Manufacturing as a Data-Driven Practice: Methodologies, Technologies, and Tools

Manufacturing as a Data-Driven Practice: Methodologies, Technologies, and Tools

Tania Cerquitelli, *Member, IEEE,* Daniele Jahier Pagliari, *Member, IEEE,* Andrea Calimera, *Member, IEEE,* Lorenzo Bottaccioli, *Member, IEEE,* Edoardo Patti, *Member, IEEE,* Andrea Acquaviva, *Member, IEEE,* and Massimo Poncino, *Fellow, IEEE*

***Abstract*—In recent years, the introduction and exploitation of innovative information technologies in industrial contexts have led to the continuous growth of digital shop floor environments. The new Industry-4.0 model allows smart factories to become very advanced IT industries, generating an ever-increasing amount of valuable data. As a consequence, the necessity of powerful and reliable software architectures is becoming prominent along with data-driven methodologies to extract useful and hidden knowledge supporting the decision making process.**

This paper discusses the latest software technologies needed to collect, manage and elaborate all data generated through innovative IoT architectures deployed over the production line, with the aim of extracting useful knowledge for the orchestration of high-level control services that can generate added business value. This survey covers the entire data life-cycle in manufacturing environments, discussing key functional and methodological aspects along with a rich and properly classified set of technologies and tools, useful to add intelligence to data-driven services. Therefore, it serves both as a first guided step towards the rich landscape of literature for readers approaching this field, and as a global yet detailed overview of the current state-of-the-art in the Industry 4.0 domain for experts. As a case study, we discuss in detail the deployment of the proposed solutions for two research project demonstrators, showing their ability to mitigate manufacturing line interruptions and reduce the corresponding impacts and costs.

***Index Terms*—Data-centric architectures, data management, data analytics, Industry 4.0, Internet of Things, technologies.**

I. Introduction

INDUSTRY 4.0 advocates a scenario in which increased automation, intelligent environments, and continuous technological developments are tightly coupled with the work environment. As a revolutionary process opening many attractive possibilities, it is forcing manufacturing industries to change. This generates huge opportunities for some players and causes problems for others. The difference between these two outcomes lies in the capability to promptly understand and react, and in knowing exactly how to transform the technology-driven benefits into added value [1]. A key example of this dexterity is in knowing how to effectively extract useful advantage from available data. Intelligent production environments incorporate the use of various kinds of sensors, different in function and location, which collect huge amounts of data. In order to make constructive use of the gathered data, a radically new industrial model is needed which encompasses: (i) a merging of *Operational Technology (OT) domain*, where data is generated and which involves industrial and factory automation, supply chain management, and asset monitoring, and *Information Technology (IT) domain*, where data is consumed, and which includes business and office automation, enterprise web and mobile applications; (ii) a means for integration with efficient data storage and processing, thanks to powerful data analysis libraries that support the extraction of knowledge enabled by recent Big Data-oriented IT technologies.

As data is the key element of this new paradigm, an abstract yet general business requirement of modern manufacturing industries is to *access insights from data that can then effectively support decision-making processes in order to maximize revenues*. It is the nature of these insights (i.e., data-driven requirements) that should be used to exploit the appropriate IT solutions as a data-driven practice. To this aim, the main research goal of this manuscript is to *provide a global overview of data-driven methodologies, technologies, and tools used in the Industry 4.0 context*. This overview will include a discussion about various technological and methodological aspects, supporting the complete life-cycle of data, from their generation to the exploitation of their value to support business strategies, across the fundamental activities related to communication, management, and knowledge extraction, and properly integrated with application layers.

Throughout our analysis of these aspects, we address a number of research questions (RQ), such as:

- RQ1. *Which logical components are needed to promote intelligence in production environments?*
- RQ2. *How to allow bidirectional communications among the components identified above?*
- RQ3. *Which data-processing requirements must be considered in an intelligent factory context? And, how to model a manufacturing aspect (e.g., process, task) through a data-driven service?*
- RQ4. *How to combine data-centric services to deliver only the interesting knowledge items to the user communities of the manufacturing plant*?

The literature offers a rich landscape of surveys [2]–[8] addressing one or few specific issues among the ones listed above in great detail. However, to the best of our knowledge, no previous work has provided a complete overview of the entire data life-cycle in the specific context of an Industry

T. Cerquitelli, D. Jahier Pagliari, A. Calimera, D. Bottaccioli, E. Patti and M. Poncino are with the Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy. E-mail: name.surname@polito.it.

A. Acquaviva is with the Department of Electrical, Electronic and Information Engineering, University of Bologna, Bologna, Italy. E-mail: name.surname@unibo.it

4.0 scenario. Our paper's main objective is exactly this, i.e., to serve as a practical "entry point" for readers approaching this field for the first time, or for experts willing to have a global yet detailed summary of the current state-of-the-art in the Industry 4.0 domain. Specifically, we discuss managing and processing data generated through innovative IoT devices and architectures deployed throughout the production plant, and extracting useful knowledge for improving production processes and gaining competitive business advantages. We adopt a data-driven view of the overall system, by putting emphasis on the role of data in the various steps of processing. Rather than proposing a new architectural model, we envision an "architecture" meant as a layered view of the data management process, whose *layers correspond to key functional aspects*; lower layers are closer to the raw data, and moving up through the layers allows extracting increasing amounts of knowledge. More precisely, we stack functionalities such as data communication, data storage and management, data analytics, and data visualization working upwards from the bottom layer of data. For each layer, we thoroughly discuss state-of-the-art methodologies, technologies, and tools, classifying them systematically with respect to a rich set of criteria that take into account the requirements of an industrial scenario. To our knowledge, our classification criteria extend all those found in previous literature, and we believe that they could help readers and industrial actors easily understand each algorithm/technology/tool's advantages and drawbacks, with specific regard to Industry 4.0 scenarios and requirements.

In reviewing the state-of-the-art for each of these functionalities, we will also present the deployment of the proposed architecture and the relative technologies onto two use cases involving very diverse manufacturing contexts and business objectives. The first case addresses the prediction of the remaining useful life of a robotic arm through the analysis of streams of robotics cycles. The second case describes the near real-time monitoring of the Overall Equipment Effectiveness (OEE) of a production line in a pharmaceutical packaging company. Both use cases will be mapped with respect to our architecture template and to the technologies described in the next sections. We will then conclude our overview by identifying some open issues related to data-driven methodologies in Industry 4.0 scenarios, based both on our review of the literature and on our direct expertise on the field through these and other use cases.

This kind of survey could benefit practitioners, engineering, and researchers in academia. The first two categories could be interested in the proposed survey since they may need to design data-driven architectures and methodologies in the Industry 4.0 scenario. To them, our work provides a complete overview of the state-of-the-art, with a comparative discussion of different solutions, practical examples of their deployment, and a discussion of open issues. More specific expertise can then be acquired by reading dedicated surveys [2]–[8] addressing each single data-driven functionality/layer in depth. Researchers in academia could benefit from a global overview of the key components of a data-centric architecture tailored to an intelligent factory scenario, discussed from a technological and methodological point of view.

This paper is organized as follows. Section II describes the overview of a data-centric architecture through its different functionalities to collect, manage, store, analyze, interpret data collected in an industrial framework (Answer to the RQ1). Section III, Section IV, Section V, and Section VI detail the four main data-centric functionalities through an in-depth discussion of manufacturing needs, state-of-the-art methodologies, and enabling technologies (Answers to the RQ2-RQ4). Section VII describes the robotics industry use case, while Section VIII presents our experience in the context of a pharmaceutical packaging context. Finally, Section IX discusses open issues to be addressed by the research community to streamline the diffusion of data-driven architectures and methodologies in the Industry 4.0 era.

A list of all acronyms and abbreviations used in the rest of the paper is reported in Table I.

II. Architectures

The main functional data platform exploited in the era of Industry 4.0 is composed of several logical components, which collectively provide a variety of data management and analytics functionalities promoting intelligence in production environments.

Several standards are involved to define the integration and interoperability of technological components [9]. Figure 1 presents a high-level general-purpose architectural view of the platform for Industry 4.0 scenarios, which is highly distributed. This diagram does not introduce a new Industrial IoT (IIoT) platform or data management architecture, for which a rich landscape of options exists in the literature, including both open-source research oriented (e.g. [10]–[12]), or commercial and industrial solutions (examples can be found in [13]–[15] while a complete review of commercial solutions is reported in [16]). Rather, it presents a data-centric view of the main functionalities and data-driven services in a manufacturing context that is used as navigation map for the technologies discussed in this paper.

The bottom layer in the architecture corresponds to the "input" of the data-centric flow, and integrates heterogeneous data sources (bottom of Figure 1). The latter include: (1) Pre-existing data (Additional Data Sources) available in the manufacturing company, for instance obtained by business applications or collected from the manufacturing environment in the past; (2) Data coming directly from the production machines (Production Line), for instance obtained by built-in sensors and extracted by PLCs, or (3) data collected through ad-hoc sensors added to the machines (IoT Devices). This layer must be designed to allow the interoperability among heterogeneous Internet-connected devices by abstracting the different underlying low-level technologies. To this aim, specific *Device Connectors* (or gateways) must be developed for each technology [17]. The *Device Connector* is a software component that abstracts device features and functionalities into common and unified interfaces. Data can be ingested in a batch or streaming fashion, depending on the kind of data source and the use-case. Optionally, *Device Connectors* can implement pre-processing functionalities before sending data

TABLE I
LIST OF ACRONYMS AND ABBREVIATIONS USED IN THE PAPER, IN ALPHABETICAL ORDER. ABBREVIATED NAMES OF COMMERCIAL PRODUCTS ARE NOT REPORTED.

Abbreviation	Meaning	Abbreviation	Meaning
ACID	Atomicity, Consistency, Isolation and Durability	MAAPE	Mean Arctangent Absolute Percentage Error
AMQP	Advanced Message Queuing Protocol	ML	Machine Learning
ANN	Artificial Neural Network	MQTT	Message Queuing Telemetry Transport
ANOVA	Analysis of Variance	MSE	Mean Squared Error
API	Application Programming Interface	NoSQL	Non-relational database
AR	Augmented Reality	OEE	Overall Equipment Effectiveness
ARIMA	Auto-Regressive Integrated Moving Average	OSA-CBM	Open System Architecture for Condition-Based Maintenance
AUC	Area Under Curve	OT	Operational Technology
CART	Classification and Regression Tree	PCA	Principal Component Analysis
CNN	Convolutional Neural Network	PLC	Programmable Logic Controller
CPU	Central Processing Unit	PR	Precision-Recall
DBSCAN	Density-Based Spatial Clustering of Applications with Noise	QoS	Quality of Service
DDS	Data Distribution Service	RDBMS	Relational DataBase Management System
DL	Deep Learning	REST	Representational State Transfer
DS	Descriptor Silhouette	RF	Random Forest
FD	Fault Detection	RFECV	Recursive Feature Elimination with Cross-Validation
FP	Fault Prediction	RMSE	Root MSE
GMM	Gaussian Mixture Model	ROC	Receiver Operating Characteristic
GPU	Graphics Processing Unit	RQ	Research Question
HTTP	Hyper-Text Transfer Protocol	RUL	Remaining Useful Life
IIoT	Industrial IoT	SQL	Structured Query Language
IP	Internet Protocol	STOMP	Simple Text Oriented Message Protocol
IT	Information Technology	SVM	Support Vector Machine
IoT	Internet of Things	TCN	Temporal Convolutional Network
JMS	Java Message Service	TCP	Transmission Control Protocol
JSON	JavaScript Object Notation	TPU	Tensor Processing Unit
JSON-LD	JSON for Linked Data	UDP	User Datagram Protocol
k-NN	k-Nearest Neighbors	XMPP	Extensible Messaging and Presence Protocol
KPI	Key Performance Indicator	ZMTP	ZeroMQ Message Transport Protocol
LSTM	Long-Short Term Memory		

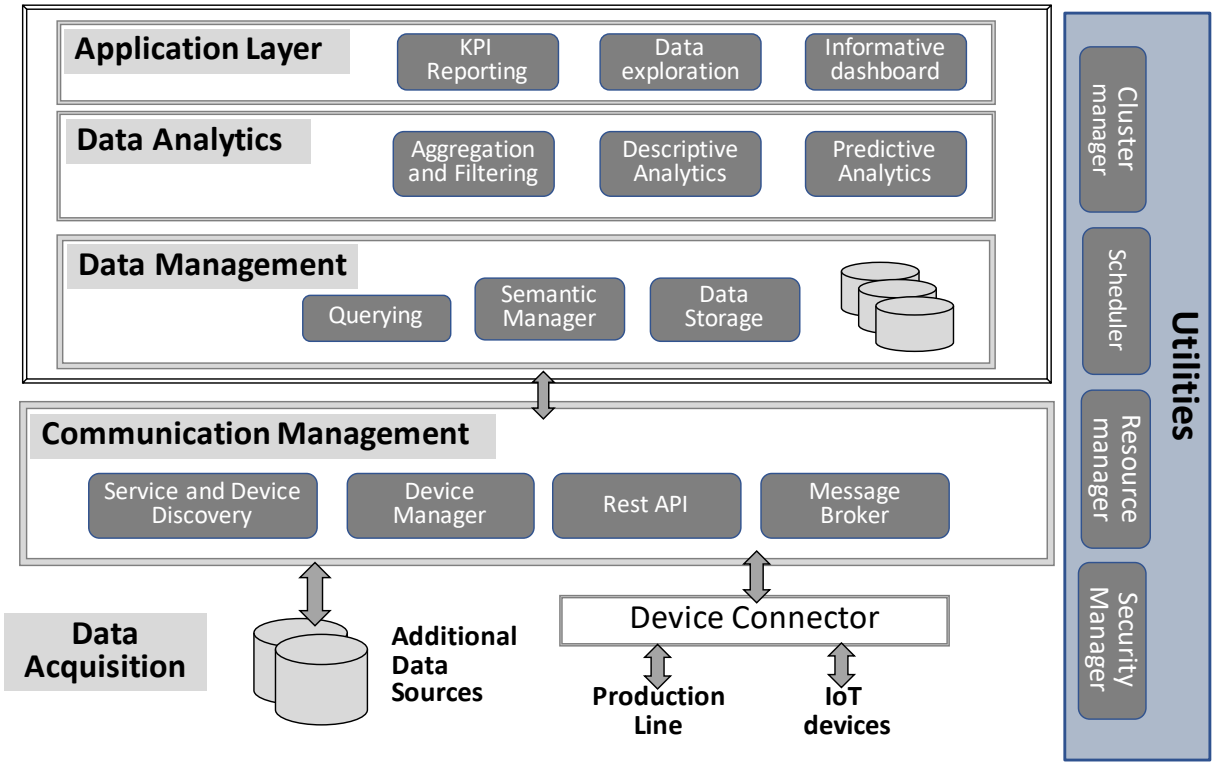


Fig. 1. The Reference Architecture used in this work.

to other entities in the platform as a form of edge computing (see Section V).

