopoulos. Information Processing and Technology. chapter Feature-based Classification of Time-series Data, pages 49–61. Nova Science Publishers, Inc., Commack, NY, USA, 2001.

[Nat12] National Science and Technology Council. A National Strategic Plan for Advanced Manufacturing. Technical report, National Science and Technology Council (NSTC), February 2012.

[NBI15] Mikel Niño, José Miguel Blanco, and Arantza Illarramendi. Business understanding, challenges and issues of big data analytics for the servitization of a capital equipment manufacturer. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 1368–1377, Oct 2015.

[Neo19a] Neo4j. Cypher Query Language. `https://neo4j.com/developer/cypher-query-language/`, 2019. Accessed: 2020-03-03.

[Neo19b] Neo4j. Neo4J. `https://neo4j.com/`, 2019. Accessed: 2020-03-21.

[NHMG20] Marco Nardello, Shengnan Han, Charles Møller, and John Gøtze. Incorporating process and data heterogeneity in enterprise architecture: Extended AMA4EA in an international manufacturing company. *Computers in Industry*, 115:103178, 2020.

[Nic11] Kristin K Nicodemus. Letter to the editor: On the stability and ranking of predictors from random forest variable importance measures. *Briefings in bioinformatics*, 12(4):369–373, 2011.

[Niñ17] Mikel Niño. *A proposal for the management of data-driven services in Smart Manufacturing scenarios*. PhD thesis, University of the Basque Country UPV/EHU, 2017.

[NIS17] NIST (National Institute of Standards and Technology). Smart Manufacturing Operations Planning and Control Program, March 2017. Accessed: 2020-05-01.

[NLM11] K. Nose-Filho, A. D. P. Lotufo, and C. R. Minussi. Preprocessing data for short-term load forecasting with a general regression neural network and a moving average filter. In *2011 IEEE Trondheim PowerTech*, pages 1–7, 2011.

[NLMBC17] Kevin Nagorny, Pedro Lima-Monteiro, Jose Barata, and Armando Walter Colombo. Big data analysis in smart manufacturing: A review. *International Journal of Communications, Network and System Sciences*, 10(3):31–58, 2017.

[NR07] Vit Niennattrakul and Chotirat Ann Ratanamahatana. On Clustering Multimedia Time Series Data Using K-Means and Dynamic Time Warping. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 733–738, April 2007.

[Nvi20] Nvidia. Recurrent Neural Network. `https://developer.nvidia.com/discover/recurrent-neural-network`, 2020. Accessed: 2020-04-09.

[OCGO96] Patrick O'Neil, Edward Cheng, Dieter Gawlick, and Elizabeth O'Neil. The log-structured merge-tree (LSM-tree). *Acta Informatica*, 33(4):351–385, Jun 1996.

[O'L14] Daniel E. O'Leary. Embedding AI and Crowdsourcing in the Big Data Lake. *IEEE Intelligent Systems*, 29(5):70–73, Sep. 2014.

[Ola15] Christopher Olah. Understanding LSTM networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. Accessed: 2020-05-22.

[OLBO15a] Peter O'Donovan, Kevin Leahy, Ken Bruton, and Dominic T. J. O'Sullivan. Big data in manufacturing: a systematic mapping study. *Journal of Big Data*, 2(1):20, Sep 2015.

[OLBO15b] Peter O'Donovan, Kevin Leahy, Ken Bruton, and Dominic TJ O'Sullivan. An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities. *Journal of Big Data*, 2(1):25, 2015.

[Ope18] OpenTSDB. OpenTSDB The Scalable Time Series Database. `http://opentsdb.net/`, 2018. Accessed: 2020-04-27.

[Ora20] Oracle. MySQL Community Edition. `https://www.mysql.com/products/community/`, 2020. Accessed: 2020-04-27.

[Pal10] George Pallis. Cloud Computing: The New Frontier of Internet Computing. *IEEE Internet Computing*, 14(5):70–73, Sep. 2010.

[Par66] Douglas F Parkhill. Challenge of the computer utility. 1966.

[Par98] Michael F. Parker. The International Intelligent Manufacturing Systems Initiative. *IFAC Proceedings Volumes*, 31(15):517 – 521, 1998. 9th IFAC Symposium on Information Control in Manufacturing 1998 (INCOM '98), Nancy, France, 24-26 June.

