Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2019

Ana Rita Antunes Santiago

Mecanismos para Manutenção Preditiva em Equipamentos de Climatização

Predictive Maintenance Mechanisms for Heating Equipment

Ana Rita Antunes Santiago

Mecanismos para Manutenção Preditiva em Equipamentos de Climatização

Predictive Maintenance Mechanisms for Heating Equipment

"*Without big data analytics, companies are blind and deaf, wandering out onto the web like deer on a freeway.*"

— Geoffrey Moore

Universidade de Aveiro
2019

Departamento de Eletrónica,
Telecomunicações e Informática

Ana Rita Antunes Santiago

Mecanismos para Manutenção Preditiva em Equipamentos de Climatização

Predictive Maintenance Mechanisms for Heating Equipment

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor João Paulo Barraca, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Diogo Gomes, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor Luís de Seabra Lopes
Professor Associado da Universidade de Aveiro

vogais / examiners committee

Doutor David Emmanuel Campos
Team Leader, Bosch Termotecnologia, Sa

Prof. Doutor João Paulo Barraca
Professor Auxiliar da Universidade de Aveiro

agradecimentos /
acknowledgements

Agradeço aos meus pais e irmão todo o apoio e dedicação ao longo deste trabalho, e por me incentivarem a procurar ser mais e melhor.
Um agradecimento especial aos professores João Paulo Barraca e Diogo Gomes, por toda a orientação e disponibilidade que permitiram a realização e apresentação deste trabalho.Por fim, e não menos importante, um agradecimento ao Doutor Mário Antunes por toda a ajuda e cooperação durante a realização desta dissertação.
O presente estudo foi também realizado no âmbito do Projeto Smart Green Homes [POCI-01-0247-FEDER-007678], desenvolvido em co-promoção entre a Bosch Termotecnologia S.A. e a Universidade de Aveiro. É financiado pelo Portugal 2020, no âmbito do Programa Operacional Competitividade e Internacionalização, e pelo Fundo Europeu de Desenvolvimento Regional.

Palavras Chave

Aplicações de Big Data, Machine Learning, HVAC, Manutenção Preditiva, Processamento de Dados, Análise de Dados

Resumo

Equipamentos de Climatização, como caldeiras e ar-condicionado, são suscetíveis a falhas que podem resultar na interrupção de operações importantes. Assim, é relevante aumentar a eficiência dessas soluções e diminuir o número de falhas detectadas. Além disso, entender o porquê da ocorrências dessas falhas torna-se importante para a criação de equipamentos futuros. Existe, assim, a necessidade de desenvolver métodos que permitam a identificação de eventuais falhas antes que elas ocorram. Isso só é possível quando são criadas soluções capazes de analisar dados, interpretá-los e obter conhecimento a partir deles. Esta dissertação apresenta uma infraestrutura que suporta a inspeção de detecção de falhas em caldeiras, viabilizando a previsão de falhas e erros. Uma parte importante do trabalho é a análise de dados e a criação de procedimentos que possam processá-los. O objetivo principal é criar um sistema eficiente capaz de identificar, prever e notificar a ocorrência de eventos de falha.

Keywords

Big Data applications, Machine Learning, HVAC, Predictive Maintenance, Data processing, Data Analysis

Abstract

Heating appliances such as HVAC systems are susceptible to failures that may result in disruption of important operations. With this in mind, it is relevant to increase the efficiency of those solutions and diminish the number of detected faults. Moreover, understand why these failures occur that be relevant for future devices. Thus, there is a need to develop methods that allow the identification of eventual failures before they occur. This is only achievable when solutions capable of analyzing data, interpret it and obtaining knowledge from it, are created. This dissertation presents an infrastructure that supports the inspection of failure detection in boilers, making viable to forecast faults and errors. A major part of the work is data analysis and the creation of procedures that can process it. The main goal is creating an efficient system able to identify, predict and notify the occurrence of failure events.

Contents

List of Figures

List of Tables

Glossary

ANN	Artificial Neural Network
AR	Autoregression
ARMA	Auto-Regressive and Moving Average
ASIC	Application Specific Integrated Circuit
CART	Classification and Regression Trees
CCA	Canonical Correlation Analysis
DBN	Deep Belief Network
DM	Data Mining
DNN	Deep Neural Network
DT	Decision Trees
DV	Data Visualization
FFN	Feedforward Neural Network
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
HVAC	Heating, Ventilating, and Air-Conditioning
IoT	Internet of Things
KDD	Knowledge Discovery in Databases
K-NN	K-Nearest Neighbors
MA	Moving Average
ML	Machine Learning
M2M	Machine to Machine
PAA	Piecewise Aggregate Approximation
PCA	Principal Component Analysis
PdM	Predictive Maintenance
PHM	Prognostics and Health Management
RF	Restoration Factor
RFC	Random Forest Classifier
RNN	Recurrent Neural Network
SAX	Symbolic Aggregate Approximation
SVR	Support Vector Regression
TTF	Time To Next Failure
TTR	Time to Replacement
UD	User Demand
VLSI	Very Large Scale Integration

Chapter 1

Introduction

"If you dont know how to ask the right question, you discover nothing."
W. Edward Deming

The industrial world is being transformed into a technological, automation and data-centric world, called *Industry 4.0*. Companies are investing in data analysis techniques to improve productivity, reduce costs, and increase efficiency [1]. Moreover, one of the biggest advantages of using extensive data analysis is failure prediction, making possible to reduce the resources (and time) dedicated to reactive repairs while reducing the failure events. Since companies are the most interested in reducing those failures, there is a strong need for Predictive Maintenance solutions, especially addressing the prevention of component failure, with impact in production quality and client satisfaction (among many other factors). Therefore, Predictive Maintenance refers to the ability to identify incoming failures before they occur [2], ensuring that important operations are not disrupted as actions can be taken to prevent them. This is a common practice for critical scenario (military, aviation, power plants, etc...), and has since been introduced to a wider range of products.

Furthermore, these systems are usually associated with real-world data that can have issues such as abnormal values. Commonly denoted as outliers, they are different from what is considered as normal in each specific case. In some situations, these values are not considered as noise, but, instead, they can represent malfunctions in the systems. Common examples which represent the importance of using abnormal values are fraud detection in bank accounts or searching for existing cancer cells where the identification of unusual values has an important impact. On Predictive Maintenance systems, anomalous values can represent a failure, responsible for the interruption of relevant operations.