The core of the architecture performs two main tasks, i.e. *communication* and *data-centric processing*. The latter can be further split into three key functionalities: data management, data analytics and data-driven applications. Each functionality is briefly discussed hereafter, and elaborated in details in the corresponding section. In Figure 1 functionalities

are represented as rectangles with double lines border. Each functionality is obtained by combining broad categories of operations corresponding to integrated *services* (represented as rounded rectangles), i.e., specific sub-functions relative to the functionality that contains them.

Communication Management. This layer provides services to allow bidirectional communications among all the entities in the platform exploiting either request/response or publish/subscribe [18] communication paradigms. The choice of either depends on the context and involves different technical solutions, as shown in Figure II. The *REST (Representational State Transfer) API Manager* service implements a request/response communication paradigm, whereas Publish/subscribe communication is implemented by the *Message Broker*.

This layer provides two other key services i) *Device Manager* to register and manage all the available IoT devices in the platform and ii) *Service and Device Discovery* to allow applications and other platform components to uncover available services and devices.

Data Management services. This layer is responsible for the persistence of data and must be built on top of database system technologies able to offer a wide range of data management functionalities. In Figure 1 the *Data Storage* service is mainly responsible for the storage of data. Specifically, raw data, including sensor measurements, alarms, assets, along with the results of analytics services, are stored in the data repository, while the information required to support the management of those data are managed by the *Semantic Manager*. The latter provides features to store and access semantic knowledge allowing interoperability across heterogeneous devices and technologies. As an example, the Semantic Manager can make available knowledge and meta-data about sensors and actuators as well as their relation to domain model objects and environments. Another crucial service is the *Querying*, which provides data indexing that allows quick response to queries and data extraction processes required by higher application levels.

Data Analytics services. This layer carries out the task of extracting actionable insights from data stored and/or streamed from the manufacturing environment, thus supporting both stream and batch data processing. As shown in Figure II, we envision three main types of analytics, jointly applied and integrated to build an efficient service: data-aggregation/filtering, descriptive and predictive modeling/analytics. The former provides different kinds of data aggregation from fine-grained versus coarse-grained information based on application requirements and objectives, data filtering based on statistical quantities, or alarm triggering based on specific thresholds. The other two services provide more complex insights hidden into large sets of data, to support more complex business actions. Specifically, descriptive analytics includes all those tasks for describing and summarizing data, or for discovering hidden and not trivial casual relationships among *past* data to describe a specific manufacturing phenomenon under analysis. The predictive analytics service provides data-driven models to predict *future* outcomes, in order to foresee what is likely to happen and to suggest viable strategies/actions to improve business outcomes. A common need in manufacturing frameworks to be addressed through this service is the prediction of assets failures, hence preventing production line stops (predictive maintenance). Notice that it is possible that the same stream of data can be processed both in streaming and in batch mode. For instance, historical production data extracted from a physical asset can be used to train a Remaining Useful Life (RUL) model (batch), while newly gathered data can be processed in real-time to generate informative dashboards containing Key Performance Indicators (KPIs).

Application-level services. This layer provides APIs and tools to develop services and applications that manage and process data coming from the underlying layers, and also to promote data exchange among services. A rich-set of applications can be developed and deployed. For example, *Informative Dashboards* collect relevant data processed by the different building blocks to present insights on monitored manufacturing processes. *Data Exploration* relies on analytical services to dynamically and visually describe data and analysis outcomes (e.g., insight characteristics, patterns discovered from data) helping to keep the manufacturing player in the analytics loop.

Finally, Figure 1 shows an additional fundamental element labeled **Utilities**. It is represented as a vertical box as its services provide different functionalities to build the cluster computing engine for large-scale data processing and guarantee data protection. A brief elaboration of utilities, in terms of a technological overview is reported in Section II-A since they are not strictly related to the *flow* of data processing, which is the main focus of our study.

In the rest of the paper, Sections III, IV, Section V, VI describe each of the main functionalities in more detail. Within each section we will first describe the main technical and methodological aspects to deliver the functionality, and then discuss the enabling technologies (hardware and/or software) involved in its deployment.

Industry 4.0 data-driven architectures could actually be considered an extension of earlier data warehousing models [19], [20] since the latter include a subset of the data services available in the former. Specifically, the data warehousing engine has faced an important evolution in the last few years due to the changes in the application requirements such as an increased growth in data generation capability, the high-speed evolution of the ICT technologies, and the shift in data modeling paradigms. However, data warehousing is mainly addressing data storage, query processing and data visualization. Conversely, the data-centric architecture surveyed in this paper supports the complete life-cycle of data, from their generation to the exploitation of their value, including communication functionality, advanced machine-learning-based services, and application-based services, which are software components that could communicate with data warehousing.

A. Utilities

As shown in Figure 1, the cross-layer *utilities* include some key services used to develop a cluster computing engine for large-scale data processing. *Cluster Manager* is the service in charge of coordinating all sets of independent processes

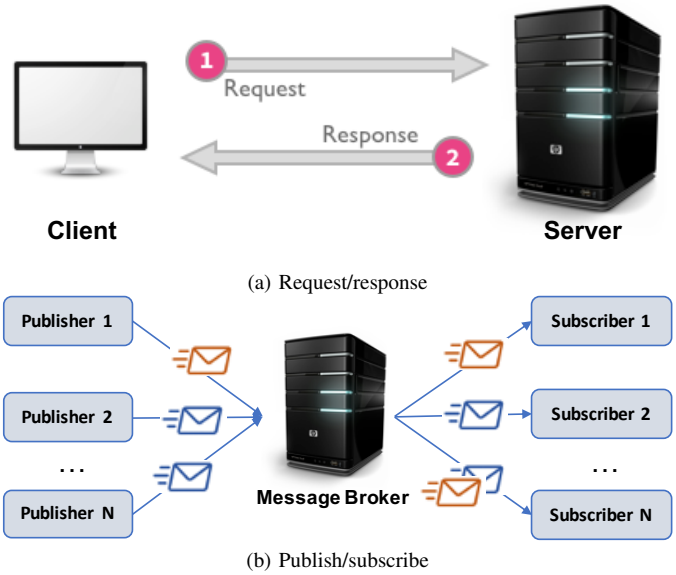
run on the engine by allocating resources to all processes. Inside the cluster manager, a *Scheduler* is responsible to properly allocate resources to the various running applications subject to different constraints including resource availability and requirements, and queue mechanisms. The scheduler monitors the application status and is responsible to accept job-submissions and to negotiate with the *Resource Manager* appropriate resources required to run the application. *Resource Manager* is then in charge of allocating jobs to the different logical and physical units of the cluster (e.g., nodes, workers, virtual machines).

A fundamental task is carried out by the *Security Manager*, which is meant to protect cloud data collected from the production field with services like authentication, authorization, encryption, and data lineage. A mutual authentication mechanism facilitating role-based access control policies is usually also implemented. This paper does not focus in depth on security aspects, despite their paramount importance, since there already exists a large body of literature reviews discussing security issues in manufacturing environments. The reader can refer to the papers [21]–[23] which give an exhaustive overview of the key approaches to design and exploit the security manager layer in manufacturing contexts.

Many cloud architectures in the context of manufacturing are designed by exploiting the micro-services [24] approach. This is a software design pattern which allows the development of an application as a set of micro-services, each running in its own process and exploiting lightweight mechanisms to communicate [25]. These services are *small, highly decoupled and focus on single tasks* [26]. An enabling technology, widely used in the deployment of micro-services architectures is the Docker system [27].

Docker is an open source technology based on the use of *containers*. This solution allows to wrap services into deployable units and isolate them from the underlying infrastructure that hosts the containers. A distribution of services through docker containers and dynamic communication channels can be implemented through a *Docker orchestration manager*. As containers are isolated from the underlying host infrastructure, and easily distributed between the available hosts, they improve the flexibility and agility of the system, allowing developers flexibility in scaling the system to meet their specific installation requirements, and to select and deploy only those services that satisfy their specific business needs. The Docker system can be deployed on different infrastructures, including servers, gateways, virtual machines or on public and private clouds. In addition, the services can then be distributed across an hybrid architecture, extending their functionality further to edge gateways on the manufacturing environments, if needed.

In the literature and in the state-of-the-practice there exist different frameworks for the manipulation and analysis of big data. Their overview is not in the scope of this survey, that instead focuses on vertical data management and processing solutions. Interested readers can refer to recent survey papers [4], [5], [28] to have an in-depth overview of the key technologies and tools to be properly integrated to deploy a cloud-based architecture able to fulfil application requirements and manage resources so to allow each service to work in



(a) Request/response

(b) Publish/subscribe

Fig. 2. Communication paradigms

efficiency and effectiveness. As an example we refer to a popular solution: Apache Spark [29]. It is a well-known open-source platform to build cluster computing engines for large-scale data processing. Spark provides a rich set of high-level APIs in different programming languages (Python, Java, Scala) and supports a variety of higher-level libraries to perform different kinds of data processing: batch and streaming (Spark Streaming [30]) processing, advanced analytics and machine learning (MLlib [31]), graph-based processing (GraphX [32]), and SQL (Spark SQL [33]).

III. Communication Management

The *Communication Management* layer consists of all those components for bidirectional communication across the different entities, either hardware or software, involved in Industry 4.0 scenarios. Components in this layer rely on protocols based on two different communication paradigms: *request/response* or *publish/subscribe* [18].

Request/response (see Figure 2a) is a communication paradigm typically implemented in a synchronous fashion involving two actors: i) a *client*, which requests some information (i.e., a resource) and ii) a *server*, which replies to clients sending back the requested information. In this paradigm, clients always start the communication, and servers must be always up and running, ready to reply to incoming requests. In modern distributed software platforms, request/response is commonly implemented via REST (Representational State Transfer) web services [34] over HTTP (Hypertext Transfer Protocol) [35].

A *REST API Manager* should be implemented by each entity in the platform to expose REST web services and enable such request/response communication.

Publish/subscribe enables an asynchronous communication that complements request/response. This communication paradigm removes all the dependencies between producers and

consumers of information [18], called in this paradigm *publishers and subscribers*, respectively. Thus, publish/subscribe allows the development of scalable loosely-coupled event-based systems, services and applications able to react in (near-) real-time to different events. This feature is essential for Industry 4.0 scenarios where we deal with several events coming either from hardware or software components.

In most of the publish/subscribe implementations, publishers send messages to a *Message Broker* and subscribers register to the same Message Broker for a specific information flow (see Figure 2b). In this scenario, the role of the broker is crucial to enable the communication, as it stores and forwards messages from publishers to subscribers. For this reason, it must be horizontally scalable and fault-tolerant to support advanced (near-) real-time applications with very high throughput. Moreover, some Message Broker implementations can also prioritize messages providing Quality of Service (QoS) functionalities. From the architectural point of view three are the possible configurations: *i) Centralised broker*, exploiting a server through which all traffic is transmitted; *ii) Distributed broker*, exploiting a messaging system based on IP multi-cast protocol for message routing, with no centralised message server; *iii) Decentralised Global Data Space* characterised by producers writing to a common data space and consumers reading from it, where the connection is implemented with a common data bus.

Table II provides an overview of the communication protocols used in an industrial context by summarizing a number of key features. Specifically, each protocol is described through 12 relevant criteria to highlight the main features. Based on such criteria readers can easily glance the key aspects of each protocol and select the one(s) that could be useful for their own use case. Among the defined criteria, the most characterizing ones correspond to the type of communication paradigm (Publish/subscribe and/or Request/Response, which are not mutually exclusive), and the support of a RESTful architecture, although all the listed features in the first column are relevant for their adoption.

Notice that most of these protocols are broker-based except DDS and XMPP, which exploit a distributed global data space and a centralised server, respectively. These protocols support communication between publishers and subscribers over TCP/IP to provide reliable communications. DDS however provides both TCP and UDP connections, while XMPP is the only one that does not support QoS features.

Talaminos et al. [3] have shown that all the considered protocols feature a low CPU usage, regardless of the number of publishers or subscribers. A distinction can be made between centralised and decentralised protocols as the publication rate increases. Indeed, a linear increase is observed in the CPU time requirement for centralised solutions compared to an exponential increase for distributed protocols. The reason is that in centralised protocols a publisher sends the message to the broker while in distributed protocols (e.g. DDS) a publisher directly communicates to subscribers. Concerning bandwidth consumption, distributed protocols show highest consumption with a publisher with high messaging rate. In centralized protocols, MQTT yields the lowest bandwidth consumption, while XMPP protocol requires a large bandwidth even for low sampling rates.

Table III reports the most relevant implementations of the protocols listed in Table II based on commercial solutions. To properly detail the different tools, we used 17 features to provide a rich set of characteristics that could guide the practitioners to select the best fits for their application requirements. First, we summarize the main features that affect reliability and scalability of the solution, such as distributed capabilities for brokers clustering.

Other important features listed include the types of messages and message patterns such as publish-subscribe, request-response and point-to-point communication, the computational load of the solution, the type of supported protocols, the routing support, the presence of priority queues, the possibility of using a cluster, load balancing support, type of security encryption and authentication, the available user API and the presence of Quality of Service (QoS) features.

Table II and Table III indeed show a very rich landscape for implementing communication in a data management flow. It is therefore possible to pick the solution that best fits the requirements of the specific use-case to be implemented in industrial scenarios, although this choice may require an accurate knowledge of the scenario.

The service provided by the Message Broker is definitely the most relevant in this layer and the one that characterizes the adopted solution. As shown in Figure 1, there are two additional relevant services in the *Communication Management*: i) the *Device Manager* and ii) the *Service and Device Discovery*. The *Device Manager* maintains a registry of available IoT devices in the platform (i.e. not disconnected from the Internet) and the resources they expose.