[PB19] Themis Palpanas and Volker Beckmann. Report on the First and Second Interdisciplinary Time Series Analysis Workshop (ITISA). *SIGMOD Rec.*, 48(3):36–40, December 2019.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[PBN12] Aditya B. Patel, Manashvi Birla, and Ushma Nair. Addressing big data problem using Hadoop and Map Reduce. In *2012 Nirma University International Conference on Engineering (NUiCONE)*, pages 1–5, Dec 2012.

[PE04] James L Peugh and Craig K Enders. Missing data in educational research: A review of reporting practices and suggestions for improvement. *Review of educational research*, 74(4):525–556, 2004.

[PF02] Kevin B. Pratt and Eugene Fink. Search For Patterns in Compressed Time Series. *International Journal of Image and Graphics*, 02(01):89–106, 2002.

[PFCZ18] Antonella Petrillo, F De Felice, Raffaele Cioffi, and Federico Zomparelli. Fourth industrial revolution: Current practices, challenges, and opportunities. *Digital Transformation in Smart Manufacturing*, pages 1–20, 2018.

[PHT08] Pengtao Jia, Huacan He, and Tao Sun. Error Restricted Piecewise Linear Representation of Time Series Based on Special Points. In *2008 7th World Congress on Intelligent Control and Automation*, pages 2059–2064, 2008.

[PL05] Liu Peng and Lei Lei. A review of missing data treatment methods. *Intelligent Information Management Systems and Technologies*, 1(3):412–419, 2005.

[Pla16] Plattform Industrie 4.0. Reference Architectural Model Industrie 4.0 (RAMI4.0) - An Introduction. Technical report, Plattform Industrie 4.0, October 2016.

[PM02] I. Popivanov and R. J. Miller. Similarity search over time-series data using wavelets. In *Proceedings 18th International Conference on Data Engineering*, pages 212–221, 2002.

[PMS17] Arbër Perçuku, Daniela Minkovska, and Lyudmila Stoyanova. Modeling and Processing Big Data of Power Transmission Grid Substation Using Neo4j. *Procedia Computer Science*, 113:9 – 16, 2017.

[PMS18] Arbër Perçuku, Daniela Minkovska, and Lyudmila Stoyanova. Big Data And Time Series Use In Short Term Load Forecasting In Power Transmission System. *Procedia Computer Science*, 141:167 – 174, 2018.

[Pos19] PostgreSQL Global Development Group. PostgreSQL. `https://www.postgresql.org/`, 2019. Accessed: 2020-04-27.

[PPAI16] Irfan Pratama, Adhistya Erna Permanasari, Igi Ardiyanto, and Rini Indrayani. A review of missing values handling methods on time-series data. In *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*, pages 1–6, 2016.

[Pre11] President's Council of Advisors on Science and Technology. Report to the President on Ensuring American Leadership in Advanced Manufacturing. Technical report, President's Council of Advisors on Science (PCAST), June 2011.

[Pro14] Prometheus. Prometheus From metrics to insight. `https://prometheus.io/`, 2014. Accessed: 2020-04-27.

[PS90] Gregory Piatetsky-Shapiro. Knowledge Discovery in Real Databases: A Report on the IJCAI-89 Workshop. *AI Magazine*, 11(4):68, Dec. 1990.

[PS14] Gregory Piatetsky-Shapiro. CRISP-DM, still the top methodology for analytics, data mining, or data science projects. `https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html`, October 2014. Accessed: 2020-03-27.

[PTRC07] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and Samir Chatterjee. A Design Science Research Methodology for Information Systems Research. *J. Manage. Inf. Syst.*, 24(3):45–77, December 2007.

[PVK+04] T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos, and W. Truppel. Online amnesic approximation of streaming time series. In *Proceedings. 20th International Conference on Data Engineering*, pages 339–349, April 2004.

[PWC] PWC. Industry 4.0 Smart Manufacturing Analytics Platform. `https://www.pwc.com/it/it/publications/assets/docs/pwc-smart-manufacturing-analytics-platform-overview-deck-september2017.pdf`. Accessed: 2020-01-02.