1.1 Motivation

The presented system emerges from the "Smart Green Homes" [1] project, that deals with problems in the realm of Big Data, and also aims to create platforms and services to

[1] http://www.ua.pt/smartgreenhomes/

identify anomalies to increase the useful equipment' lifetime, as well as its efficiency, and the development of future products. In this dissertation, we aim at identifying and predicting failures on a Heating, Ventilation, and Air-Conditioning (HVAC) system, specifically in boilers. For this purpose we analyze real data from thousands of devices, operating at customer premises, over a wide range of product lines and scenarios.

This work evolves from previous contributions [3], which focused on methods to identify outliers in the used datasets, by means of batch-processing and classification algorithms. While there are techniques that can analyze time-series data, being able to identify some outliers, there are still limitations such as time performance, and the fact that it does not allow forecasting of failures, making impossible to predict when and why there will be anomalies in the equipment. Since the main goal of the project is to predict failures in real-time, there was a need to develop a computational infrastructure allowing automatic data processing for a problem that can become a Big Data problem due to the number of devices being analyzed.

1.2 Work summarization

The first step for the system's development was the analysis of the dataset, which contains device operational data. Since we are considering real data, it is more frequent to find normal operation than failure states, which presents both advantages and limitations. The immediate conclusion is that we must operate over an unbalanced dataset as the frequency of outliers is lower than the normal states (standard operation), creating the issue that some classes have a bigger distribution than others which creates a bias in the classification algorithms.

The dataset is also complex, as it has a great variety of alphanumeric variables, from multiple devices running different firmware versions. Therefore, an important part of the presented work deals with data normalization and the development of techniques that can be applied to multivariate time-series data. Consequently, a good implementation of data preparation methods allows the implementation of machine learning algorithms, making possible the identification of failures/outliers.

Second, the data should be processed through an infrastructure that allows effective data processing and a connection to upper blocks responsible for the Machine Learning implementation and visualization. Thus, this work is a vertical extension of an M2M platform, entitled *SCoTv2*, that promotes data acquisition, data processing, and data storage. Connected to the platform, the Data Mining block is responsible for the implementation of predictive techniques. Several algorithms were analyzed on chapter 2, being decided to implement a combination between a Neural Network [4] and a Markov Chain.

1.3 Research and Contributions

The following section lists the key contributions associated with the present work.

- Ana Rita Santiago, Mário Antunes, João Paulo Barraca, Diogo Gomes, and Rui L. Aguiar. 'Predictive Maintenance System for efficiency improvement of heating equipment'. In:

2019 The Fifth IEEE International Conference On Big Data Service And Applications. IEEE, 2019. (Published and Presented)

- Ana Rita Santiago, Mário Antunes, João Paulo Barraca, Diogo Gomes, and Rui L. Aguiar. 'SCoTv2: Large Scale Data Acquisition, Processing, and Visualization platform'. In: *2019 The 7th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2019. (Accepted for publication)

Besides these publications, an important contribution was made regarding the organization, normalization, and data cleaning of the used dataset.

1.4 Document organization

This document is organized as follows: chapter 2 presents the State of the Art of current solutions relevant to this work, chapter 3 describes a possible solution to implement a system able to deal with real-world HVAC devices, and chapter 4 describes the solution implementation and designed architecture. Finally, chapter 5 expresses the obtained results and analysis of the major key features, and chapter 6 exposes the conclusions and future work that is intended to be pursued.

Chapter 2

State of the art

"There's birth, there's death, and in the between there's maintenance."

Tom Robbins

The industrial world is being transformed into a technological one making possible to use data analysis to improve productivity and reduce costs. Time reduction is one of the five needs for data analysis that were defined in 2017 [1]. The other four are increased product quality, decreased cost of the guarantee, improved production and predictive maintenance.

For companies, it is important to reduce the money that is used for repair since that money could be used for developing new products. This is the principal reason for implementing Predictive Maintenance - prevent component failures to minimize reparations [2].

Moreover, Predictive Maintenance emerges as one of the five categories of Maintenance. In 1998, *Eisenmann and Eisenmann* [2] had proposed three categories such as Corrective Maintenance (repair after failure), Preventive Maintenance (sequence of preventive actions without considering the internal state of the product like periodic inspections), and Predictive Maintenance (attempting to find failures before they occur). Two more were defined in 2006 [5]: Scheduled Maintenance, that is performing maintenance regularly, and Condition-based Maintenance that is based on the current application information. Condition-based Maintenance, often called Diagnosis and Prognosis, is similar to Predictive Maintenance. Diagnosis is the way to understand why the equipment is degrading and Prognosis is the time estimated until the failure, usually called Remaining Useful Life. In the industrial context, it is important to implement Diagnosis to be able to fix the problem so the next equipment will not have the same issue, and the Prognosis to create solutions that can be performed when the equipment stops working.

Furthermore, to create a solution for a Maintenance System, it is necessary to determine which methodologies will be implemented, called Prediction Methodologies [6]. There are four types: Model-Based, Analytical-Based, Knowledge-Based, and Hybrid. Model-Based is implemented by creating a model that is simulated considering the configuration and the use of the application, making possible to compare it with the existing historical data. The

Analytical-based methodology is built on the physical analysis of the component from the user experience, considering his understanding of the application geometry. Besides, Knowledge-Based is comparing the system performance with the existing historical knowledge, and the Hybrid methodology creates a collection of techniques with parametric and non-parametric data, such as methods used to analyze data that are not its property.

To define a Prediction Methodology is necessary to implement a Prediction Technique [6]. There are five Prediction Techniques such as Statistics, Experience, Computational Intelligence, Physics-of-Failure, and Fusion; the first is based on the creation of relations between old and current data, and the second on the constant observation of the application usage. Computational Intelligence is based on Neural Networks methods, that take advantage of the computational capacities to create models that project output based on several inputs. Finally, Physics-of-Failure uses parametric data such as methods that rely on data with a normal distribution, and Fusion is about gathering data to create a refined state.