The *Service and Device Discovery* can be considered as the entry point for applications and other components to discover registered services (e.g. Message Brokers, Device Connectors and Rest API Managers) and devices in the platform. Some implementation of *Device Manager* and *Service and Device Discovery* can support either request/response via REST or publish/subscribe protocols. Sometimes *Device Manager* and *Service and Device Discovery* functionalities are implemented as a single software component as for [12].

IV. Data Management

In an intelligent factory context, full of sensors throughout the whole production chain, the ability to manage and process data in real time is of paramount importance. The requirements of real-time processing, reliability and continuity of operation, persistence of the data over time, consistency among all distributed copies, low latency in accessing the data, and high data transfer speed are among the most desired features in various applications of Industry 4.0. Specifically, data collected in production and industrial contexts must be versioned over time, easily accessible and processed efficiently and in real time, to support the different industrial players in the decision-making process. These operations are carried out efficiently and effectively by the database management systems currently representing the predominant technology for

TABLE II
COMPARISON OF COMMUNICATION PROTOCOLS

	KAFKA	MQTT	AMQP	XMPP	JMS	DDS	STOMP	ZMTP
Year	2011	1999	2003	1999	1998	2004	2010	2007
Licensing	Open Source and Commercially Licensed	Open Source and Commercially Licensed	Open Source and Commercially Licensed	Open Source and Commercially Licensed	Open Source and Commercially Licensed	Open Source and Commercially Licensed	Open Source and Commercially Licensed	Open Source and Commercially Licensed
XML Support	Yes	No	No	Yes	Yes	No	Yes	No
Encoding Format	Binary	Binary	Binary	Character	Binary	Binary	Character or Binary	Binary
Publish/Subscribe	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Request/Response	Yes	No	No	Yes	No	No	Yes	Yes
RESTful	Yes	No	No	No	No	No	No	No
Transport Protocol	TCP	TCP	TCP	TCP	TCP	TCP/UDP	TCP	TCP
Architecture	Distributed	Centralised	Centralised	Centralised	Centralised	Decentralised	Centralised	Centralised
Style	Broker	Broker	Broker	Server	Broker	(Global Data Space)	Broker	Broker
QoS	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Security	SSL - SASL	SSL	SSL - SASL	SSL	TLS/SSL - JASS	SSL/DTLS	TSL/SSL	CurveZMQ
Hard Real-time	Yes	Near real-time	Near real-time	Near real-time	Near real-time	Yes	No	Near real-time

TABLE III
COMPARISON OF COMMERCIAL SOLUTIONS

	KAFKA	RABBITMQ	MOSQUITTO	ZEROMQ	ACTIVEMQ	VERNEMQ	ROCKETMQ	APACHE QPID	YAMI4	WildFly	OpenFire
Development language	Scala / Java	Erlang	C	C++	Java	Erlang	Java	JAVA, C++	C++,Objective C	Java	Java
Year of release	2011	2007	2009	2007	2004		2012	2005	2010	2008	2007
Brokered/Brokerless	Brokered	Brokered	Brokered	Brokerless	Brokered	Brokered	Brokered	Brokered From AMQP 1.0. Brokerless form can also be implemented.	Brokered and Brokerless	Brokered and Brokerless	Brokered
Message supported formats	String / bytes	Map String	String / bytes	String / bytes	String, Map, Byte, Object	String / bytes	String, Byte	String, Byte	String / bytes	String / bytes	bytes
Message patterns supported	Pub-sub	Point-to-point Pub-sub Request-Response	Pub-Sub	Request-Response Pub-sub Workload distribution	Point-to-point Pub-Sub	Request-Response Pub-sub	Point-to-point Pub-Sub	Request-Response Pub-sub	Request-Response Pub-sub	Request-Response Pub-sub	Request-Response Pub-sub
Persistance Queue	Yes	Yes	Yes	No	Yes	No	Yes	Yes	No	No	No
Ligthweigth	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Application protocol supported	Kafka	AMQP, STOMP, MQTT, XMPP	MQTT Web-Socket	ZMTP	JMS, OpenWire, STOMP, REST, XMPP, AMQP	MQTT	RocketMQ	AMQP, JMS	YAMI4- a WIRE level protocol	JMS, MQTT, AMQP	XMPP, MQTT
HA support	Yes	Yes	Not Directly , tries to do this through bridging between two brokers.	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Routing support	Yes	Yes	No	Yes	Yes	No	No	Yes(through AMQP)	Yes	Yes	No
Priority Queue	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Cluster	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Load balacing	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Security Encription	SSL	TLS	TLS	CurveZMQ over TCP	TLS	TLS/SSL	TLS	SSL	SSL	SSL	TSL/SSL
Security Autentication	SASL	SASL	-	SASL	JASS	SASL	-	SASL	-	SASL / JASS	SASL
User API	Java, .NET, PHP, Python, Erlang, Perl, JavaScript, Ruby, Go	Java, .NET, PHP, Python, Erlang, JavaScript, Ruby, Go	Java, PHP, Python, C, C#, JS	C, C++, Java, .NET, Python	Java, PHP, Python, C, C#, C++, Perl, Ruby	Python, Go, C#, .Net	Java, C++, Go	C++, Java, Ruby, Perl, Python, C#	C++, Ada, Java, .NET, Python, Wolfram	Java, Python, C#	Java, Python, C#, .Net, Ruby, PHP, JavaScript
QoS support	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	No

storing structured data in all business-oriented applications. The relational database management systems (RDBMS) proposed by Codd in 1970 [36] has been the data model widely used over the years thanks to efficient and effective data modeling and their availability to multiple users concurrently. However, the entity-relationship model is mainly suitable for transactional applications, while they may be less efficient to manage heterogeneous, complex and large quantity data flows such as those monitored in industrial environments [37].

To overcome this limitation, in recent years we have witnessed the evolution of non-relational databases (NoSQL) [38], able to effectively support both transactional and analytics activities. Non-relational databases are able to process high volumes of data both in writing and in reading and seem the most promising technological solution for the management, storage and querying of data collected in next-generation industrial contexts. They guarantee the collection of large quantities of heterogeneous data continuously and reliably, their storage and distributed indexing, thus offering high scalability and reliability. They also provide good interoperability with various technological platforms for data analysis [39].

Following a comparative analysis of the most popular and promising technological solutions for addressing Industry 4.0 requests in terms of data storage using NoSQL systems, the following four platforms were identified: Cassandra [40], CouchDB [41], MongoDB [42], and DynamoDB [43]. CouchDB, Cassandra and MongoDB are open-source solutions, while DynamoDB is released under a commercial license and accessible only through the Amazon API. The ACID properties (Atomicity, Consistency, Isolation, and Durability) are offered at different extents by each solution. The user is often offered the opportunity to choose the desired consistency level, possibly limiting performance. In addition, transactions are partially supported for individual operations on the storage unit (for example, for a single document). The document represents a structured collection of pairs (key = value), where the key is usually used to indicate the name of the attribute of interest, thus allowing different composition of the attributes for each record in the database. CouchDB specifically adds a revision attribute to the document and provides a dedicated Replication Protocol that ensures seamless data flows among different nodes of the cluster, with the nodes possibly also being mobile devices or embedded platforms. Such an offline-first approach based on a true multi-master sync is probably one of the most differentiating features of CouchDB with respect to the alternatives. In terms of performance, the scalability offered by DynamoDB is certified by Amazon which manages the entire hardware infrastructure ensuring latencies in the order of milliseconds. The scalability and throughput performance for Cassandra and MongoDB, on the other hand, were analyzed in detail, with a clear preference for Cassandra in terms of scalability. For example, Cassandra offers higher throughput values and ensures fast and reliable management of large amounts of data in real time. It also offers a well-structured data representation model while maintaining dynamic elements. Finally, the vibrant developer community is also a key factor. Cassandra is widely supported by open-source communities as well as commercial organizations. There are also a variety of important open-source projects that provide native integrations of their systems with Cassandra, such as Apache Spark (see Section II-A). Cassandra is therefore complete from several points of views and could be an adequate choice for data management in Industry 4.0 applications that use devices at the crossroads between cybernetics and the physical world.

A. Metadata management: The MIMOSA open standard

Within the context of Industry 4.0 data management, a parallel research effort has been devoted to defining open standards to metadata management by guaranteeing compatibility among different products, components, and services supplied by different providers. The definition of a standard allows interpretation of general guidelines to reduce costs, improve interoperable services, increase business competition, incorporate design changes, and further cooperation in this field. In addition, the open characteristic of the standard allows its own growth, exploitation and improvement. In the context of manufacturing, the most widespread open standard is MIMOSA OSA-CBM [44] (Open System Architecture for Condition-Based Maintenance). It provides a detailed description of the data model to be used to integrate data and metadata useful for interpreting manufacturing data and deliver the right information subset to the right user. The data model, based on a relational database, comprises the integration of engineering, maintenance, operation, and reliability data into a single area of storage.

The MIMOSA data model defines all the metadata and the corresponding logical relationships (i.e., ontology-based) for achieving Condition-based Predictive Maintenance. It includes a large number of different relational tables all related to automatic maintenance and reliability. In the context of a manufacturing enterprise, the main metadata model requires the definition of the enterprise, a site, a segment, an asset, and for each monitored phenomenon, the location of the IoT device within the plant and all the necessary entities relating to data storage, alarm limits, alarm severity, and compensated actions when needed.

V. Data Analytics

Data analytics services represent the core “intelligence” behind any data-driven knowledge platform. They are central in the architecture of Figure 1, fueled by data gathered, transmitted and stocked by services below, and in charge to deliver knowledge upwards in the form of highly informative data about the manufacturing plant. The kind and format of distilled information may vary depending on the service requested at the application layer, and can be used in multiple ways and for different purposes of increasing complexity. They can be simply utilized in hindsight to justify/evaluate the actions taken, or to provide valuable insights for decision making, or, even more challenging, for a foresight of the upcoming trends and behaviors to anticipate corrective actions.

To a higher complexity must correspond a growing level of intelligence, often achieved combining two main types of analytics: *descriptive analytics* and *predictive analytics*.

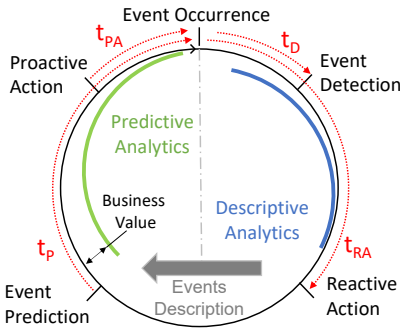


Fig. 3. Analytics services: descriptive vs. predictive.

Both are distinguished from (although built upon) elementary data manipulation services for *aggregation and filtering*, like computations of statistical moments, thresholding and comparisons. These latter, often used as preprocessing stages embedded into descriptive and predictive analytics, can also be used stand-alone to filter, map and generate higher level events with more expressive power, used for example in KPI computations.

Descriptive analytics include services for describing and summarizing data, or for discovering hidden and not trivial casual relationships among events. In a sense, these services "look to the past" to enable the understanding of the manufacturing process, what happened and (possibly) why.

Predictive analytics, in contrast, include services to predict future outcomes. They help to foresee what is likely to happen, and can be ultimately deployed to suggest viable strategies to improve decision making. To do this, they leverage descriptive technologies to infer aggregated forms of knowledge. Fig. 3 gives a pictorial description of such cooperation. Descriptive analytics can be deployed to detect an event already occurred, e.g. the presence of a fault on a production tool/machinery, which in turn may trigger a corrective action, e.g. the maintenance on the production line. The time-to-detect (t_D) and the time-to-reaction (t_{RA}) are two important metrics that define the responsiveness of the closed-loop control.

Similarly, descriptive analytics can be instructed to recognize patterns that indicate potential *future* problems or opportunities, empowering predictive analytics able to understand the when, why and how of upcoming events. If properly caught, event predictions may trigger proactive actions with the aim of minimizing costs, e.g. scheduling a maintenance plan that ensures the shortest downtime. The time interval between the prediction of an event (t_P) and the corresponding preventive action (t_{PA}) is a measure of proactiveness.

Apart from the responsive/proactive times and the quality of description/prediction, the overall performance are strictly related to the decisions that are taken and the actions that are implemented. However, there is no doubt that predictive analytics methods have a much higher business value than descriptive analytics. Therefore, despite their infancy, the many open issues, and the lack of a standard set of tools and reliable techniques, predictive analytics are considered the key novelty enabled by the ubiquitous data gathering from sensors in Industry 4.0. For such reason they are attracting the most of the industrial interest.

The recent literature is suggesting viable methodologies for building rather complex analytics services, which is the focus of the next sections. We hereby introduce a general pipeline, from Sections V-A to V-D, which serves both descriptive and predictive analytics, and does reflect the complementary nature of the underlying technologies, described later in Section V-E, from data aggregation & filtering stages, to model building and deployment via statistical or machine learning (ML) methods. With no lack of generality, we used two maintenance tasks as reference: *Remaining Useful Life* (RUL) estimation and *Fault/Anomaly Detection and Prediction*.

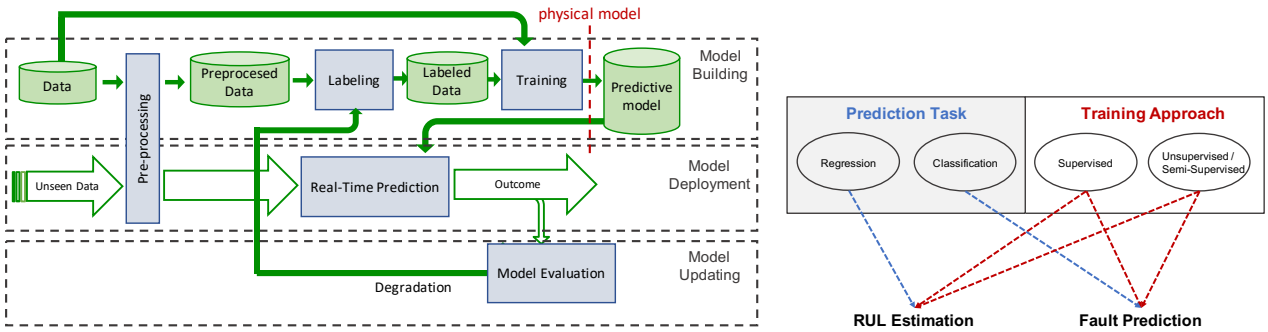
A. Analytics Tasks Formulation

Figure 4a shows a block diagram of the main services offered by an industrial data management platform and of their inter-operations. Services are represented as rectangles, while the flow of data is represented by green data stores and thick arrows for *batch* and *streaming* respectively. It is worth emphasising this pipeline should not be seen as a static flow of actions, but rather as a general template data engineers can use to build different analytics services, both descriptive and predictive.