[PWZP00] C. Perng, H. Wang, S. R. Zhang, and D. S. Parker. Landmarks: a new model for similarity-based pattern querying in time series databases. In *Proceedings of 16th International Conference on Data Engineering (Cat. No.00CB37073)*, pages 33–42, 2000.

[QWW98] Yunyao Qu, Changzhou Wang, and X. Sean Wang. Supporting Fast Search in Time Series for Movement Patterns in Multiple Scales. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, page 251–258, New York, NY, USA, 1998. Association for Computing Machinery.

[RC67] A. H. Robinson and C. Cherry. Results of a prototype television bandwidth compression scheme. *Proceedings of the IEEE*, 55(3):356–364, March 1967.

[RGKG+17] Sergio Ramírez-Gallego, Bartosz Krawczyk, Salvador García, Michał Woźniak, and Francisco Herrera. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39 – 57, 2017.

[RGR18] David Reinsel, John Gantz, and John Rydning. Data Age 2020 The Digitalization of the World. Technical report, 2018.

[Ris18] Michael Risse. The new rise of time-series databases, Feb 2018.

[RK09] Teemu Räsänen and Mikko Kolehmainen. Feature-Based Clustering for Electricity Use Time Series Data. In *Adaptive and Natural Computing Algorithms*, pages 401–412, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[RKBL05] Chotirat Ratanamahatana, Eamonn Keogh, Anthony J. Bagnall, and Stefano Lonardi. A Novel Bit Level Time Series Representation with Implication of Similarity Search and Clustering. In *Advances in Knowledge Discovery and Data Mining*, pages 771–777, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[RRJP13] Vineet Richharya, DJ Rana, DR Jain, and DK Pandey. Design of trust model for efficient cyber attack detection on fuzzified large data using data mining techniques. *International Journal of Research in Computer and Communication Technology*, 2(3):126–130, 2013.

[RSS16] Dharavath Ramesh, Ashay Sinha, and Suraj Singh. Data modelling for discrete time series data using Cassandra and MongoDB. In *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, pages 598–601, 2016.

[RW05] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[RW+11] Patrick Royston, Ian R White, et al. Multiple imputation by chained equations (MICE): implementation in Stata. *J Stat Softw*, 45(4):1–20, 2011.

[SA07] Fadzilah Siraj and Mansour Ali Abdoulha. Mining enrolment data using predictive and descriptive approaches. *Knowledge-Oriented Applications in Data Mining*, pages 53–72, 2007.

[Sad18] Lamyaa Sadouk. CNN Approaches for Time Series classification. In *Convolutional Neural Network*. IntechOpen, 2018.

[Sam59] A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[SBS+13] Maxim Vladimirovich Shcherbakov, Adriaan Brebels, Nataliya Lvovna Shcherbakova, Anton Pavlovich Tyukov, Timur Alexandrovich Janovsky, and Valeriy Anatol'evich Kamaev. A survey of forecast error measures. *World Applied Sciences Journal*, 24(24):171–176, 2013.

[SBS+20] Saptarshi Sengupta, Sanchita Basak, Pallabi Saikia, Sayak Paul, Vasilios Tsalavoutis, Frederick Atiah, Vadlamani Ravi, and Alan Peters. A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowledge-Based Systems*, page 105596, 2020.

[SC07] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[Sch18] John Schulz. Cassandra As A Time Series Database, March 2018. Accessed: 2020-04-24.

[SCS+12] Jim Seeger, Naresh Chainani, Aruna De Silva, Karen Mcculloch, Kiran Chinta, Vincent Kulandai Samy, and Tom Hart. DB2 V10.1 Multi-temperature Data Management Recommendations, April 2012. Accessed: 2020-01-28.

[SdRRSB13] Walter J. Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E. Boult. Toward Open Set Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, July 2013.

[She00] Colin Shearer. The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing*, 5(4):13–22, 2000.

[SIL07] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[SKRC10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The Hadoop Distributed File System. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–10, May 2010.

[Sma11] Smart Manufacturing Leadership Coalition. Implementing 21st Century Smart Manufacturing: Workshop Summary Report. Technical report, Smart Manufacturing Leadership Coalition (SMLC), June 2011.