Finally, when a methodology and technique are chosen, it's possible to define a Maintenance System. As it is expected this type of system has advantages that are used to fix the failure problem such as Fault Detection, Fault Isolation, and Fault Identification [2]. Those advantages are important to detect the failure and identify the component where it occurred and what caused it, so it helps the company to improve its products and fix their issues without losing money.

2.1 Predictive Maintenance System

To implement a Predictive Maintenance (PdM) system it is necessary to take into account key aspects, such as the problem under study, determine the indicators that need to be considered, and the techniques that should be defined to measure them [2]. Besides those, forecasting of indicators is also important to build a PdM system, since they can be predicted and analyzed, enabling institutions to make decisions according to the upcoming problems [7]. Like any system, a PdM system requires the identification of the input variables, a target application, and the model that creates a relationship between them [7]. After the model development, it is possible to obtain results to make a decision that can help on the evaluation of the overall system which is very important for the system reliability analysis.

After the system confirmation, it is important to evaluate the amount of available data. Unfortunately, there are boundaries, such as the existence of large amounts of data requiring reduction and normalization. A PdM system can also face a common problem, that is the lack of information about the target application, which will affect the relationship between the application and the inputs. To face those boundaries, it is required to build a practical and comprehensible system that offer technologies with large storage capacity and speed of interaction [8].

Regarding the construction of an effective PdM system, some key blocks are required. Data Source, such as sensors, where data is created, is the input block, also, Data Acquisition is the responsible block to connect sources with the remaining architecture by retrieving the data which will be processed next. Moreover, since data should be persisted it comes in useful

the implementation of Data Storage that stores all the information, and Data Processing that controls the transmission between storage and upper blocks [9]. Furthermore, it is also required the implementation of a Data Mining block that should identify important features on data and discover patterns based on it. This block is one of the upper blocks, which interacts with the application user, is the Data Presentation block that shows important information about the application and the obtained predictions [10].

Then, when the PdM system is mature, it is possible to integrate it with Quality Control systems, using the PdM result for future control and Quality Control as an extra maintenance indicator [11]. This interaction is depicted on Figure 2.1 where the blocks of a PdM system are described and, also, the connection between all system and a Quality Control one. Moreover, this interaction can be helpful for the construction of systems that require notifications for detected faults and interaction with the application users.

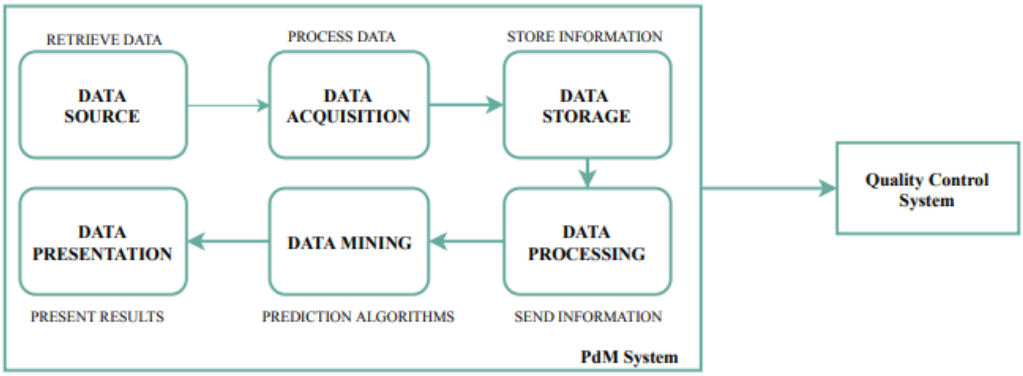

Figure 2.1: Interaction between PdM System and Quality Control

2.1.1 Model-Based System

When implementing a Predictive Maintenance System with a Model-Based methodology it is possible to interpret it in two perspectives: System and Component perspectives [8], [12]. System-perspective refers to the system analysis as one, considering the overall performance. in contrast, Component-perspective is developed analyzing each part of the system and its individual performance, making possible to check if a specific component has a higher impact.

As this type of system is built considering a model, it is important to consider factors related to the application and its use. Model-Based Systems include parameters such as Restoration Factor (RF) and User Demand (UD). The first parameter is the average percentage of system recovery, and the second one is fault identification by the user; *Cher Mig Tan and N. Raghavan* [8] have demonstrated that RF has an influence on the system lifespan since a higher RF leads to longer maintenance and cost savings. As far as the user's opinion is concerned, this may be a drawback to the system availability since when the user's minimum objectives are not achieved his initial expectation cannot be restored. It was also proved that both parameters have an influence on Time To Next Failure (TTF) and Time to Replacement (TTR) since they can influence maintenance.

Model-Based Systems are normally coupled with Machine Learning (ML) since their biggest advantage is the ability to continuously control a process to find the best solution without blocking normal operations. This integration enables error detection by using the Fault Detection mechanism that identifies relationships between inputs and outputs [13], thus, the model performance depends on the quality and quantity of the data since ML algorithms rely on it.

Besides Machine Learning, there is a fundamental part of the system that can detect abnormal conditions, diagnose the failure and its cause, and prognosis about his progression in the future, called Prognostics and Health Management. Since the major goal of Prognostics and Health Management (PHM) is the estimation of the Remaining Useful Life it was implemented on several PdM systems such as [14]–[16], making possible to reduce the failure rate. The exchange of data between processing phases is fundamental and how the data is manipulated is one of the conditions for its performance. The use of a PHM improves the system result and reduces costs in repairs, increasing the efficiency of the company.

2.1.2 Error Detection on PdM Systems

Like any system, PdM Systems are susceptible to the occurrence of mistakes and wrong predictions. There are two categories of errors: Unnecessary Maintenance (False Positive) and Un-Prevented Out-of-control State (False Negative). The first error implies a wrong forecast that generates an unnecessary interaction and the second one is identified when a fault occurred and isn't detected by the model [2]. Solutions have been tested to find reasons for the occurrence of these errors, allowing developers to know that large temporal windows can lead to False Positives [17], however, designate small windows can emerge as the loss of important values and generate False Negatives.

Besides the detection of wrong predictions, it is necessary to report a system failure to prevent the user from thinking that the system is working perfectly when it isn't. It is common to use Console Logs that report all types of error on Predictive Maintenance (PdM) systems since they are heterogeneous and can be applied to several formats and applications [18].