As shown in the picture, there are three main sub-functionalities, generally labeled as *Model Building*, *Model Deployment* and *Model Updating*. While all elements of the pipeline are also present in other domains, an industrial setting poses peculiar challenges for some of the involved services, as detailed hereafter.

Figure 4b schematizes how the two reference tasks are formulated and their mapping to canonical prediction problems. When dealing with predictive maintenance of plant equipment, many authors [6], [7], [45]–[51] formulate the problem as a prediction of the RUL of the target asset, defined as the time (or number of production cycles) left before the end of its expected operational life [45]. The same asset operating under different conditions can incur a different degradation level, and have as a result a varying RUL value at the same point of time. RUL estimation is formulated as regression problem, where the quantity to be estimated is a continuous time value.

Alternatively, predictive maintenance can also be regarded as a *Fault Prediction (FP)* [52]–[58]. In this case, the problem is formulated as a classification, where the goal is to estimate the probability of a particular asset failing within a user-defined time horizon (e.g. one week from the current time). This tends to make the prediction easier with respect to RUL estimation, at the cost of a coarser time-granularity of the estimate. In the realm of descriptive analytics, FP becomes Fault Detection (FD), and can be applied to some interesting novel problems, such as computer vision-based quality control, where correctly manufactured components or partial assemblies have to be distinguished from faulty ones [59]–[61]. In both cases, the classification can be either *binary* (i.e. fault/no-fault) or *multi-class*, when the goal is also to distinguish different categories of faults based on their severity or on their root cause [53].



(a) General pipeline (b) Task formulation

Fig. 4. Industrial descriptive and predictive analytics services pipeline and corresponding task formulation and training approach

More than 50 industrial predictive analytics works published after 2014 were analyzed using 12 main criteria aimed to describe the data analytics pipeline, the main characteristics of the dataset used to validate each proposed approach along with the machine learning tools. Specifically, we describe each of the three sub-functionalities of Figure 4a with reference to these works, outlining prevalent trends and open issues. Table IV contains a summary chart of the main characteristics of some of the most relevant works analysed and discussed in the following subsections. We also included a criteria related to the used dataset to provide key aspects (e.g., class imbalance) of data distribution and its availability. The Machine Learning library used to develop the data-analytics methodology is also discussed since we believe that this criterion is very relevant to assess a possible integration into the industrial architecture.

B. Model Building

Models for RUL estimation and FD/FP can be either *physical*, i.e. based on the physics laws governing the asset or *data-driven*, i.e. based on statistical or Machine Learning (ML) algorithms [7], [49], [54]. While the former are more interpretable, they require a much greater domain expertise. Therefore, data-driven models, which are the main focus of this work, tend to have wider applicability.

From an implementation point of view model building is a computationally intensive procedure performed *offline* (i.e. separately from plant operations). It is not typically a latency-sensitive task, hence it can leverage the *batch* processing features of the underlying analytics platforms.

As shown in Figure 4, this functionality is based on three main services.

1) Preprocessing: This is the first step in the pipeline, and consists of filtering, cleaning and aggregating the data under analysis. The major preprocessing steps in industrial scenarios are reported below.

Outliers elimination is the detection and removal of extreme values that deviate significantly from the average. While outliers detection can itself be used for fault detection (see Section V-B3), it is also useful as pre-processing, for example to identify data that does not come from real production cycles but, e.g., from test cycles. For instance, in [45], the authors eliminate samples for which a wear-out reliability metric (i.e. RUL) increases rather than decreasing, under the assumption that is the result of a maintenance event.

Other works use clustering [48] or filtering [56], [62], [63] to eliminate outliers.

Features extraction, in the context of time-series, such as those used for predictive maintenance, is normally performed by first splitting the raw data into fixed segments. The size of segments is a user-defined parameter and in general corresponds to a fraction of the production cycle [46], [47], [53]. Then, a set of features is extracted for each segment using standard data aggregation and filtering techniques. The most commonly used features are simple statistical aggregates over the values in the segment, such as the mean, standard deviation, minimum and maximum, quartiles, Kurtosis, Skeweness, etc [6], [7], [46], [53], [56].

Time-frequency analysis is also common [6], [64].

Data normalization, which is required to improve the performance of some ML models [65], can also be considered as a basic form of feature extraction.

Features selection, also referred to as *dimensionality reduction*, consists in eliminating features (either raw or extracted) that provide little information for the target task. Covering the vast literature on feature selection algorithms is out of the scope of this work.

In practice, simple filter methods are often effective. These eliminate features that are constant [53] or highly correlated with others [6], [54], [56], [58].

Methods that combine feature selection and extraction, such as Principal Component Analysis (PCA) [54] are also common. However, the latter have interpretability issues, as the generated features do not have physical meaning. Correlation-based approaches are therefore preferable, as the identified correlations can be easily double-checked by plant experts. Finally, another common trend is to use predictive models which internally *embed* feature selection, such as LASSO regression [46] or decision trees and random forests [56], [58].

Time alignment of cycle-related data/features into multi-cycle time windows is needed because, in slowly-degrading environments, single cycle predictions have a too short horizon. Multi-cycle time-series are normally aligned to a fixed structure, possibly filled by means of padding, sometimes referred to as the *design matrix* [46]. Data from multiple sources (e.g. different physical sensors associated to the same target asset) are also aggregated in this step [6].

2) Labeling: A labeling service is almost invariably present during model building. The most common approach is to

TABLE IV
COMPARISON CHART OF RECENT PREDICTIVE MODELING ANALYTICS APPROACHES FROM SCIENTIFIC LITERATURE.

		Preprocessing			Models				Datasets			
	Ref.	Outlier detection	Feature Extraction	Feature selection	Types	Characteristics	Selection	Update	Origin	Imbal	Use cases	ML Library
Binary FD	[53]		Segment stats	Constants elimination	Ensemble (SVM and k-NN)	Interpretable/ Black-box, Supervised	Cross validation of cost estimate		real	yes	Tungsten filaments for ion implantation	
	[58]			Minimum redundancy	SVM, Decision Tree, One-Class SVM, Autoencoder	Interpretable/ Black-box, Supervised/Semi-Supervised			real	yes	Lathe machines	
	[48]	Clustering	Segment mean, log transform		ARIMA + Naïve Bayes, ANN, SVM, CART	Interpretable/ Black-box, Supervised	Validation set accuracy	Periodic	real	no	Slitting Machines	
	[52]			Correlation, ANOVA	State-space model + Kalman filter	Interpretable/ Black-box, Unsupervised	Custom algortihm		real	yes	Popcorn machines, metal ball conveyor belts, chemical reactors	
	[57]		Segment stats	PCA	K-Means, C-Means, GMM	Black-box, Unsupervised	Elbow method for # of clusters		real	yes	Furnace fans	
	[69]		Segment stats (time and frequency domain)	PCA	K-Means, GMM	Black-box, Unsupervised			real	yes	Furnace fans	R
	[66]		Segment stats	Correlation	SVM, Decision Trees, ANN	Interpretable/ Black-box, Supervised	F1-score, 10-fold cross-validation		real	no	Combustion engine	sklearn
	[61]		Resizing, min-max normalization		CNN	Black-box, Semi-supervised			real, synthetic*	yes	Visual surface quality control	Tensorflow
	[62]	Statistical Thresholds	Segment stats	RFECV, spectral embedded analysis, Laplace score	One-Class SVM, Isolation Forest, Local Outlier Factor	Interpretable, Semi-supervised	Test set F1-score		real	no	Electrified monorail systems (EMS)	sklearn
	[55]			RFECV, Wilcoxon test	RF	Interpretable, Supervised	Validation set precision and recall	Continuous (online RF)	real*	yes	Hard disk drives	
	[63]	Time-series filtering	Segment stats	Correlation	Gradient Boosting, RF	Interpretable, Supervised	Validation set F1-score		real	yes	White goods	MLLib
Multi-Class FD	[56]	Feature filtering	Segment stats	Distribution-based, Correlation	RF, SVM, Logistic Regression, Gradient-Boosting	Interpretable, Supervised	Validation set F1-score	Periodic	synthetic*	yes	Hard disks drives	MLLib
	[54]		Segment mean	PCA, Correlation	RF	Interpretable, Supervised	Test set Accuracy	Periodic	real	yes	Wind turbines	MLLib
	[60]		32x32 patches		CNN	Black-box, Supervised	Test set Accuracy		synthetic*	yes	Visual surface quality control	
RUL Estimation	[46]		Segment stats		LASSO/Ridge Regression	Interpretable, Supervised	Cross validation of cost estimate	Periodic	real	no	Optical emission spectrometers	
	[6]		Wavelet transform, Segment stats, min max normalization	Correlation	Feed-forward NN	Black-box, Supervised			real	no	Vertial milling center	
	[64]		Empirical mode decomposition		TCN, CNN, LSTM	Black-box, Supervised	Test set MSE		real*	no	Engines	
	[68]				LSTM	Black-box, Supervised	Test set RMSE or AUC-PR		real*	yes	Hard disk drives, turbofan engines	sklearn
	[100]		Segment stats	Correlation	Gradient Boosting, RF	Interpretable, Supervised	Validation set F1-score	Concept-drift detection	real	no	Robotics belts	MLLib

formulate RUL estimation and FP/FD as *supervised* or *semi-supervised* learning problems [6], [7], [45]–[49], [53], [54], for which ground truth labels for the input data are necessary during training. Even when an *unsupervised* approach is adopted [52], [56]–[58], however, a smaller number of labels is still needed for *validation*, i.e. assessing the performance of the models.

Ideally, labels would be available for all data. These could be assigned manually by a domain expert, or through some ad-hoc techniques, e.g., generating the dataset in a controlled environment and deliberately injecting faults. In practice, however, this is not always possible, for obvious time and manpower limitations. Therefore, *semi-supervised labeling* services can be provided [66], where classes are automatically discovered by grouping similar data, e.g. through a clustering algorithm, and only a subset of representative data from each group is manually inspected by the domain expert.

3) Training: As shown in Figure 4b, both RUL estimation and FD/FP can be approached either with *supervised* or *unsupervised/semi-supervised* training. In both cases, as shown in Table IV, classical ML algorithms are still the most widely used, especially due to their higher interpretability, which is fundamental for diagnosing errors [56]. However, Deep Learning (DL) is becoming increasingly popular also in this field, especially for unstructured data (e.g. computer vision-based quality inspection) [49], [59]–[61]. One of the main advantages of DL is that it makes most of the feature engineering steps described in Section V-B1 unnecessary, because features are automatically extracted by the model at the outputs of so-called *hidden layers* during training.

Model selection is typically carried on while training. This phase consists in comparing the performance of different models (or different parameters configurations of same model). When large amounts of data are available, model selection can be performed simply splitting them into train, validation and test subset and comparing models on the validation set. In scenarios with small data available, two alternative methods are employed. *Stratified k-fold cross validation* [67] divides the dataset into k folds of equal size, keeping the proportions of the original label distribution in each fold. Then, in k different iterations, this strategy alternatively uses a fold as test set and the other $k-1$ as training set. *Time-series split validation* [67] considers a training window with fixed origin and increasing end. At each iteration, the training window is increased, and the test set is a fixed window right after the training set.

Models are compared using standard metrics, such as Mean Squared Error (MSE) for RUL regression [6], [47] and classification accuracy for FD/FP [53], [54]. However, FD/FP tends to have strongly *imbalanced* classes, since equipment failures or badly manufactured products are generally sporadic and exceptional events. Therefore, sheer classification accuracy might favor models that lean towards predicting all samples as "good". For this reason, other metrics that can take imbalance into account are often considered, such as *precision* and *recall* [54], [68], *Area Under ROC* [61] and *F1-Score* [56].

Supervised training is the most straight-forward approach, in which models are trained entirely using labeled data. For RUL estimation, each training sample is associated with the corresponding asset's RUL, computed based on its actual failure time. For FD/FP, samples are associated with a label specifying if they correspond to an asset that failed within the selected time horizon (or a badly manufactured artifact), and possibly the severity or type of failure. Depending on the type of monitored asset, the number of available raw data sources and the complexity of the problem, a wide spectrum of different models can be employed. These include simple linear and logistic regression [45], [46], [48], Support Vector Machines (SVM) [56], instance-based models such as k-Nearest Neighbors (k-NN) [53] and tree-based models such as Random Forests (RF) [54], [56], [58]. Shallow and deep Artificial Neural Networks (ANN) of various kinds are also common [6], [49], [59]. In particular, standard feed-forward NNs and Convolutional NNs (CNNs) are commonly used when dealing with fixed-length time segments, whereas recurrent NNs based on the Long-Short Term Memory (LSTM) architecture are the most popular choice for variable length time-series. Finally, Temporal Convolutional Networks (TCNs) [64] are recent architectures that combine 1D convolution with causality and dilation to achieve a time-series processing accuracy comparable to LSTMs with higher computational efficiency. An effective and increasingly popular approach consists in using an *ensemble* of models [49], [53], [56]. Ensembling helps by combining the best features of different models. Moreover, for FP, models can be trained to predict with a different time horizon, so that they can yield different precision vs recall trade-offs [53].

Clearly, the most appropriate model or ensemble depends on the task at hand, and there is no *a priori* optimal choice. In most cases, an extensive model selection is therefore required. A good taxonomy of supervised learning algorithms for industrial applications can be found in [7].

Despite its simplicity, the supervised approach has several limitations. First, class imbalance limits the number of available training samples relative to failing equipment or faulty artifacts. Even for RUL estimation, generating labeled training samples requires monitoring the entire life of the equipment until its failure. Second, equipment or product degradation may be due to a multitude of causes, each of which may have a different effect on the monitored features. In other words, the few and incomplete available samples hinder the ability of a ML model to learn the characteristics of *all* patters that should generate a fault alarm or a low RUL estimate.

Unsupervised/semi-supervised training methods are therefore an interesting alternative, although still relatively less explored (as shown in Table IV). In unsupervised approaches, the model infers cohesive and well-separated groups of samples *from unlabeled data*. Under the assumption that the majority of samples correspond to normally behaving equipment and good artifacts, anomalies (outliers) are identified and used to trigger alarms. Indeed, unsupervised FD/FP is often referred to simply as *anomaly detection*. The typical algorithm used in this scenario is a *clustering*, such as K-means, DBSCAN or Gaussian Mixture (GMM) [57], [62], [69].