[Smi17] Taylor G. Smith. pmdarima: Arima estimators for Python, 2017. Accessed 2020-01-21.

[SMN+10] Doruk Sart, Abdullah Mueen, Walid Najjar, Eamonn Keogh, and Vit Niennattrakul. Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs. In *2010 IEEE International Conference on Data Mining*, pages 1001–1006, Dec 2010.

[SMT+18] Sakti Saurav, Pankaj Malhotra, Vishnu TV, Narendhar Gugulothu, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Online Anomaly Detection with Concept Drift Adaptation Using Recurrent Neural Networks. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, CoDS-COMAD '18, page 78–87, New York, NY, USA, 2018. Association for Computing Machinery.

[SNN18] Sima Siami-Namini and Akbar Siami Namin. Forecasting economics and financial time series: Arima vs. lstm. *arXiv preprint arXiv:1803.06386*, 2018.

[Sol19] Solid IT. DB-Engines Ranking. `https://db-engines.com/en/ranking`, 2019. Accessed: 2020-03-05.

[Spa19] Apache Spark. Spark. `https://spark.apache.org/`, 2019. Accessed: 2020-01-21.

[SPLH17] Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. Online deep learning: Learning deep neural networks on the fly. *arXiv preprint arXiv:1711.03705*, 2017.

[SR18] Gangadhar Shobha and Shanta Rangaswamy. Chapter 8 - Machine Learning. In *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, volume 38 of *Handbook of Statistics*, pages 197 – 228. Elsevier, 2018.

[Sre14] A. T. Sree Dhevi. Imputing missing values using Inverse Distance Weighted Interpolation for time series data. In *2014 Sixth International Conference on Advanced Computing (ICoAC)*, pages 255–259, 2014.

[SS98] Zbigniew R. Struzik and Arno Siebes. Wavelet transform in similarity paradigm. In *Research and Development in Knowledge Discovery and Data Mining*, pages 295–309, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[SS99] Zbigniew R. Struzik and Arno Siebes. The Haar Wavelet Transform in the Time Series Similarity Paradigm. In *Principles of Data Mining and Knowledge Discovery*, pages 12–22, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[SS11] Naphaporn Sirikulviriya and Sukree Sinthupinyo. Integration of rules from a random forest. In *International Conference on Information and Electronics Engineering*, volume 6, pages 194–198, 2011.

[SS17] Pallavi Sethi and Smruti R Sarangi. Internet of things: architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, 2017, 2017.

[SSH+18] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734, Nov 2018.

[SSHT06] M Shahbaz, M Srinivas, J A Harding, and M Turner. Product Design and Manufacturing Process Improvement Using Association Rules. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 220(2):243–254, 2006.

[SSRG13] Ilari Shafer, Raja R. Sambasivan, Anthony Rowe, and Gregory R. Ganger. Specialized Storage for Big Numeric Time Series. In *Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems*, San Jose, CA, 2013. USENIX.

[Sti74] Stephen M Stigler. Gergonne's 1815 paper on the design and analysis of polynomial regression experiments. *Historia Mathematica*, 1(4):431 – 439, 1974.

[STS16] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315, 2016.

[STZ00] C. Shahabi, X. Tian, and W. Zhao. TSA-tree: a wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data. In *Proceedings. 12th International Conference on Scientific and Statistica Database Management*, pages 55–68, 2000.

[SVG+17] Sreelekshmy Selvin, R Vinayakumar, E. A Gopalakrishnan, Vijay Krishna Menon, and K. P. Soman. Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647, Sep. 2017.

[SW65] Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[SWHZ11] Xiao-Sheng Si, Wenbin Wang, Chang-Hua Hu, and Dong-Hua Zhou. Remaining useful life estimation – A review on the statistical data driven approaches. *European Journal of Operational Research*, 213(1):1 – 14, 2011.

[SXL17] Lei Shu, Hu Xu, and Bing Liu. Doc: Deep open classification of text documents. *arXiv preprint arXiv:1709.08716*, 2017.

[Sys20] Savvy Data Systems. Savvy Data Systems. `https://www.savvydatasystems.com/es/inicio`, 2020. Accessed: 2020-04-09.