As well as knowing if the system is working properly, it is essential to know the timing of an alert to properly evaluate the quality of the system. There are three timing alerts: Predictive Interval, Infected Interval, and Responsive Duration [19]; the first is the pre-defined time before a failure, if an alert occurs in this interval it is possible to perform a successful maintenance; the second is the time after a failure, where the equipment is being repaired; finally, Responsive Duration reflects the real duration of a necessary repair after a failure, important for the maintenance estimation. When a Predictive Maintenance System has these features it is possible to verify its quality and capacity to report problems, allowing the user to know what to rely on during the system usage.

2.2 Data Processing

To obtain the data from the storage and send it to upper blocks, that actually process the data and issue results, it is necessary to guarantee a fast data transmission. This is particularly

relevant when we deal with a large number of devices, as PdM systems. Therefore, the data processing pipeline is one of the most important aspects of PdM systems, since it is one of the aspects that limit its scalability. There are two typical approaches to Data Processing, that are described describe next: Batch Processing and Real-Time (or Stream) Processing.

Batch Processing follows a strategy where data is analyzed as one large block (or several blocks), creating the need to integrate all information into a master file or contextualized segments, that can be further processed. This approach works over large datasets, and scales very well over a virtualized infrastructure, enabling the exploitation of high parallelization. This assumes that blocks can be processed independently and there is little to no dependencies between blocks. Processing can be started at any time, without time constraints, exploiting energy efficiency and complexity aware allocation of computation resources. Therefore, it is an inexpensive system easily to be audited, and easy to be deployed. The most common examples of Batch Processing are credit card transactions and processing of input/output of an operating system since it is possible to obtain more detailed information about the data.

Real-Time Processing, often called Stream Processing, requires the continuous availability of computing and network resources. Data arrives continuously and needs to be processed in real-time, which means fast enough for the specific scenario. Furthermore, since data is continuously being processed, there is little opportunity to restart processes or to backtrack on errors and failures. This implies that real-time processing applications become very hard to manage unless a proper architecture is used to manage the services and guarantee minimal requirements.

2.2.1 Frameworks and Platforms

There are several frameworks for batch and stream processing [20]. Regarding Batch Processing, there are three principal ones: Hadoop MapReduce which provides a framework that is used for easy development of applications that deal with large amounts of data in parallel, splitting the dataset into independent clusters. Also, Hadoop HDFS is used as a shared repository of data [10], allowing the storage of intermediate results so it ensures that data is available despite failures. Finally, Google Dremel combines the Hadoop capability to scale with columnar storage that enables reading fewer data from secondary storage, thus, reducing CPU overhead.

Stream Processing frameworks can be divided into four dominant ones. Apache Flink is a hybrid framework for stateful computations over unbounded and bounded data streams. It provides a high-throughput, low-latency engine and fault-tolerant applications in the event of machine failure. Also, Apache Storm is a stream processing framework that allows low latency and can handle a great quantity of data, and Apache Samza which is a near-realtime, asynchronous computational framework for stream processing. It is used when scenarios require fault tolerance and buffering since it allows workflow with multiple streams sending data. Finally, Azure Stream is a scalable complex event processing engine by Microsoft that enables users to develop and run real-time analytics on multiple streams of data. It is recommended for IoT scenarios such as real-time remote monitoring.

Considering this division, Figure 2.2 represents the distribution between the two processing

approaches. Besides a processing framework, a PdM system requires data source such as sensors. To connect them, an IoT platform can be used since it fills the gap between sensors and the processing frameworks (through remote and heterogeneous networks). These platforms support data retrieval from the devices (directly or through context storage solutions). Moreover, IoT platforms can be divided into several types such as Hobbyists (built for prototypes), Consumer Electronics (home automation), Industrial IoT Solutions (predictive and remote maintenance), and Industry-Driven (specific concepts such as Healthcare or Agriculture).

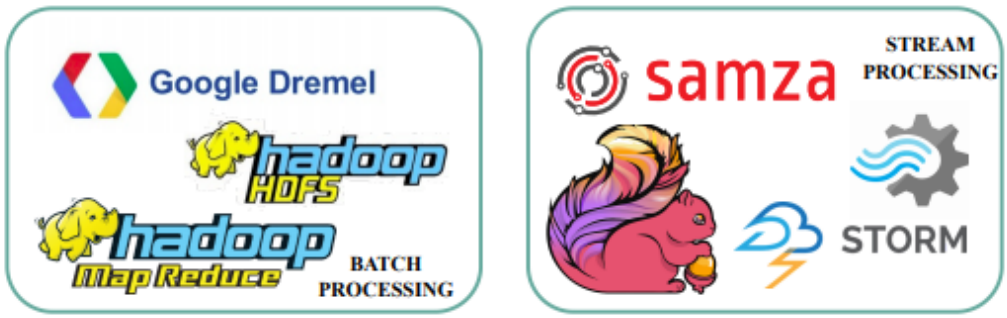

Figure 2.2: Framework distribution between Batch and Stream Processing

Currently, there are several platforms that can be relevant as they fill the gap between the sensors and the processing frameworks. Therefore, it comes in useful to analyze them since they can be implemented as a Processing Block of a PdM System. The most preeminent are Amazon Web Services [13] which provides secure, bi-directional communication between Internet-connected devices such as sensors, embedded micro-controllers, or smart appliances, and the AWS Cloud, ThingWorx[1] that is an end-to-end technology platform that delivers tools and technologies to empower businesses. It allows the immediate development of applications and augmented reality (AR) experiences. And, ThingSpeak[2] is an IoT analytics platform that allows aggregation, visualization, and data analysis in the cloud, and SCoT (Smart Cloud of Things) [21], presented as an M2M platform, that allows multiple users to deploy their services over a rich service execution environment. It is an academic project that focuses on flexibility, scalability, and innovation.