Semi-supervised techniques are similar, but in this case models are trained using *labeled* data coming only from the "good" (i.e., non-anomalous) class, exploiting the fact that the latter is typically well-defined and labeled samples are abundant and relatively easy to obtain. Then, anomalies are identified as samples that the model assigns a low probability of belonging to that class. The most common semi-supervised models are one-class SVMs and Isolation Forests [62], [68]. However, DL models are gaining traction, especially in fault detection based on images and videos. Deep models can be used either as feature extractors for one-class classifiers [70] or standalone. In the latter case, the most common architectures are *autoencoders* [71].

C. Model Deployment

Model deployment consists in using a predictive model to generate (i.e. *infer*) new knowledge. The main service involved in this phase is therefore dedicated to performing *real-time predictions* on new data coming from the field (see Figure 4).

Conceptually, deployment simply consists in *executing* the model previously selected and trained using one of the strategies discussed in Section V-B on new data. Therefore, the analysis of Section V-B3 on the different problem formulations and the corresponding models applies also here. Notice that the preprocessing phase is also involved in this step, as shown in Figure 4. In fact, features must be extracted and correctly aligned exactly as during training before running a prediction.

The main differences between model building and deployment are essentially *technological*. In the latter phase, predictions are executed on *streams* of data, with minimum latency as the key objective, in order to enable timely reactions to alarms triggered by the model. Therefore, this phase requires different features from the underlying analytics platform with respect to model building, which is an offline batch-based task. Moreover, the two phases may differ also in the *location* where analytics are performed.

The most common approach is to deploy the entire analytics pipeline in the cloud. However, while model building can be conveniently left in the cloud, partially or totally delocalizing deployment at the edge provides several benefits [7], [49], [72]–[75]. Indeed, executing predictive models at the edge reduces the traffic on the plant's network infrastructure, avoiding the transmission of large amounts of raw data. In turn, this yields lower prediction latency and consequently faster reaction times [72]. Moreover, the transmission of large data sizes from in-field edge devices to the cloud also has a high *energy* cost, which can be significantly reduced by doing prediction at the edge [74]. Last but not least, to infer information from a small bunch of localized data gets much more efficient than playing off-line on a cumbersome data base. This is the motivation for so-called *edge computing* or *edge analytics* paradigms [72]–[75].

Clearly, deploying a model using edge computing requires an additional set of tools and technologies, e.g. allowing to convert ML models designed for cloud execution so that they can be run on resource-limited edge devices. These conversion steps range from reducing the number of parameters in the model to using compact data representations (quantization), or to determine whether *partitioning* the model between edge and cloud is more convenient than pure edge computing [72]–[74], [76]. A comprehensive list of conversion techniques for edge ML is reported in [74], with particular focus on highly complex deep models.

D. Model Updating

A last fundamental yet still partially unexplored functionality in an industrial predictive analytics pipeline is *Model Updating*. This consists of exploiting newly available information from the field to ensure that the deployed model keeps providing the required performance over a long time-span [7], [46], [54]–[56]. Updating can be either periodic or triggered by a measure of model *degradation*. These are also referred to as *passive* and *active* updating respectively.

Passive updating, in a data-driven setting, boils down to a standard *online learning* (or *incremental learning*) strategy [46], [55]. This is particularly useful in combination with supervised learning approaches. In fact, given the aforementioned scarcity of data labeled as "bad", each new asset failure or faulty product represents a precious source of new data that should be fed to the model. However, periodic updating is also useful in unsupervised and semi-supervised settings, as sensor data distributions invariably tend to change over time, even in absence of faults, due to variations in external conditions, equipment settings, etc (so called *concept drift*) [77]. In [56] a passive model updating approach is proposed where the best performing algorithm from a set (including RFs, SVMs, logistic regression and others) is periodically selected by performing re-training and k-fold cross-validation on newly available data.

Active updating is triggered only when a measure of model degradation exceeds a pre-defined threshold [7], [77], based on the observation that passive solutions are often unnecessarily computationally intensive, as they require periodic re-training even when data distributions did not change. Degradation metrics are either based on comparing the distribution of new data with training samples or on assessing the quality of the model's outputs [7]. In [77], an unsupervised degradation metric is proposed, based on measuring negative changes in the intra-class cohesion and inter-class separation obtained by the model for new data. The rationale is that concept drift can be identified when the deployed model starts to poorly separate new data into classes compared to its results on training samples. When this degradation, measured by the *silhouette*, exceeds a threshold, model updating is triggered. This method can also identify previously undetected classes of data and consequently request a new (semi-supervised) labeling iteration, as shown by the feed-back arrow in Figure 4b.

From a technological point of view, model updating involves the same algorithms as model building, hence sharing most of its characteristics (i.e. batch-based, throughput-centric processing). While re-training is typically performed in the cloud, model degradation evaluation may benefit from edge computing, as discussed in Section V-C.

E. Enabling Technologies

A limitation of the current literature on industrial descriptive and predictive data analytics is the lack of focus in integrating innovative and articulate pipelines with existing big data architectures. This can also be seen in Table IV: many works do not even mention the ML library used to build their pipeline, and even fewer refer to specific big data architectures. While there is a separate large body of literature on such architectures, bridging the gap between these two worlds is still an open issue. In this section, we try to summarize the main underlying technologies for industrial data analytics, based on our experience with real industrial use cases.

The de-facto standard framework for implementing data-driven analytics on a distributed cloud platform is MLlib [31], Apache Spark's machine learning library, which provides distributed implementations of most relevant classic ML algorithms for predictive and descriptive analytics mentioned in the previous sections.

While MLlib mostly offers classical ML models, for implementations of deep learning algorithms, TensorFlow [78] and

PyTorch [79] are currently the two dominant frameworks. Despite some API differences, they offer similar features allowing fast and efficient design, training and deployment of extremely complex DL models. Most importantly, they optimize and abstract computations on multi-dimensional tensors on CPUs, GPUs or specialized Tensor Processing Units (TPUs), hiding from users details such as the automatic back-propagation of gradients required for training. Moreover, they also offer vast libraries of pre-designed implementations of popular layers or even entire NN architectures. Both frameworks offer their own distributed training functionality for cloud clusters. Alternatively, there also exist several community-driven projects allowing the execution of TensorFlow or Pytorch models on Spark, such as [80].

TensorFlow Lite and PyTorch Mobile are lightweight engines for the execution of deep learning models on *edge* devices. With respect to their cloud counterparts, these stripped-down versions only allow inference (i.e. deployment) and not training, and reduce overheads to a minimum in order to support execution on resource-limited devices such as single-core CPUs or even microcontrollers. Moreover, they offer conversion tools to make standard models for cloud edge-compatible, using the transformations described above.

For simpler ML models and for performing other types of data transformations (e.g. pre-processing steps), Node-RED is very popular in the edge computing world [81]. This tool offers a simple visual interface for designing pipelines using flow-based programming, and supports a large library of pre-designed nodes for ML and analytics, as well as interfaces with various APIs for communication with the cloud.

VI. Application Layer

Within the data-centric framework of industry 4.0, the application layer includes a rich set of applications allowing the user communities (e.g., business analysts, physical scientists, manufacturers, and decision-makers) to effectively exploit combined data-centric services to obtain interesting insights on data collected from the field and to transform these insights into business actions. Each kind of user needs only one subset of data/insights, those relevant for his role within the manufacturing context, allowing the user to make informed decisions that rely on a good perception of the current industry conditions.

Although the spectrum of deployable applications within this layer is virtually infinite, many rely on some common components. One of them is certainly represented by *visualization techniques*, which help humans to correctly interpret, interact, and exploit data and its value. Keeping the user in the analytics loop by leveraging human visual perceptions on both intermediate and final data analytics results is an efficient way to achieve good decision making [82]. To this aim, we focus the discussion of this section on *data visualization issues, techniques and enabling technologies*.

According to Keim and Kriegel [83], data visualization techniques can be classified into five different groups: geometric (e.g., scatterplots, landscapes, hyperslide, parallel coordinates), pixel-oriented (e.g., recursive pattern techniques, spiral and axes techniques), icon-based (e.g., shape-coding, color icons), hierarchical (e.g., treemap, infocube), and graph-based (e.g., basic graphs such as straight-line, polyline, or specific plots) techniques. Furthermore, arbitrary combination from the above techniques is allowed and named hybrid techniques. Selected subsets of visualization techniques are used to develop application services including (1) *informative dashboard*, (2) *report generation*, (3) more-advanced *data exploration* based on visual representations, and (4) *augmented-reality(AR)-based applications*. While (1) and (2) are mainly fed by the aggregation and filtering service, the last two are mainly based on more advanced data analytics algorithms (e.g., descriptive versus predictive analytics relying on data mining approaches and machine learning algorithms, as described in Section V).

Both informative dashboards and report generation services are usually designed with the final goal of presenting data and insights using high-quality visualization techniques to effectively support the decision-making process. While informative dashboards exploit an interactive process, reports rely on a static presentation. Specifically, informative dashboards usually provide interactive data exploration through the navigation of different kinds of chart images at different data granularity levels, as explained in [84], supporting users in quickly gathering a broad insight of their dataset. A variety of visualization options are exploited including (i) traditional plots as pie, bar and line charts, (ii) more-complex graphics such as heat and tree maps, geographic maps, scatter plots and (iii) new data/application-driven visual representation defined ad-hoc based on the key aspect of the application or on specific data properties. Real-time monitoring of manufacturing processes are critical success factors in the smart factory. To address this issue, the authors in [85] proposed a Mobile Manufacturing Dashboard, to show a subset of useful and actionable items to both shop floor workers and production supervisors based on the manufacturing context where the user is located. Subsets of graph charts can be combined to create a variety of (static) reports tailored to the roles of decision-makers (i.e., a report is often designed for a specific business user). The corresponding visualization services periodically generate and distribute a set of reports to different business users. Reports usually exploit grid-layout, are multipaged, and include some relevant chart images to capture the decision-makers' attention on the most relevant information that might be of interest and influence any final business decisions. According to this, in the high volume automotive industry context, the authors in [86] proposed innovative visualization techniques to compactly show manufacturing sequence data to manufacturing management, supervision and operations personnel.

More advanced, goal-oriented visual data exploration services also exist: (a) explorative analysis which is mainly based on an interactive process with the final goal of finding motivations and recurrent trends and to provide data hypotheses, showing intermediate analytics results step-by-step; (b) confirmative analysis starts with data hypotheses and has the objective of confirming/rejecting such hypotheses through a goal-oriented examination process.

A. Enabling Technologies

Over the last years, different research and business efforts have been devoted to designing and developing tools to visualize insights for different target users, thus empowering the decision-making process through visual representations. Different analytics and visualization tools are available today, many of them as commercial products offered by a variety of worldwide vendors (e.g., Tableau, Qlik, Microsoft, SalesForces, SAP, SAL, Oracle). Recently, in February 2020, Gartner published the Magic Quadrant for Analytics and Business Intelligence Platforms [87] to compare the key functionalities offered by IT-commercial products and declared that available analytics platforms are no longer differentiated by their data visualization capabilities, of which the data reporting and generation of informative dashboards are becoming commodities. The key sources of competitive differentiation are mainly based on analytics services which efficiently generate and explain insights automatically extracted from data. Beyond different available commercial platforms and tools, we are witnessing the widespread use of high-level and efficient data-science oriented languages such as Python and R. Both languages enable the easy development of step-by-step data analytics workflows enriched with different visual representations to report useful insights. These two languages are mainly exploited to design and develop more-advanced visual data exploration services relying on analytical notebooks, to visually describe the analysis outcomes, enable reproducibility, and help to share results allowing the target communities to understand properties of the data type, their complexity, and insight characteristics, patterns discovered from data.

Data visualization benefits from developments in technologies that provide novel strategies for presenting complex and heterogeneous data. These have extensive applications in communicating the actionable information in manufacturing data useful to support decision making. In [88] authors discuss the visualization functionalities, applications, and tools available in literature, map these to manufacturing data presentation requirements, and discuss the open issues by proposing some interesting development paths. The issue of visualization in industry 4.0 is also addressed in [89], that presents a review of the visualization techniques and tools tailored to smart manufacturing applications. In the same context, of considerable importance is also the contribution made in [90], where authors present a set of tools to monitor processes in (near) real-time by highlighting interesting insights. More specifically, the authors focus on the usage of simulation and visualization techniques to jointly show the physical and non-physical environments of the machine.

VII. Deployment of the Architecture: A Use Case in the Robotics Industry

As a first deployment example of the architecture of Figure 1 we report our experience in designing and developing a set of data-driven services for predictive maintenance in the context of the robotic industry.

Specifically, five integrated analytics services have been designed as briefly described below and detailed in the next subsections.

There are many challenging aspects to consider in the maintenance of a robotic machine. We focused in particular on the maintenance of the axis belt tension, which is in charge of the power transmission from the engine to the adaptor. An overly low tension causes more skidding and wearout; on the other hand, an overly high tension is source of stress on the components and hence causes overheating. We thoroughly discussed with robotics engineers and manufactures in order to characterize the main industrial needs to be addressed by means of a data-driven methodology. A number of research issues arose; for each of them a proper data-driven analytics service has been designed with the final aim to provide a complete and automatic data analytics pipeline (as shown in Figure 4a) able to infer the main relationships among robotics data cycles (in terms of belt tension), automatically capture variations over time, and correctly react to the data changes. The proposed pipeline can be also used for different robotics components or tailored to more complex robotic activity. In addition, the estimation of the remaining useful life of the robot is also provided with an additional data-driven service although it slowly degrades over time.

Subsequently we will discuss each of these research issues and the relative services designed to manage it; the latter map directly onto services discussed in the previous sections.

The variations in raw data are useful to monitor detailed time-based variations but not relevant to capture trend variations over time. Raw data are collected every two milliseconds to constantly monitor the robot activities, while each robot cycle lasts 24 seconds. The fine-grained collected data is relevant for the robot monitoring, while coarse-grained feature-based statistics could be more useful to capture trend variations over time in cyclic industrial processes, whose duration may vary.

To this aim, we implemented a customized *Feature engineering* service to transform raw data (a time series) collected from industrial plants into a set of statistics-based features used as inputs to model the relevant aspects characterizing the problem under analysis.