[SZ96] H. Shatkay and S. B. Zdonik. Approximate queries and representations for large data sequences. In *Proceedings of the Twelfth International Conference on Data Engineering*, pages 536–545, 1996.

[TAL14] Jiliang Tang, Salem Alelyani, and Huan Liu. *Feature selection for classification: A review*, pages 37–64. CRC Press, 1 2014.

[Tan08] Martín Tanco. *Metodología para la aplicación del Diseño de Experimentos (DoE) en la industria.* PhD thesis, Universidad de Navarra, 2008.

[TBAS12] Souhaib Ben Taieb, Gianluca Bontempi, Amir F. Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, 39(8):7067 – 7083, 2012.

[TFLC08] Tak-chung Fu, Fu-lai Chung, Robert Luk, and Chak-man Ng. Representing financial time series based on data point importance. *Engineering Applications of Artificial Intelligence*, 21(2):277 – 300, 2008.

[TGDH93] J. Timmer, C. Gantert, G. Deuschl, and J. Honerkamp. Characteristics of hand tremor time series. *Biological Cybernetics*, 70(1):75–80, Nov 1993.

[The19] The Apache Software Foundation. Apache Cassandra. `http://cassandra.apache.org/`, 2019. Accessed: 2020-03-21.

[TQ19] Fei Tao and Qinglin Qi. New IT Driven Service-Oriented Smart Manufacturing: Framework and Characteristics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(1):81–91, 2019.

[TQLK18] Fei Tao, Qinglin Qi, Ang Liu, and Andrew Kusiak. Data-driven smart manufacturing. *Journal of Manufacturing Systems*, 48:157 – 169, 2018. Special Issue on Smart Manufacturing.

[Trz10] Michał Trzęsiok. The importance of predictor variables for individual classes in SVM. 2010.

[TSM08] Sudhakar Teegavarapu, Joshua D Summers, and Gregory M Mocko. Case study method for design research: A justification. In *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 495–503. American Society of Mechanical Engineers Digital Collection, 2008.

[TWW17] Klaus-Dieter Thoben, Stefan Wiesner, and Thorsten Wuest. "Industrie 4.0" and smart manufacturing-a review of research issues and application examples. *International Journal of Automation Technology*, 11(1):4–16, 2017.

[VBD+18] Kevin Villalobos, Idoia Berges, Borja Diez, Alfredo Goñi, and Arantza Illarramendi. A multi-services architecture for Smart Manufacturing scenarios. In *International Conference on Industrial Internet of Things and Smart Manufacturing*, Lecture Notes on Data Engineering and Communications Technologies, Imperial College London, London, United Kingdom, September 2018. Springer.

[VDI+18] Kevin Villalobos, Borja Diez, Arantza Illarramendi, Alfredo Goñi, and José Miguel Blanco. I4TSRS: A System to Assist a Data Engineer in Time-Series Dimensionality Reduction in Industry 4.0 Scenarios. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 1915–1918, New York, NY, USA, 2018. Association for Computing Machinery.

[VGD] Michail Vlachos, Dimitrios Gunopulos, and Gautam Das. *Indexing Time-Series Under Conditions of Noise*, pages 67–100.

[Vil15] Kevin Villalobos. Pre-procesado de Datos Captados por Sensores Industriales en un Caso Real de Fabricación Inteligente. Master's thesis, University of the Basque Country UPV/EHU, 2015.

[VJT+17] P. Vazan, D. Janikova, P. Tanuska, M. Kebisek, and Z. Cervenanska. Using data mining methods for manufacturing process control. *IFAC-PapersOnLine*, 50(1):6178 – 6183, 2017. 20th IFAC World Congress.

[VR88] Sandra Vandermerwe and Juan Rada. Servitization of business: Adding value by adding services. *European Management Journal*, 6(4):314 – 324, 1988.

[VRD+] Kevin Villalobos, Víctor Julio Ramírez, Borja Diez, José Miguel Blanco, Alfredo Goñi, and Arantza Illarramendi. A Three-Level Hierarchical Architecture for an Efficient Storage of Industry 4.0 Data. *Computers in Industry*. Forthcoming.