Connected with Google Dremel and Azure Stream there are, respectively, Google Cloud IoT[3] that is an assemblage of tools to connect, process, store, and analyze data both at the edge and in the cloud. Being able to accelerate business agility and decision making, it has machine learning capabilities and an integrated software stack that allows predictive maintenance scenarios. And Microsoft Azure IoT Suite[4] offers both preconfigured solutions and the possibility to create new ones according to the project specifications. The platform allows the connection of hundreds of devices, gathering data analytics and the use of IoT data

[1]https://www.ptc.com/en/resources/iot/product-brief/thingworx-platform
[2]https://thingspeak.com/
[3]https://cloud.google.com/solutions/iot/
[4]https://azure.microsoft.com/en-us/features/iot-accelerators/

for machine learning purposes. Finally, Bosch IoT Suite[5] considers the project requirements that contain various devices and technologies. Moreover, it supports the full app development cycle from its prototype to its deployment and maintenance.

2.3 Data Persistence

Heating, Ventilating, and Air-Conditioning (HVAC) system naturally produces stream data, i.e. measurements of the various aspects of the HVAC recorded on a time or event basis, but always associated with a moment in time. Consequently, a Time Series Database (TSDB) is typically used as a storage solution for such scenarios. A TSDB is a database optimized for time-series data, where "*time series is a collection of temporal data objects*" [22]. Therefore, a TSDB is built for dealing with data that changes over time. Some of the most used TSDB are represented in Table 2.1.

Database	Advantages	Disadvantages
InfluxDB[6]	Fast storage Time-series data retrieval	Database for log
TimescaleDB[7]	SQL scalability Easy interpretation	Large time-based queries
OpenTSDB[8]	Scalability Storage as a server	Hadoop knowledge
Redis[9]	Wide variety of data types Open source	No joins or queries
MongoDB[10]	BSON document as unit Horizontal scalability	Non-indexed queries Hardware requirements
Apache Cassandra[11]	Fault-tolerant Linear scalability	No unanticipated queries
MapR Database[12]	High-performance Operational analytics capabilities	Non-indexed queries

Table 2.1: Time-series databases review

To analyze the effectiveness of the presented solutions regarding Data Persistence, some aspects such as queries were analyzed since PdM Systems must deal with time-series data and be able to perform several queries over it. Therefore, analysis regarding how fast are the queries and if they are possible shall be implemented. Also, PdM Systems shall report maintenance with an efficient time window, so the information shall be efficiently stored and its scalability is an important issue since the number of applications can increase. Finally, the persistence may deal with the necessity of using a specific processing approach and this analysis is also important.

[5] https://www.bosch-iot-suite.com/
[6] https://www.influxdata.com/
[7] https://www.timescale.com/
[8] http://opentsdb.net/
[9] https://redis.io/
[10] https://www.mongodb.com/
[11] http://cassandra.apache.org/
[12] https://mapr.com/docs/home/MapROverview/maprDB-overview.html

2.4 Data Mining

The correct extraction of data features is one of the great factors for the success of a Predictive Maintenance System since it will be used as diagnostic and prognostic parameters [23]. The extraction is part of the Data Mining block which objective is the prediction of relationships between the input data so that the model can infer results in the future. Thus, Data Mining works as a knowledge acquisition tool of the databases [24] and can be used to find hidden information on Internet of Things (IoT) data [25]. Moreover, considering the architecture of a Predictive Maintenance System, the data preparation and interpretation are performed by this block. After obtaining the data, historical and real-time, it is necessary to implement data preparation, transforming data into unified formats, and data cleaning, with the calculation of missing data [23]. Besides that, it is necessary to perform data modeling, which consists of greatly reduce the number of existing features, called Data Reduction, and then extracting the relevant characteristics implementing Feature Selection, as it was done on [26], [27].

In addition, PdM systems are build based on time-series data that includes high dimensionality and is *"recorded at consecutive time periods"* [28], so it is necessary to implement a Data Mining (DM) block able to deal with time-series tasks [29]. Thus, the correct implementation of this block has a huge impact on the system results, being necessary to define approaches to its development.

Consequently, in 2006, *Dengiz et al* presented the two-stage data-mining approach [24] to estimate the distribution of failures. The first stage is Image Processing for automatic identification and recovery of major flaws in microscopic images; and the second stage, Fault Information, is used to be inserted into a distribution function. It was concluded that this technique allows a greater knowledge about the fault and the application where it occurred. Due to a large number of failures and the need to identify them before they occur, the two-stage data-mining approach can be applied to HVAC systems [30], [31] such as [32] where data mining mechanisms were applied to create a system for fault detection, diagnostics, and prognostics.

In addition, the use of data mining was described as necessary for concept description [24] which includes fault diagnostics, scheduling, dispatching, maintenance, and manufacturing processes. This description brings two concepts: characterization used to identify the characteristics that have the most impact on the application quality, and discrimination which allows comparing multiple collections of data, giving their description.

Data Mining can be divided into two types: Descriptive, focused on discovering patterns, and Predictive that intends to predict the behavior of the model and then discover future variables. When implementing this block, regardless of the DM type, it is necessary to know which functions and algorithms can be used. To find the best function within a learning task, it is necessary to know which Machine Learning task solves the problem better. The known functions for data mining algorithms are Clustering, Association, Classification, Summarization, Prediction, and Regression. Within these functions, it is possible to identify multiple learning types: Unsupervised (Clustering), and Supervised Learning (Classification

and Regression) [24].

Clustering is a function implemented in specific data, where the object class is not identified so the output for some input values is not known, creating the need to perform clustering mapping based on the metric similarity of the data [24] to split it into clusters. After, it is certain that objects in different clusters are different, making possible to identify outputs that weren't known before. To implement this function there are several methods that can be defined into five types: Separation (mapping without order), Hierarchical (split by hierarchy), Density-Based (based on the data density), Network-Based (based on the network modeling) [33] and Model-Based. Finally, as the main goal of Clustering is separate data into clusters and identify them, this function is commonly used for service support, production increase, and fault diagnosis [24]. To understand the differences between the five approaches, Figure 2.3 depicts a fruit organization on each Clustering type.