The robot performance is also affected by different external conditions (e.g., the temperature and humidity within the plants, or other environmental settings of the working place) that cannot be monitored, and hence remain unknown. In other words, the same robot working under different conditions may perform differently in heterogeneous conditions. This aspect does not negatively effect the overall robot performance that continues to correctly perform its tasks, but the set of fine-grained raw data collected may have distribution differences. These differences represent an offset with respect to the baseline of another robot working in a different environment and are present over the lifecycle of the robot without causing any negative effect of the robot activity (i.e., data distribution characterizing the ground-truth knowledge may slightly vary based on external parameters). Thus, to indirectly capture the external effects of environmental conditions, the modeling of statistics-based features distribution is needed as it may vary based on the external working settings where the robot is positioned to work. Thus, the ground-truth knowledge (i.e., class labels) of the

phenomenon under analysis (e.g., belt tension) is not always immediately available and may vary from one manufacturing plant to another. To this aim, the *Semi-supervised data labeling* service has been designed to support the domain expert to semi-automatically characterize the baseline robot cycles and correctly define prediction labels (e.g., low belt tension versus high-level tension) by automatically analyzing the data-driven robot cycles.

The predictive maintenance service is required to estimate the belt tension value cycle by cycle. To this aim, a *predictive analytics* service has been designed and developed. It learns from a set of historical robot cycles the main relationships among features-based statistics computed on each cycle and the corresponding level of belt tension (i.e., event to predict, discovered through the semi-supervised data labeling service).

Once the model is built, labels for new incoming robot cycles are forecast by the *real time prediction* block.

The changes in the distribution of the collected data over time affect the performance of the predictive model that will not work as expected. In real-life settings, data characterizing robotics cycles may change over time due to different factors affecting the belt degradation. The industrial need is to automatically capture concept-drift on collected real-time data and trigger the building of a new predictive model. To this aim, the *Concept-drift detection* service has been designed (corresponding to the *Model updating* service in Figure 4a). It uses a scalable version of a state-of-the-art unsupervised index (i.e., Silhouette), which estimates both the cohesion and separation among groups of data.

An important requirement is to estimate the Remaining Useful Life (RUL) by analyzing data collected from different phases of robotic activities. Although the robots useful life reduces very slowly, it is essential to estimate the RUL to promptly react when the estimated degradation becomes larger than expected. To this aim the *RUL estimation* service (an instance of the Training and real-time prediction services in Figure 4a) has been designed to analyze data collected from different phases of an operational life asset in order to identify deviations from nominal operational profiles.

The remainder of this section is devoted to the description of the implementation details of services created to solve the above issues.

A. Feature engineering

The Feature engineering service transforms each raw time-series into a corresponding time-independent set of features, able to capture trend variations over time. In more detail, each time signal is divided into several segments (the size of the segments is a user-defined parameter). Then, for each of these segments, a set of statistics is computed to summarize the time series trend. The considered statistics include mean, standard deviation, quartiles, Kurtosis, Skewness [63]. Once all these features have been calculated, those with little informative content are dropped, while those more significant are kept in the next analytics steps. The multicollinearity-based approach [91] has been exploited for this purpose. Based on a given threshold (user-defined), all the attributes whose values can be inferred by a multiple regression model of the other attributes, are automatically disregarded.

B. Semi-supervised data labelling

The aim of this service is to help the domain experts assign class labels to production cycles and effectively support predictive maintenance approaches relying on supervised methodologies (i.e., classification algorithms and regression techniques) in the absence of standard ground-truth knowledge (i.e., class labels) on equipment failures. It exploits a data-driven methodology relying on the most advanced clustering algorithms to automatically discover groups of data sharing common properties without requiring a-priori knowledge. The service integrates different *state-of-the-art clustering algorithms* (the K-means algorithm [92], the Bisecting K-Means [93] and the Guassian Mixture [94]) and a *self-tuning strategy*. While clustering discovers a set of groups of production cycles characterized by similar properties given a specific setting of input parameter, self-tuning is used to (i) automatically discover the optimal input parameter setting for each algorithm and (ii) select the algorithm by finding the optimal partition tailored to the data under analysis.

The latter is based on the evaluation of the Silhouette index [95], which measures how similar a production cycle is to its own class (cohesion) compared to production cycles in other classes (separation). Thus, it evaluates the appropriateness of the assignment of a production cycle to a class rather than to another one. The Silhouette values range in $[-1, 1]$, where -1 represents a bad assignment of the production cycle in its group, and 1 describes a good assignment. Each group is then locally characterized by means of a few statistics to help the domain experts easily understand each cluster content separately and promptly assign class labels.

C. Predictive analytics

The predictive analytics services rely on two different steps: *model building* and *real-time prediction*. Based on an existing model, real-time predictions assign a label to every new incoming data. It runs at the edge, namely on the machinery, providing fast predictions on fresh data. Instead, the model building service extracts the prediction labels using historical data to detect latent relations with the collected data. Two ensemble learning classification algorithms were integrated in the platform: Gradient Boosted Tree Classifier [96] and Random Forest [97]. According to the F-score metric and using *stratified k-fold cross validation*, the most efficient is automatically selected.

D. Concept-drift detection

In industrial environments the collected data may change over time due to equipment degradation or some unknown (environmental) factors. The concept-drift detection service automatically checks for deviations which trigger the retraining of the model using the new data. This analytics service detects changes over time in the data distribution to understand

when a predictive model is no longer performing as expected. Since the ground truth label is never available in near-real time, an unsupervised strategy has been integrated. It relies on a new and scalable index, i.e., Descriptor Silhouette (DS) proposed in [98] able to assess both cohesion and separation between groups of data. DS estimates the Silhouette index [95] for each record under analysis and the curve formed by all sorted DS scores provides a global overview of cohesion and separation for all records examined. Given a labeled dataset, the DS curve allows measurement of the cohesion and separation of both training data and new incoming data.

The predicitve model degradation is computed for each class c and at timestamp t. It is estimated by using the *Mean Arctangent Absolute Percentage Error* (MAAPE) [99] between the DS scores evaluated on training data and the DS scores obtained with all the data collected until time t (including the training data). To give a fair comparison the latter is properly sampled. The degradation for a class c is weighted by the number of production cycles predicted to belong to c

w.r.t. the cardinality of all collected data [98].

E. RUL estimation

The RUL estimation service estimates the remaining useful life of industrial machinery using a data-driven approach. On start up, the machine works with an ideal (or nominal) operational profile, therefore collected data are representative of the correct behavior. During machine use for production purposes, degradation gradually starts to deteriorate its behavior. The difference between data collected at the beginning of the machines life and the data collected later can be used to estimate the remaining useful life. Furthermore, the degradation of equipment usually increases over time. Based on these observations, we propose to estimate the machine RUL by considering (1) the *joint probability* of the selected feature values to belong to the nominal operational distribution, and (2) the presence of *probability drifting* in the collected data over time

since there exists a complementary analogy between the RUL and the predictive model degradation [100].

F. Experimental settings

The proposed data-analytics pipeline with the five services described previously were tested for a period of six months on three RobotBoxs similar to those described in [100]. A RobotBox is composed of an engine, an adaptor, a rubber belt for the transmission, and a weight of five kilos to simulate a realistic application. Experiments on the RobotBoxes placed in different manufacturing environments demonstrated semi-supervised data-labeling are actually needed to correctly model the feature-based distribution of the referenced profiles slightly affected by environmental parameters that are not directly monitored. The predictive model service, trained on labels discovered through semi-supervised data labeling, provides accurate predictions of the belt tension achievinga F-score higher than 97%. During the monitoring period, and separately for each RobotBox, the concept-drift service detected data-drift only once, and the retraining of the model has been performed accordingly, after new labels had been discovered through the semi-supervised data labeling. The Remaining useful life service estimates the RUL value weakly, and results revealed to be in line with expectations. Additional details for each analytics service are provided below.

G. Deployed technologies

Figure 5 shows the technological choices made to deploy the reference architecture described in Section II in the context of a robotics industry. For each functional layer the logo of the selected technology is shown and grey rectangles model the kind of deployed services.

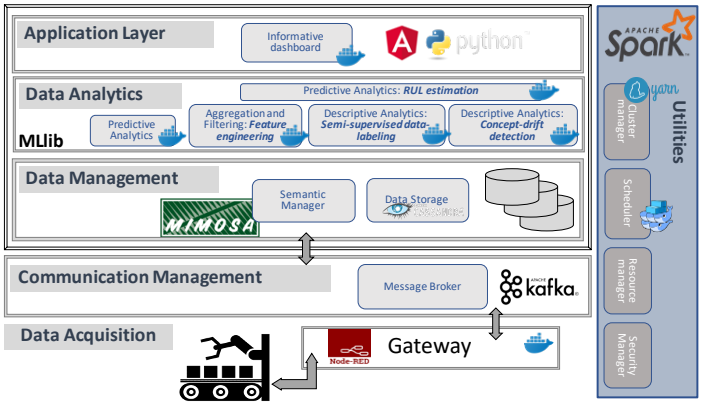


Fig. 5. The proposed architecture in the context of the robotics industry use case.

In the context of the data analytics layer, we deployed five services, each one addressing a specific task: two of them *Predictive analytics* (including model building deployed in the cloud and real-time prediction at the edge) and *RUL estimation* are examples of the tasks belonging to the *Predictive analytics* service shown in Figure 1, while *Semi-supervised data labelling* and *Concept-drift detection* are examples of tasks categorized as *Descriptive analytics* in Figure 1, and *Feature engineering* is an example of task in the context of *Aggregation and Filtering* service in Figure 1. Furthermore, the deployed analytics services address all the main industrial analytics tasks shown in Figure 4a and described in Section V. The predictive analytics service is composed of two services: *model building* deployed in the cloud and *real-time prediction* deployed at the gateway (Data acquisition layer in Figure 5).

The services have been deployed on top of a lightweight and flexible micro-services architecture, in a container-based environment using Docker 17.09.

The data flow engine at the edge level is developed on Node-RED [81]. The edge application collects raw sensor data from the field, performs feature engineering from the raw time series data and encodes all information into JSON-LD [101] (an extension of JSON supporting Linked Data). The JSON-LD file is then forwarded to the message broker, developed through Apache Kafka 1.0, for further distributions to the other cloud services. Cloud resources are exploited instead to perform data management, advanced data analytics, and data visualization as well. Cassandra 3.11 and MIMOSA have been

selected for data storage and semantic manager respectively. As a machine learning library, MLlib has been chosen [31], while to develop informative dashboard keeping the user in the analytics loop we used Python and Angular (an open-source web application framework) [102]. Finally, all services are based on Apache Spark [29], along with its streaming extensions [30], and YARN [103] as cluster manager.

H. Extension to other use cases

While the proposed data-analytics architecture of Figure 5 is specifically meant for a robot application, it can be easily adapted and applied to other scenarios. We are currently working in this direction for similar maintenance problems in the white-goods industry and on a rolling mill machine. In the former case, two chemicals are injected by a nozzle to produce foam, and the process data is collected to predict possible alarms and failures in the injection process itself [63]. The ultimate goal is to predict in advance nozzle failures within a proper time horizon and link them with the corresponding production cycles.

The second application addresses the trailing arms production and the rolling mill machine which might affect the quality and yield of production. The existing preventive maintenance plan requires a biweekly downtime of approx. 120 minutes for component replacements. The goal is to estimate the wear of a rolling mill machine at different stages of life and usage. To this aim, additional data are collected, such as position, straightness, length, etc. The data analytics pipeline aims at automatically extracting correlations of different measurements of the machine's condition, providing maintenance alert that suggests the upcoming need for segment replacement.

VIII. Deployment of the Architecture: A Use Case in the Food Supplement Industry

The second use case concerns the real-time monitoring of a production line with the objective of generating informative dashboards to visualize a set of Key Performance Indicators (KPIs).

We focus in particular on the Overall Equipment Effectiveness (OEE) index, a widely used metric that has become a standard KPI in manufacturing. OEE gives insights on how to enhance manufacturing processes by eliminating waste and identifying losses. OEE ranges from 0 to 100% and estimates the ratio of productive manufacturing time. A score of 100% signifies that the manufacturing line is producing at maximum speed (100% performance), with excellent material components (100% quality), and without breakdowns (100% availability). The outcome of the OEE computation is relevant for both production managers and shift supervisors on production lines. The former are mainly interested in the estimation of the OEE index over time, as they must guarantee the highest level of production. The latter, conversely, are mainly responsible for production line inefficiencies, thus they need to assess the OEE to identify unforeseen issues and promptly react to remove the causes.

The state-of-the-practice approach for OEE calculation relies on a *delayed* extraction carried out when all the data are available (e.g., at the end of a production shift or at the end of the production order); in this context, we introduced a new strategy to compute a *dynamic OEE every ten-minutes of machinery production* to provide interesting insights on the efficiency and effectiveness of the production lines in near-real time. The availability of a set of OEE values as opposed to a single aggregate one allows manufacturing operational roles to promptly react to production inefficiencies and easily identify their main causes.

The informative dashboard is therefore updated every ten minutes and includes a section for each production line, order, and shift. For example, the *shift* section includes four graphic objects: (1) a combined graph plotting the trend of the dynamic OEE computed every ten minutes with respect to the traditional OEE computed at the beginning of the shift; (2) a pie plot showing the target OEE with respect to the last OEE value computed; (3) a graph comparing the dynamic OEE over time with the trend efficiency obtained by the same shift, production line, and order on the day before; (4) a graph comparing the dynamic OEE with the best trend efficiency obtained by a given shift since the begin of the production of order on the same production line.

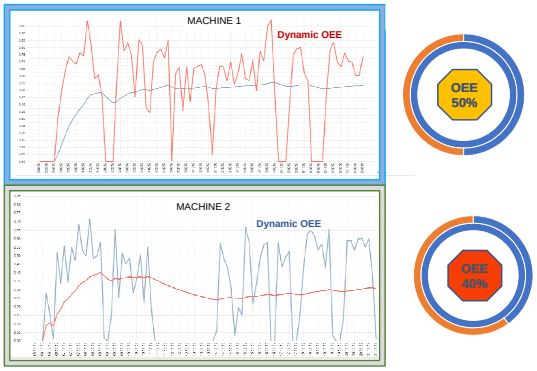


Fig. 6. A portion of OEE Dashboard.