[VRD+19] Kevin Villalobos, Víctor Julio Ramírez, Borja Diez, José Miguel Blanco, Alfredo Goñi, and Arantza Illarramendi. A Hierarchical Storage System for Industrial Time-Series Data. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 699–705, 2019.

[VVD+18] Kevin Villalobos, Jon Vadillo, Borja Diez, Borja Calvo, and Arantza Illarramendi. I4TSPS: a Visual-Interactive Web System for Industrial Time-Series Pre-processing. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2012–2018, Dec 2018.

[Wal19] Robert Walters. Time Series Data and MongoDB: Part 1 – An Introduction, September 2019. Accessed: 2020-04-24.

[WDH18] Qianhui Wu, Keqin Ding, and Biqing Huang. Approach for fault prognosis using recurrent neural network. *Journal of Intelligent Manufacturing*, pages 1–13, 2018.

[Wil17] Seunghye J Wilson. Data representation for time series data mining: time domain approaches. *Wiley Interdisciplinary Reviews: Computational Statistics*, 9(1):e1392, 2017.

[WL18] Timothy Wong and Zhiyuan Luo. Recurrent Auto-Encoder Model for Large-Scale Industrial Sensor Signal Analysis. In *Engineering Applications of Neural Networks*, pages 203–216, Cham, 2018. Springer International Publishing.

[WLL+10] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du. Research on the architecture of Internet of Things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 5, pages V5–484–V5–487, Aug 2010.

[WLZ+19] Kang Wang, Kenli Li, Liqian Zhou, Yikun Hu, Zhongyao Cheng, Jing Liu, and Cen Chen. Multiple convolutional neural networks for multivariate time series prediction. *Neurocomputing*, 360:107 – 119, 2019.

[WMD+13] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Scheuermann Peter Trajcevski, Goce and, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, Mar 2013.

[WMW+19] Renzhuo Wan, Shuping Mei, Jun Wang, Min Liu, and Fan Yang. Multivariate Temporal Convolutional Network: A Deep Neural Networks Approach for Multivariate Time Series Forecasting. *Electronics*, 8(8):876, 2019.

[WMZ+16] Jost Wübbeke, Mirjam Meissner, Max J Zenglein, Jaqueline Ives, and Björn Conrad. Made in China 2025: The making of a high-tech superpower and consequences for industrial countries. *Mercator Institute for China Studies*, 17:2017–09, 2016.

[WMZ+18] Jinjiang Wang, Yulin Ma, Laibin Zhang, Robert X. Gao, and Dazhong Wu. Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems*, 48:144 – 156, 2018. Special Issue on Smart Manufacturing.

[WTL+17] Jiafu Wan, Shenglong Tang, Di Li, Shiyong Wang, Chengliang Liu, Haider Abbas, and Athanasios V. Vasilakos. A Manufacturing Big Data Solution for Active Preventive Maintenance. *IEEE Transactions on Industrial Informatics*, 13(4):2039–2047, Aug 2017.

[WW00] Changzhou Wang and X. Sean Wang. Supporting Subseries Nearest Neighbor Search via Approximation. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, CIKM '00, page 314–321, New York, NY, USA, 2000. Association for Computing Machinery.

[WWIT16] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, 2016.

[WYCS16] Jiandong Wang, Fan Yang, Tongwen Chen, and Sirish L. Shah. An Overview of Industrial Alarm Systems: Main Causes for Alarm Overloading, Research Status, and Open Problems. *IEEE Transactions on Automation Science and Engineering*, 13(2):1045–1061, April 2016.

[WYO17] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585, May 2017.

[XPK10] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A Brief Survey on Sequence Classification. *SIGKDD Explor. Newsl.*, 12(1):40–48, November 2010.

[Xu12] Xun Xu. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(1):75 – 86, 2012.

[XYGG18] Hansong Xu, Wei Yu, David Griffith, and Nada Golmie. A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective. *IEEE Access*, 6:78238–78259, 2018.

[YDJY17] Liu Yunpeng, Hou Di, Bao Junpeng, and Qi Yong. Multi-step Ahead Time Series Forecasting for Different Data Patterns Based on LSTM Recurrent Neural Network. In *2017 14th Web Information Systems and Applications Conference (WISA)*, pages 305–310, Nov 2017.