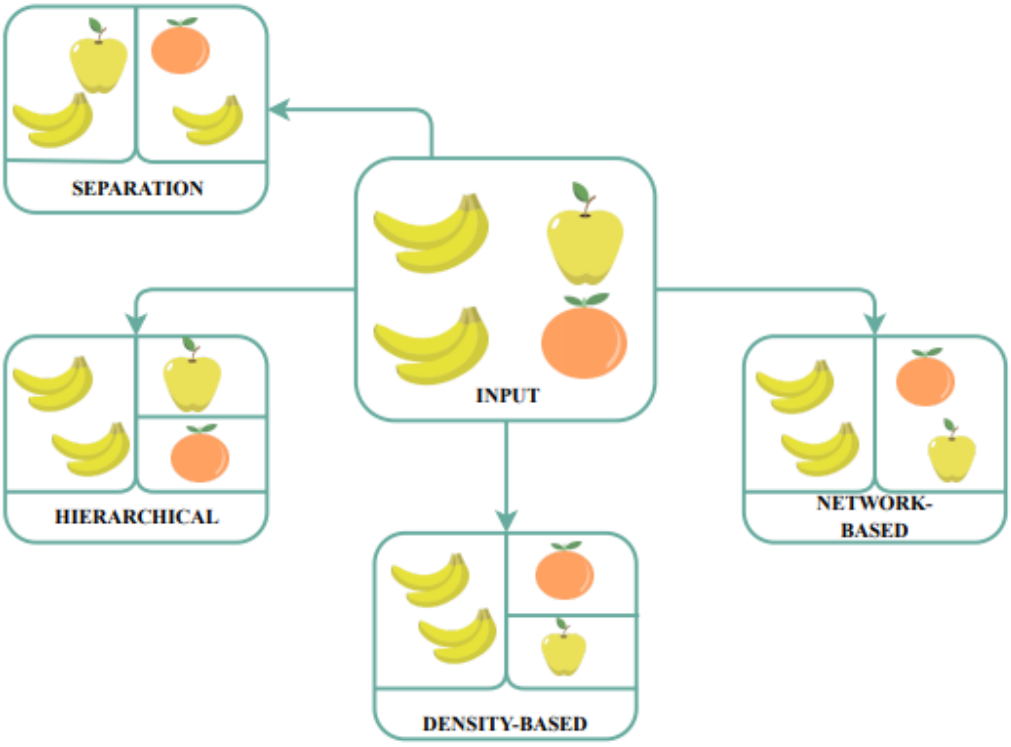

Figure 2.3: Clustering approaches on a fruit organization

Association rules, which appeared in 1993 [24], are used for the identification of relationships between values from a large database. This function is implemented to discover rules between data, being possible to create new rules and associations considering the obtained relationships. This function is commonly used to predict telecommunications failures by identifying events that occurred and establishing associations between them.

Classification comes in useful for manufacturing problems to distinguish between faulty and non-faulty behaviors [34], so, unlike Clustering, Classification works with data that was previously organized into categorized classes. This function is commonly implemented with two steps: first, the creation of the model and, then, the data that was previously split is

classified. The model is usually represented as a set of classification rules, decision trees or mathematical formulas [24], and is constructed with a pair of input/output, creating a set of concepts according to those attributes.

Prediction is about mapping data into prediction values, creating models that assign classes to unclassified data and produce predictions based on those models [24], [35]. This type of learning is mostly used in industrial processes, defect prediction and decision support systems, using methods such as Regression Trees, Neural Networks, and Decision Trees for its implementation [2].

Regression is similar to Classification since they both use data that were previously categorized. However, Regression uses regression tasks, labels composed of one or more continuous variables [36], contrary to Classification that uses labels of a finite number of categories, called classification tasks. So, Regression aims to create a relationship between the variables and find information from the input to discover, what is its impact on the output and how are they related.

As Data Mining aims to implement those functions to predict relationships between the input data and its outputs, it is necessary to have a criterion that evaluates the algorithms results. Therefore, *R.Duda, P.Hart, and D.Stork* [23] proposed three criteria to analyze Data Mining algorithms: Interpretability, how the model helps to understand the data, Predictive Accuracy, how good is the model, and Computational Efficiency that aims to determine how fast and scalable is the algorithm.

As a result, to correctly implement a DM function, a correct learning technique should be implemented. These techniques are one of the ML methods that will be presented in the next section.

2.4.1 Machine Learning

Machine Learning uses the ability of computers to gain knowledge about a dataset [13] and is commonly used in manufacturing applications to find the main factors of the quality decrease [1] since feature engineering is considered one of the most important aspects of Machine Learning. Within Machine Learning there are three sets that will be presented next: training, testing, and validation. The first one is used to train the implemented model, the second one is used to evaluate the performance of the trained model and, finally, the validation set is used as an independent metric for a better evaluation. After splitting into sets it is necessary to decide which learning function will be used. There are several learning techniques: Supervised, Unsupervised, Reinforcement, Deep, and Ensemble Learning that will be explained in the next sections.

Supervised Learning

Supervised Learning is the task of learning a function that obtains an output based on an input through an input-output pair, composed by an input vector and an expected output value. This Machine Learning task analyses the training set and produces a function that maps the relationship between the vector and the output, to use it for future mapping examples.

Supervised Learning can be split into three groups: Classification, Regression, and Combining Models which combines Classification and Regression.

Classification. Classification is very useful for Predictive Maintenance Systems since it can help identify anomalies. Those methods are focused on linear relationships and based on the Autoregression (AR) method, that models the next steps based on a linear function that relates the previous ones. The creation of Auto-Regressive and Moving Average (ARMA) is done by joining Autoregression with Moving Average (MA) which models the next step based on a linear function that correlates the previous errors, therefore, ARMA is thought as a linear function that couples AR and MA, and relates the previous observations and errors. However, since time-series data is not stationary invariance, it can have seasonal errors that should be considered.

Classification models are linear functions that consider not only temporal observations but also normal and seasonal errors, allowing its implementation on time-series data that doesn't follow a trend such as engineering or science applications. Those models make possible the algorithm adaptation when errors occur [37]. Finally, this learning function can be implemented through several methods such as Support Vector Machines, Naive Bayes, K-Nearest Neighbors, and Decision Trees.

Support Vector Machines. Support Vector Machines (SVMs), represented on Figure 2.4, are non-probabilistic classifiers used to establish hyperplanes between training classes. Since they are able to find the largest margin between data, they are useful for IoT problems [36] and Predictive Maintenance Systems [7], [23]. The established hyperplane must be created with the possibility of being used for the largest number of classes since it can change according to the number of input classes [13]. For this reason, when defining the hyperplane, its scalability should be considered. Therefore, SVMs are computationally complex although they are powerful classifiers that can generalize data easily and model almost all kind of datasets [34]. Since SVMs are able to classify all type of data, they are widely used for outliers detection and network anomalies as predictions are based on the location of the hyperplane - if data is distanced from the hyperplane it can be an outlier.