As an example of the visual information provided by the informative dashboard service, Figure 6 shows for a given shift the 10-minute dynamic OEE tracking for two machines with respect to the aggregated one (i.e., traditional OEE computed since the begin of the shift). It is interesting to see that the dynamic OEE exhibits strong oscillations, in some cases reaching very low scores that are well below the aggregated one. This may indicate a possible anomaly in the production that will likely not be detected by an aggregated indicator, which, in particular for the Machine 1, stays to pretty constant and acceptable levels.

The pie plot shows the target OEE (100% in this case) with respect to the last OEE value computed, separately for each machine. This indicator highlights in red the required improvement in terms of OEE to reach the target value. The inner hexagon is coloured in green, red, or yellow based on the difference between the target and real value to easily capture the attention of the user.

A. Technological solution

The designed system has been deployed in a production plant in a food supplement industry. It was initially applied to a production line including four machines equipped with different sensors. As a proof of concept, the cloud computing data processing engine includes a cluster of two nodes configured with Apache Kafka 1.0 as a message broker (*Communication layer* in Figure 1), Apache Spark 2.2.0 (*Utilities* in Figure 1) as a data processing engine, Cassandra 3.11 as a data storage (*Data management* in Figure 1), Python and Angular as a generator of an informative dashboard (*Application-level services* in Figure 1). All services are deployed in a containerized environment by using Docker 17.09.

The proposed architecture, deployed on-premise, takes care of only of the cloud aspect, and allows data collection through the *Message broker* on the monitored manufacturing environment or selected production lines and equipment. We selected Kafka as a message broker to reliably route a virtually unlimited number of sensor measurements and log events from heterogeneous manufacturing-plant data sources at different rates. The *Querying* service of Figure 1 is used to retrieve the data to compute the dynamic OEE. The computation is performed via the *Data Aggregation and Filtering* analytics service, and is separate for each production order, production line machine or production shift as described above. Both the execution of the querying task and the long-term storage of the collected data are performed by Cassandra. The OEE values are then shown in an informative dashboard every ten minutes, separately for each production order, for each machine within the production line, and for each production shift. The dashboard is the application level service keeping the operational users in production plants continuously informed on the OEE evolution of machines over the desired granularity (order, machine, shift) to allow them to promptly react to the possible anomalies.

IX. Open issues and future research directions

Over recent years we have witnessed the diffusion of a rich set of technological advances to support the data powered Industrial revolution. As discussed in this paper, beginning with industrial plants full of Internet of Things technologies, a huge amount of data is being generated and transmitted to data-centric architectures with a high expected value to increase business's competitiveness. Innovative data management and analytics methodologies able to translate data into knowledge to effectively support the decision-making process and to turn such data-powered knowledge into real value for the market and the society are continuously investigated. Despite a wide range of technological software advances discussed throughout this paper, there is still room to enhance and facilitate the dissemination and exploitation of data-centric/powered methodologies and related architectures.

Some open problems, which can be seen both as challenges and opportunities at the same time are summarized below. This should not be interpreted as an exhaustive enumeration of all possible open problems, which would be impossible, but rather as a guide to some of the most relevant future research directions. In particular, the section focuses mostly on the level of data analytics (see Section V), because it is at this level what the Industry 4.0 scenario poses most of its domain-specific open issues. The points listed below should be addressed to make the best use of the available methodologies for value creation from data in the manufacturing context, and to allow subjects without specific expertise on data techniques to use data-centric architectures autonomously.

Self-configuration of algorithms and automatic selection of the best solution. The performance of data analytics algorithms often depends on the configuration of the input parameters. Furthermore, there exist several state-of-the-art approaches that can be exploited to support a specific analytics task in the manufacturing context. Both tasks are mainly based on the analyst's experience and an intense experimental activity is always required. Innovative and effective techniques that can intelligently support the analyst in the automatic identification of the best algorithm, properly and automatically configure it, for the use case of interest could significantly increase the spread of the data analysis techniques in production and industrial contexts.

Self-evolution of predictive models. Data-driven predictive maintenance models have a limited temporal validity given that the data collected in the production context may vary over time in an unpredictable way. Only a few solutions exist to define when and how a predictive model must be updated. Alternative and automatic mechanisms capable of assessing the accuracy of the prediction together with the degradation of the prediction model are necessary in order to ensure autonomous updates. Models should independently evaluate themselves and decide when to update themselves by starting a new training using the most recently collected data. Thus, innovative approaches providing self-evolutionary functionality of the predictive models are widely needed.

Interpretable prediction models. The predictive algorithms of greatest interest in Industry 4.0 applications are those that generate interpretable models. However, for some very complex applications and for those requiring the analysis of unstructured data such as images, videos or sounds, it may be necessary to use complex algorithms (e.g., neural networks) that generate opaque models (black-boxes), to ensure a higher level of accuracy. New strategies capable of explaining the internal mechanisms of opaque algorithms (mainly relationships between the input data and the predictions generated as outcomes) are necessary to facilitate the diffusion of the more complex and often more accurate but opaque predictive models in Industry 4.0 applications.

Class imbalance. Historical data collected from manufacturing contexts often present many samples of ordinary behavior and a small number of data associated with failure events. This situation, which is desirable in practice as it describes a healthy operating system, becomes critical for the application of classic machine learning techniques. In fact, the latter must analyze a lot of data with examples of faults in order to learn patterns useful for prediction. Innovative techniques along with special precautions are therefore necessary in the training phase of traditional models to let them correctly manage and identify unexpected situations not encountered during training.

Privacy-preserving data-driven methodology. Since the first deployments of data-driven solutions in intelligent manufacturing, privacy issues have always been critical aspects to be addressed in order to guarantee the right level of data protection. The preliminary studies on Federated Machine Learning algorithms [104] seem a promising direction to realize privacy-preserving data strategies, while showing comparable performance to the traditional approach. However, many research issues are still open to effectively evaluate and integrate such methodologies in the intelligent manufacturing context.

In perspective, **interactive data visualization** will become more adaptive and personalized to guide the manufacturing users in the visual analytics process and in easily detecting data changes. The new era of visual analytics systems should recommend which queries and views of the data the user might need to analyze and what data granularities might better help one to make the final decision. The new services should include *data storytelling* capabilities, i.e., provide interactive data visualizations combined with narrative techniques which deliver data stories and explanations that can be easily understood by the human user.

Finally, to effectively support the technological transfer of innovative data analytics solutions and their easy integration in data-centric architectures, new data-analytics approaches should be released as deployable units to be easily tested and adopted in manufacturing contexts. The release of the source code is useful for the research community but not effective for industries. An **open-access database of deployable data-driven services** tailored to manufacturing contexts would be profitably exploited most by interested industries. The project should at least define the submitted policies, the APIs, the management rules of updated releases and should be supported by cross-disciplinary research communities with the common objectives to transfer knowledge to manufacturing companies and to share new solutions able to continuously answer manufacturing needs.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the following programs.

- European Commission under the H2020-IND-CE-2016-17 program, FOF-09-2017, Grant agreement no. 767561 "SERENA" project, VerSatilE plug-and-play platform enabling REmote predictive mainteNANce.
- POR 2014-2020 of Piedmont Region (Italy), FESR (European Fund of Regional Development), and MIUR (Ministry of Euducation, University and Research, Italy) under the program "Fabbrica Intelligente" (Smart Factory), Action 3, "DISLOMAN" project, Dynamic Integrated Shop fLoor Operation MANagement for industry 4.0.

REFERENCES

[1] B. Peccarelli, "Bend, don't break: how to thrive in the fourth industrial revolution. In World Economic Forum Annual Meeting," https://www.weforum.org/agenda/2020/01/the-fourth-industrial-revolution-is-changing-all-the-rules/, 2020.

[2] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," ACM Computing Surveys (CSUR), vol. 51, no. 6, pp. 1–29, 2019.

[3] A. Talaminos-Barroso, M. A. Estudillo-Valderrama, L. M. Roa, J. Reina-Tosina, and F. Ortega-Ruiz, "A machine-to-machine protocol benchmark for ehealth applications–use case: Respiratory rehabilitation," Computer methods and programs in biomedicine, vol. 129, pp. 1–11, 2016.

[4] A. Homay, A. Zoitl, M. d. Sousa, and M. Wollschlaeger, "A survey: Microservices architecture in advanced manufacturing systems," in 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), vol. 1, 2019, pp. 1165–1168.

[5] R. Sahal, J. G. Breslin, and M. I. Ali, "Big data and stream processing platforms for industry 4.0 requirements mapping for a predictive maintenance use case," Journal of Manufacturing Systems, vol. 54, pp. 138 – 151, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0278612519300937

[6] J. Yan, Y. Meng, L. Lu, and L. Li, "Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes, and Applications for Predictive Maintenance," IEEE Access, vol. 5, pp. 23 484–23 491, 2017.

[7] M. Compare, P. Baraldi, and E. Zio, "Challenges to IoT-enabled Predictive Maintenance for Industry 4.0," IEEE Internet of Things Journal, p. 1, 2019.

[8] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. S. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," Computers & Industrial Engineering, vol. 137, p. 106024, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835219304838

[9] "I4.0 Standards," http://i40.semantic-interoperability.org, accessed: 2020-04-20.

[10] "FiWare," https://www.fiware.org, accessed: 2020-04-20.

[11] "The Open Platform for the Internet of Things," https://sitewhere.io/en/, accessed: 2020-04-20.

[12] "Linksmart open-source IoT platform," https://linksmart.eu/, accessed: 2020-04-20.

[13] "iottly IoT Platform," https://iottly.com/, accessed: 2020-04-20.

[14] "Eclipse kapua IoT Platform," https://www.eclipse.org/kapua/, accessed: 2020-04-20.

[15] "Predix Platform," https://www.ge.com/digital/iiot-platform, accessed: 2020-04-20.

[16] "Industrial IoT Platforms," https://www.gartner.com/reviews/market/industrial-iot-platforms, accessed: 2020-04-20.

[17] E. Patti, A. Acquaviva, and E. Macii, "Enable sensor networks interoperability in smart public spaces through a service oriented approach," in Advances in Sensors and Interfaces (IWASI), 2013 5th IEEE International Workshop on, June 2013, pp. 2–7.

[18] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," ACM CSUR, June 2003.

[19] P. Chandra and M. K. Gupta, "Comprehensive survey on data warehousing research," International Journal of Information Technology, vol. 10, no. 2, pp. 217–224, Jun. 2018. [Online]. Available: https://doi.org/10.1007/s41870-017-0067-y

[20] U. Aftab and G. F. Siddiqui, "Big data augmentation with data warehouse: A survey," in 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 2785–2794.

[21] N. Tuptuk and S. Hailes, "Security of smart manufacturing systems," Journal of Manufacturing Systems, vol. 47, pp. 93 – 106, 2018.

[22] M. R. Asghar, Q. Hu, and S. Zeadally, "Cybersecurity in industrial control systems: Issues, technologies, and challenges," Computer Networks, vol. 165, p. 106946, 2019.

[23] M. Lezzi, M. Lazoi, and A. Corallo, "Cybersecurity for industry 4.0 in the current literature: A reference framework," Computers in Industry, vol. 103, pp. 97 – 110, 2018.

[24] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a smart city internet of things platform with microservice architecture," in 2015 3rd International Conference on Future Internet of Things and Cloud. IEEE, 2015, pp. 25–30.

[25] M. Fowler and J. Lewis, "Microservices," http://martinfowler.com/articles/microservices.html/, 2014, accessed 2020-04-20.

[26] S. Newman, Building microservices: designing fine-grained systems. " O'Reilly Media, Inc.", 2015.

[27] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," Linux J., vol. 2014, no. 239, Mar. 2014.

[28] A. Oussous, F.-Z. Benjelloun, A. Ait Lahcen, and S. Belfkih, "Big data technologies: A survey," Journal of King Saud University - Computer and Information Sciences, vol. 30, no. 4, pp. 431 – 448, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319157817300034

[29] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in NSDI'12, 2012.

[30] "Apache Spark Streaming," https://spark.apache.org/streaming/, accessed: 2020-04-23.

[31] A. Spark, "The Apache Spark scalable machine learning library. Available: https://spark.apache.org/mllib/. Last access on May 2018," 2018.

[32] "Apache Spark GraphX," https://spark.apache.org/graphx/, accessed: 2020-04-23.

[33] "Apache Spark SQL," https://spark.apache.org/sql/, accessed: 2020-04-23.

[34] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," ACM Transactions on Internet Technology, vol. 2, no. 2, pp. 115–150, 2002.

[35] "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," https://tools.ietf.org/html/rfc7230, accessed: 2020-04-20.

[36] E. F. Codd, "A relational model of data for large shared data banks," Commun. ACM, vol. 13, no. 6, pp. 377–387, 1970.

[37] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland, "The end of an architectural era (it's time for a complete rewrite)," in Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007, 2007, pp. 1150–1160.

[38] "NoSQL," https://hostingdata.co.uk/nosql-database/, accessed: 2020-04-23.

[39] N. Leavitt, "Will nosql databases live up to their promise?" Computer, vol. 43, no. 2, pp. 12–14, 2010.

[40] "Apache Cassandra," https://cassandra.apache.org/, accessed: 2020-04-23.

[41] "Apache CouchDB," https://couchdb.apache.org/, accessed: 2020-04-23.

[42] "MongoDB," https://www.mongodb.com/, accessed: 2020-04-23.

[43] "Amazon DynamoDB," https://aws.amazon.com/it/dynamodb/, accessed: 2020-04-23.

[44] "Open System Architecture for Condition-Based Maintenance," https://www.mimosa.org/mimosa-osa-cbm/, accessed: 2020-04-23.

[45] S. Goto and K. Tsukamoto, "On-line residual life prediction including outlier elimination for condition based maintenance," International Journal of Innovative Computing, Information and Control, vol. 8, no. 3 B, pp. 2193–2202, 2012.

[46] G. A. Susto, J. Wan, S. Pampuri, M. Zanon, A. B. Johnston, P. G. O'Hara, and S. McLoone, "An adaptive machine learning decision system for flexible predictive maintenance," in 2014 IEEE International Conference on Automation Science and Engineering (CASE), 2014, pp. 806–811.

[47] G. A. Susto and A. Beghi, "Dealing with time-series data in Predictive Maintenance problems," in 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016, pp. 1–4.

[48] A. Kanawaday and A. Sane, "Machine learning for predictive maintenance of industrial machines using IoT sensor data," in 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017, pp. 87–90.