[YF00] Byoung-Kee Yi and Christos Faloutsos. Fast Time Sequence Indexing for Arbitrary Lp Norms. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, page 385–394, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[YK15] Shen Yin and Okyay Kaynak. Big Data for Modern Industry: Challenges and Trends [Point of View]. *Proceedings of the IEEE*, 103(2):143–146, 2015.

[YKBT19] Hui Yang, Soundar Kumara, Satish T.S. Bukkapatnam, and Fugee Tsung. The internet of things for smart manufacturing: A review. *IISE Transactions*, 51(11):1190–1216, 2019.

[YL03] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.

[YRD16] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. In *Advances in Neural Information Processing Systems 29*, pages 847–855. Curran Associates, Inc., 2016.

[YTA18] Ozal Yildirim, Ru San Tan, and U. Rajendra Acharya. An efficient compression of ECG signals using deep convolutional autoencoders. *Cognitive Systems Research*, 52:198 – 211, 2018.

[YYY+11] Zhihong Yang, Yingzhao Yue, Yu Yang, Yufeng Peng, Xiaobo Wang, and Wenji Liu. Study and application on the architecture and key technologies for IOT. In *2011 International Conference on Multimedia Technology*, pages 747–751, July 2011.

[YZvdS18] Jinsung Yoon, William R Zame, and Mihaela van der Schaar. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Transactions on Biomedical Engineering*, 66(5):1477–1490, 2018.

[YZZL16] Xiuwen Yi, Yu Zheng, Junbo Zhang, and Tianrui Li. ST-MVL: Filling Missing Values in Geo-sensory Time Series Data. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. IJCAI 2016, June 2016.

[ZCB10] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[ZCF+10] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster Computing with Working Sets. In *HotCloud*, 2010.

[ZD18] Sophie Zareei and Jeremiah D. Deng. Impact of Compression Ratio and Reconstruction Methods on ECG Classification for E-Health Gadgets: A Preliminary Study. In *AI 2018: Advances in Artificial Intelligence*, pages 85–97, Cham, 2018. Springer International Publishing.

[ZGL+18] Weishan Zhang, Wuwu Guo, Xin Liu, Yan Liu, Jiehan Zhou, Bo Li, Qinghua Lu, and Su Yang. LSTM-Based Analysis of Industrial IoT Equipment. *IEEE Access*, 6:23551–23560, 2018.

[Zha03] G. Peter Zhang. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50:159 – 175, 2003.

[Zha12] Shichao Zhang. Nearest neighbor selection for iteratively kNN imputation. *Journal of Systems and Software*, 85(11):2541 – 2552, 2012.

[Zho19] Zhaofeng Zhou. Table Store Time Series Data Storage – Architecture, May 2019.

[ZL08] Zhen-Yu He and Lian-Wen Jin. Activity recognition from acceleration data using AR model representation and SVM. In *2008 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2245–2250, 2008.

[ZLC+17] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, Feb 2017.

[ZPH98] Guoqiang Zhang, B. Eddy Patuwo, and Michael Y. Hu. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1):35 – 62, 1998.

[ZSWY17] Aoqian Zhang, Shaoxu Song, Jianmin Wang, and Philip S. Yu. Time Series Data Cleaning: From Anomaly Detection to Anomaly Repairing. *Proc. VLDB Endow.*, 10(10):1046–1057, June 2017.

[ZWL+16] Jianfeng Zhu, Chunli Wang, Chuankun Li, Xinjiang Gao, and Jinsong Zhao. Dynamic alarm prediction for critical alarms using a probabilistic model. *Chinese Journal of Chemical Engineering*, 24(7):881 – 885, 2016.

[ZZCW10] S. Zhang, S. Zhang, X. Chen, and S. Wu. Analysis and Research of Cloud Computing System Instance. In *2010 Second International Conference on Future Networks*, pages 88–92, Jan 2010.

[ZZL19] Bin Zhang, Shaohui Zhang, and Weihua Li. Bearing performance degradation assessment using long short-term memory recurrent network. *Computers in Industry*, 106:14 – 29, 2019.

[Ås69] K.J. Åström. On the choice of sampling rates in parametric identification of time series. *Information Sciences*, 1(3):273 – 278, 1969.