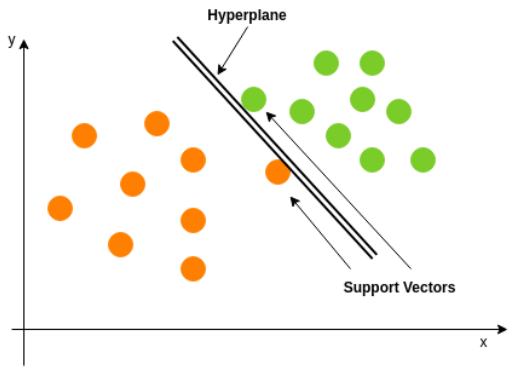

Figure 2.4: Support Vector Machines

Naive Bayes. Naive Bayes is a probabilistic classifier that requires a small dataset to train a model. Its development is based on the concept that a feature value is independent of another feature, and also, that each feature has the same contribution to the class. Naive Bayes requires the establishment of two probabilistic conditions [38] in a sequence of states, in which the next state depends on the previous one, and each observation of the dataset corresponds to one state.

As it can model multi-dimensional data, Naive Bayes is easy to scale and is widely used for creating data filters [36] and validating the data usage in other algorithms and classifiers [1].

K-Nearest Neighbors. K-Nearest Neighbors aims to classify a dataset by considering points from the training set that are closer to an input feature space, being identified as a non-parametric algorithm. When points are identified, they are considered as the K nearest neighbors.

The relation between the feature space and the points which represent different classes is described on Figure 2.5. As can be seen, the closer the circle is to the class points, more similar they are. To calculate the K-Nearest Neighbors (K-NN), a metric distance is calculated such as Euclidean or Hamming distance. Although K-NN is probably the simplest classification algorithm since it only uses computation for calculating distances [34], it requires saving the whole training set, which isn't efficient for large datasets classification [36].

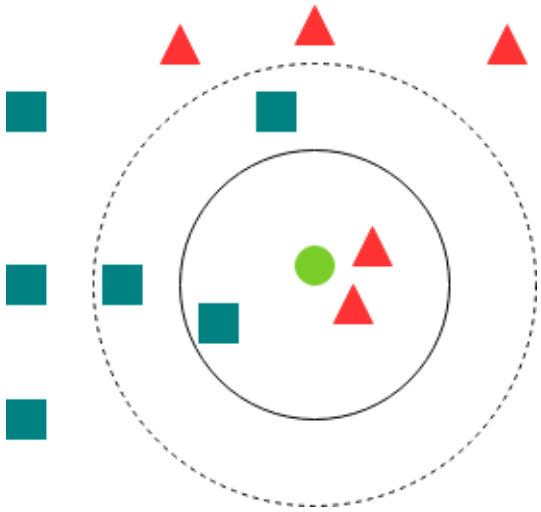

Figure 2.5: K-Nearest Neighbors

Decision Trees. In the IoT context, Decision Trees have been used for fault detection on heating systems [2]. The main idea of this algorithm is to create a binary tree that separates the features into rectangles, and each rectangle is assigned to a pattern [39]. Then, each rectangle is assigned to several leaf nodes, often called ovals, labeled with a different class [22]. This type of classification is often improved with meta-classifiers that combine

several classifiers with low accuracy to create better performance, such as Adaptive Boosting (AdaBoost[13]).

Decision Trees were proven to be effective as a form of feature selection on [23], where researchers used Decision Trees (DT) to extract relevant data from large datasets since this algorithm can easily interpret results and analyze multivariable problems. Figure 2.6 represents a DT used for classify food.

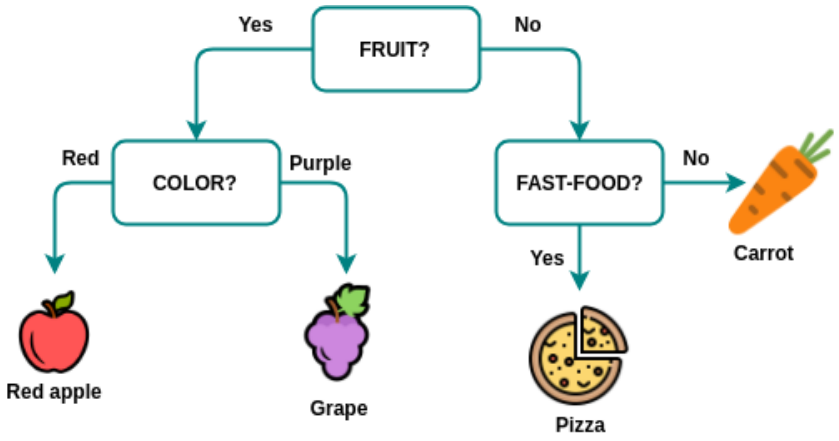

Figure 2.6: Decision Tree for food classification

Regression. Regression algorithms are commonly known as Approximation Functions since they are part of supervised learning and their objective is generating a function that approximates the result of the input value so that future predictions are continuous [13]. This technique can be implemented through two methods: Linear Regression and Support Vector Regression, an extension of SVM explained before.

Linear Regression. Linear Regression generates a relation between a variable y dependent on an independent variable x. Therefore, the objective of Linear Regression is defining a function f that maps the relation between these two variables and can be represented as $f : \varphi(x) \longrightarrow y$. The mapping is done by considering a linear combination of linear and nonlinear functions of the input variable x [36], where the linear combination is represented by $\varphi(x)$. This method allows the relation extension to other methods such as independent variables, called Multiple Linear Regression, or dependent variables, called Multivariate Linear Regression [13]. Since Linear Regression has a higher rate and can use several methods for training such as Bayesian Linear Regression, based on Naive Bayes probability, and Regularized Least Squares, it comes in useful for energy prediction since it is possible to identify the transition between the different energy stages [36].

Support Vector Regression. Support Vector Regression is an extension of the previously shown Support Vector Machines, defining a hyperplane for regression problems. To create a Support Vector Regression (SVR) model, it is only necessary a set of training points

[13]https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/

since the model requires fewer data to produce a result [34], therefore, this model should be complemented with other algorithms. Besides that, Support Vector Regression allows time-series data prediction such as temperature and humidity [40].