[49] Z. Liu, N. Meyendorf, and N. Mrad, "The role of data fusion in predictive maintenance using digital twin," AIP Conference Proceedings, vol. 1949, no. April 2018, 2018.

[50] F. D. S. Lima, F. L. F. Pereira, L. G. M. Leite, J. P. P. Gomes, and J. C. Machado, "Remaining Useful Life Estimation of Hard Disk Drives based on Deep Neural Networks," in 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–7.

[51] P. Anantharaman, M. Qiao, and D. Jadav, "Large Scale Predictive Analytics for Hard Disk Remaining Useful Life Estimation," in 2018 IEEE International Congress on Big Data (BigData Congress), 2018, pp. 251–254.

[52] B. Kroll, D. Schaffranek, S. Schriegel, and O. Niggemann, "System modeling based on machine learning for anomaly detection and predictive maintenance in industrial plants," in Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), 2014, pp. 1–7.

[53] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine Learning for Predictive Maintenance: A Multiple Classifier Approach," IEEE Transactions on Industrial Informatics, vol. 11, no. 3, pp. 812–820, 2015.

[54] M. Canizo, E. Onieva, A. Conde, S. Charramendieta, and S. Trujillo, "Real-time predictive maintenance for wind turbines using Big Data frameworks," in 2017 IEEE International Conference on Prognostics and Health Management (ICPHM), 2017, pp. 70–77.

[55] J. Xiao, Z. Xiong, S. Wu, Y. Yi, H. Jin, and K. Hu, "Disk Failure Prediction in Data Centers via Online Learning," in Proceedings of the 47th International Conference on Parallel Processing, ser. ICPP 2018. New York, NY, USA: Association for Computing Machinery, 2018.

[56] D. Apiletti, C. Barberis, T. Cerquitelli, A. Macii, E. Macii, M. Poncino, and F. Ventura, "istep, an integrated self-tuning engine for predictive maintenance in industry 4.0," in IEEE ISPA/IUCC/BDCloud/SocialCom/SustainCom 2018, Melbourne, Australia, December 11-13, 2018, 2018, pp. 924–931.

[57] N. Amruthnath and T. Gupta, "A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance," in 2018 5th International Conference on Industrial Engineering and Applications (ICIEA), apr 2018, pp. 355–361.

[58] M. Fernandes, A. Canito, V. Bolón-Canedo, L. Conceição, I. Praça, and G. Marreiros, "Data analysis and feature selection for predictive maintenance: A case-study in the metallurgic industry," International Journal of Information Management, vol. 46, pp. 252–262, 2019.

[59] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber, and G. Fricout, "Steel defect classification with Max-Pooling Convolutional Neural Networks," in The 2012 International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1–6.

[60] D. Weimer, B. Scholz-Reiter, and M. Shpitalni, "Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection," CIRP Annals, vol. 65, no. 1, pp. 417–420, jan 2016.

[61] B. Staar, M. Lütjen, and M. Freitag, "Anomaly detection with convolutional neural networks for industrial surface inspection," Procedia CIRP, vol. 79, pp. 484–489, jan 2019.

[62] P. Strauß, M. Schmitz, R. Wöstmann, and J. Deuse, "Enabling of Predictive Maintenance in the Brownfield through Low-Cost Sensors, an IIoT-Architecture and Machine Learning," in 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 1474–1483.

[63] S. Proto, F. Ventura, D. Apiletti, T. Cerquitelli, E. Baralis, E. Macii, and A. Macii, "Premises, a scalable data-driven service to predict alarms in slowly-degrading multi-cycle industrial processes," in 2019 IEEE International Congress on Big Data, BigData Congress 2019, Milan, Italy, July 8-13, 2019, 2019, pp. 139–143.

[64] W. Yang, Q. Yao, K. Ye, and C.-Z. Xu, "Empirical Mode Decomposition and Temporal Convolutional Networks for Remaining Useful Life Estimation," International Journal of Parallel Programming, vol. 48, no. 1, pp. 61–79, 2020.

[65] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," IEEE Transactions on Nuclear Science, vol. 44, no. 3, pp. 1464–1468, 1997.

[66] F. Giobergia, E. Baralis, M. Camuglia, T. Cerquitelli, M. Mellia, A. Neri, D. Tricarico, and A. Tuninetti, "Mining Sensor Data for Predictive Maintenance in the Automotive Industry," in 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), oct 2018, pp. 351–360.

[67] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, Second Edition, ser. Springer Series in Statistics. Springer New York Inc., 2009.

[68] K. Aggarwal, O. Atan, A. K. Farahat, C. Zhang, K. Ristovski, and C. Gupta, "Two Birds with One Network: Unifying Failure Event Prediction and Time-to-failure Modeling," CoRR, vol. abs/1812.0, 2018. [Online]. Available: http://arxiv.org/abs/1812.07142

[69] N. Amruthnath and T. Gupta, "Fault class prediction in unsupervised learning using model-based clustering approach," in 2018 International Conference on Information and Computer Technologies (ICICT), 2018, pp. 5–12.

[70] P. Perera and V. M. Patel, "Learning Deep Features for One-Class Classification," CoRR, vol. abs/1801.0, 2018. [Online]. Available: http://arxiv.org/abs/1801.05365

[71] R. Chalapathy, A. K. Menon, and S. Chawla, "Robust, Deep and Inductive Anomaly Detection," CoRR, vol. abs/1704.0, 2017. [Online]. Available: http://arxiv.org/abs/1704.06743

[72] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016.

[73] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li, "Edge-Oriented Computing Paradigms: A Survey on Architecture Design and System Management," ACM Comput. Surv., vol. 51, no. 2, pp. 39:1—-39:34, apr 2018.
[74] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," Proceedings of the IEEE, vol. 107, no. 8, pp. 1655–1674, 2019.
[75] A. Thomas, Y. Guo, Y. Kim, B. Aksanli, A. Kumar, and T. S. Rosing, "Hierarchical and Distributed Machine Learning Inference Beyond the Edge," in 16th IEEE International Conference on Networking, Sensing and Control (ICNSC 2019), 2019, pp. 1004–1009.
[76] D. Jahier Pagliari, R. Chiaro, Y. Chen, E. Macii, and M. Poncino, "Optimal Input-Dependent Edge-Cloud Partitioning for RNN Inference," in 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2019, pp. 442–445.
[77] T. Cerquitelli, S. Proto, F. Ventura, D. Apiletti, and E. Baralis, "Towards a real-time unsupervised estimation of predictive model degradation," in Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics, BIRTE 2019, Los Angeles, CA, USA, August 26, 2019, 2019, pp. 5:1–5:6.
[78] "Tensorflow," https://www.tensorflow.org/, accessed 2020-04-23.
[79] "Pytorch," https://pytorch.org/, accessed 2020-04-23.
[80] "Tensorflow on spark," https://github.com/yahoo/TensorFlowOnSpark, accessed 2020-04-23.
[81] "Node-RED," https://nodered.org/, accessed: 2020-04-20.
[82] C. C. Aggarwal, "Towards effective and interpretable data mining by visual interaction," SIGKDD Explorations, vol. 3, no. 2, pp. 11–22, 2002.
[83] D. A. Keim and H. Kriegel, "Visualization techniques for mining large databases: A comparison," IEEE Trans. Knowl. Data Eng., vol. 8, no. 6, pp. 923–938, 1996.
[84] S. Mazumdar, A. Varga, V. Lanfranchi, D. Petrelli, and F. Ciravegna, "A knowledge dashboard for manufacturing industries," in Extended Semantic Web Conference. Springer, 2011, pp. 112–124.
[85] C. Gröger and C. Stach, "The mobile manufacturing dashboard," in 2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS). IEEE, 2014, pp. 138–140.
[86] P. J. Sackett and D. K. Williams, "Data visualization in manufacturing decision making," Journal of Advanced Manufacturing Systems, vol. 2, no. 02, pp. 163–185, 2003.
[87] "2020 Gartner Magic Quadrant," https://www.securonix.com/resources/2020-gartner-magic-quadrant-for-siem/?utm_source=google_ads&utm_medium=cpc&utm_campaign=gartner_mq_2020&gclid=CjwKCAjw-YT1BRAFEiwAd2WRtgkZn1SwmLgHM_PceZT7v5pSsyDihYIeQzbvN11aQ_yZ-3Vn_zXR7xoC4HIQAvD_BwE, accessed: 2020-04-23.
[88] P. J. Sackett, M. Al-Gaylani, A. Tiwari, and D. Williams, "A review of data visualization: opportunities in manufacturing sequence management," International Journal of Computer Integrated Manufacturing, vol. 19, no. 7, pp. 689–704, 2006.
[89] F. Zhou, X. Lin, C. Liu, Y. Zhao, P. Xu, L. Ren, T. Xue, and L. Ren, "A survey of visualization for smart manufacturing," Journal of Visualization, vol. 22, no. 2, pp. 419–435, 2019.
[90] S. Thalmann, J. Mangler, T. Schreck, C. Huemer, M. Streit, F. Pauker, G. Weichhart, S. Schulte, C. Kittl, C. Pollak et al., "Data analytics for industrial process improvement a vision paper," in 2018 IEEE 20th Conference on Business Informatics (CBI), vol. 2. IEEE, 2018, pp. 92–96.
[91] T. K. Kumar, "Multicollinearity in regression analysis," The Review of Economics and Statistics, vol. 57, no. 3, pp. 365–366, 08 1975.
[92] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, no. 1, pp. 100–108, 1979.
[93] P.-N. Tan, M. Steinbach, and V. Kumar, Introduction to Data Mining, (First Edition). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.
[94] B. G. Lindsay, "Mixture models: Theory, geometry and applications," NSF-CBMS Regional Conference Series in Probability and Statistics, vol. 5, pp. i–163, 1995. [Online]. Available: http://www.jstor.org/stable/4153184
[95] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," Journal of Computational and Applied Mathematics, vol. 20, pp. 53 – 65, 1987.
[96] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," Annals of Statistics, vol. 29, pp. 1189–1232, 2000.
[97] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, 2001.
[98] F. Ventura, S. Proto, D. Apiletti, T. Cerquitelli, S. Panicucci, E. Baralis, E. Macii, and A. Macii, "A new unsupervised predictive-model self-assessment approach that scales," in 2019 IEEE International Congress on Big Data (BigData Congress). IEEE, 2019, pp. 144–148.
[99] S. Kim and H. Kim, "A new metric of absolute percentage error for intermittent demand forecasts," International Journal of Forecasting, vol. 32, no. 3, pp. 669–679, 2016.
[100] S. Panicucci, N. Nikolakis, T. Cerquitelli, F. Ventura, S. Proto, E. Macii, S. Makris, D. Bowden, P. Becker, N. O'Mahony, L. Morabito, C. Napione, A. Marguglio, G. Coppo, and S. Andolina, "A cloud-to-edge approach to support predictive analytics in robotics industry," Electronics, vol. 9, no. 3, pp. 492–513, 2020.
[101] "JSON-LD," https://json-ld.org/, accessed: 2020-04-15.
[102] "Angular," https://angular.io/, accessed: 2020-04-23.
[103] "Apache yarn," https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html, accessed: 2020-04-23.
[104] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," ACM Trans. Intell. Syst. Technol., vol. 10, no. 2, pp. 12:1–12:19, 2019. [Online]. Available: https://doi.org/10.1145/3298981

Tania Cerquitelli has been an associate professor at the Department of Control and Computer Engineering of the Politecnico di Torino, Italy, since March 2018. Her research activities have been mainly devoted to fostering and sharing research and innovation on automated data science, machine learning in real-life settings and explainable artificial intelligence solutions. Tania has been involved in many European and Italian research projects addressing research issues on machine learning for Industry 4.0. She got the master degree with honours in Computer Engineering (in 2003) and the PhD degree (in 2007) from the Politecnico di Torino, Italy, and the master degree with honours in Computer Science (in 2003) from the Universidad De Las Américas Puebla, Mexico.

Daniele Jahier Pagliari (M'15) received the M.Sc. and Ph.D. degrees in computer engineering from Politecnico di Torino, Italy, in 2014 and 2018, respectively. He is currently an Assistant Professor in the same institution. His research interests are in the computer-aided design and optimization of digital embedded systems, with particular focus on energy-efficiency aspects, and on emerging applications such as machine learning at the edge.

Andrea Calimera (M'11) took the M.Sc. degree in Electronic Engineering and the Ph.D. degree in Computer Engineering both from Politecnico di Torino. He is currently an Associate Professor of Computer Engineering at Politecnico di Torino. His research interests cover the areas of design automation of digital circuits and embedded systems with emphasis on optimization techniques for low-power and reliable ICs, dynamic energy/quality management, logic synthesis, design flows and methodologies for emerging computing paradigms and technologies.

Lorenzo Bottaccioli (M'15) is Assistant Professor at the Energy Center Lab of Politecnico di Torino (Italy). In 2018 received the PhD, cum laude, in Computer Engineering at Politecnico di Torino. His main research interests concern Smart Energy, Smart City and Smart Communities with focus on software solutions i) for planning, analysing and optimising smart energy system and ii) for spatial representation of energy information. He is a Member of the IEEE.

Edoardo Patti (M'16) is Assistant Professor with tenure track at Politecnico di Torino. He received both M.Sc. and Ph.D. degrees in Computer Engineering at Politecnico di Torino in 2010 and 2014, respectively. His research interests concern: ubiquitous computing and Internet of Things; smart systems, cities and mobility; software architectures with particular emphasis on infrastructure for Ambient Intelligence; software solutions for simulating and optimising energy systems and for energy data visualisation to increase user awareness.

Andrea Acquaviva is Full Professor at Dept. of Electric, Electronic and Information Engineering (DEI), University of Bologna. He got a PhD in electrical engineering from Bologna University and Physics of Complex Systems from University of Turin, Italy. He has been research intern at HP Labs in Palo Alto, CA, USA and visiting researcher at Laboratoire de Systemes Integrés at EPFL, CH. Research interests focus on: Programming models, compilers and runtime for IoT/CPS/HPC platforms, digital twins; Neuromorphic embedded computing.

Massimo Poncino (SM'12-F'18) is a Full Professor of Computer Engineering with the Politecnico di Torino, Torino, Italy. His current research interests include various aspects of design automation of digital systems, with emphasis on the modeling and optimization of energy-efficient systems. He received a PhD in computer engineering and a Dr.Eng. in electrical engineering from Politecnico di Torino. He is a Fellow of the IEEE.