Combining Models. Given the two types of supervised learning presented before, models that combine the two techniques can be developed, named Combining Models. These models can be implemented in several ways such as [4], where a study was developed to distinguish parameters with influence on quality control in data from Bosch.

The most known algorithms of Combining Models are Classification and Regression Trees (CART), and Random Forests, both based on binary trees [36].

Classification and Regression Trees. As input, Classification and Regression Trees use an input space divided into sections to assign classification and regression to each of them [36]. This process aims to divide the classification from the regression, thus, the classification methods focus on predicting the class of each section and the regression methods on predicting a constant of the assigned region. The two prediction processes can be unified into a single predictive process that is represented with a binary tree to create a decision-making process.

The training of CART relies on the fact that the tree structure must be based on the training set since this model is implemented by using a top-down construction and splitting the tree node by node. As a result of this structure, it is necessary to define stop criteria to ensure a greater generalization and reduce overfilling.

Classification and Regression Trees have the advantage of being easy to interpret by humans and possible to be scalable for large datasets. However, its performance is totally dependent on a good training set [36], creating the need to guarantee the best training set possible.

Random Forests. Unlike the previous method, Random Forests trains a set of trees in opposite to CART that trains only a tree. To train this algorithm, each tree is trained by a randomly selected section of the training set with a set M of random features [36]. The combination is made by using the first technique to predict the labels of each tree and the second to calculate the mean of the labels found.

As a result, the Random Forests performance depend on the value of the M features, since a small value leads to low prediction results, and a big one to very similar trees. Although it is considered a method with higher performance, it has the disadvantage of not being as perceptible as CART.

Unsupervised Learning

Unsupervised Learning, commonly known as Clustering, is a learning task that learns from data that is not labeled, classified or categorized. Instead of responding to feedback, it identifies similarities in data and produces a result based on their presence on the new data.

Clustering aims to map data into clusters taking into consideration metric similarities. It can be one of two types based on the division: hierarchical and partitioning [22]. The first combines data into subgroups that result into two different approaches: agglomerative (one cluster merges to two or more) and divisive (the opposite); the second type of clustering moves data to the most populated areas. Figure 2.7 describes the Clustering implementation where a fruit organization was made.Besides Clustering, Unsupervised Learning can be implemented through K-means and Feature extraction, which will be presented in the next paragraphs.

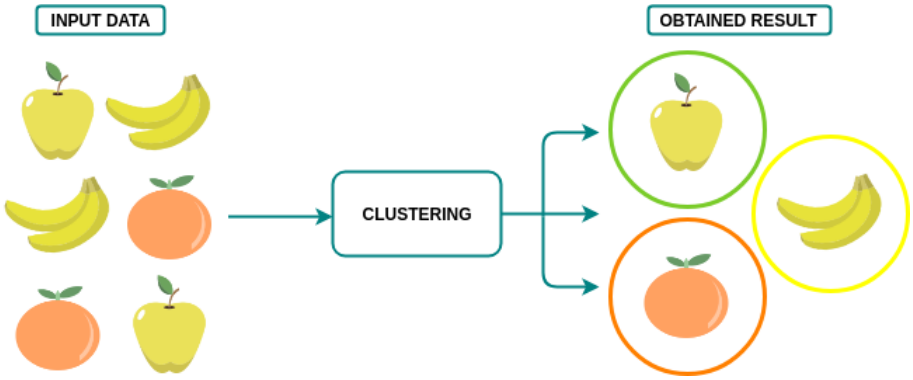

Figure 2.7: Clustering

K-means maps data into K clusters, where points within the same cluster are certain to be similar, being its objective to find the K clusters center to minimize the distance between the points and their center [36]. Consequently, K-means creates a linear complexity on the number of clusters and, in case of a cluster's loss, it is easily recoverable [38]. This method is highly scalable being used for hard clustering problems [25], however, it is not robust against outliers since it can lead to incorrect mappings.

Feature Extraction aims to transfer the input data to a new space, being used as a preprocessing algorithm [36]. This method is a dimensionality reduction process since the initial set of variables is reduced to groups of features (called feature vectors), therefore it is used on datasets that are thought to be redundant. The selected features are expected to contain relevant information, so the task can be performed by using the produced reduction of the original data.

This algorithm is often called Principal Component Analysis (PCA) and it involves data compression, cleaning, and data visualization [36]. PCA is a statistical procedure that uses one transformation to convert a set of observations into a set of values called principal components. The linear reduction of Principal Component Analysis is called Canonical Correlation Analysis (CCA) and it differs from the first one because it uses at least two variables to find a match between two subspaces and then find the correlation between them [36]. The purpose of Feature Extraction algorithms is to separate data into sub-spaces to obtain the most important features of a dataset, being a useful algorithm for anomaly detection [41].

Reinforcement Learning

In addition to the already presented learning tasks, there is still Reinforcement Learning, that tries to resemble the human capability of learning from experiences. *"The goal of reinforcement learning is to minimize a cost-to-go function"* [38] to find the best actions in the system behavior and improve the feedback received by the model. Then, the agent obtains information from the environment to update his knowledge about it, called the perception-action-learning process. Consequently, Reinforcement Learning agents face challenges such as the fact that the agent only learns through the received feedback and his observations are dependent on past actions, creating temporal dependencies between the decisions that were taken [42].

Reinforcement Learning problems can be solved through two methods: Value Functions and Policy Search. The first is based on a value estimation at a given moment, while the Policy Search methods look for the best strategy from the beginning.

Deep Learning

Deep Learning algorithms have been increasingly used for IoT applications such as Smart Homes. This learning type is more efficient on architectures with a large number of features [43] and are based on distributed representations [40]. Therefore, Deep Learning comes in useful for classification problems (associated with SVM), size reduction and forecasting, being used for both supervised and unsupervised problems [44]. Moreover, Deep Learning reduces the need for large-scale predictive analysis since feature reduction is done during training [45], leading to better accuracy.

For example, on Smart Homes problems, Deep Learning works with data from different sources creating a diverse dataset, generally composed by time-series data that need to be modeled with an ARMA predictor [45] or integrated into time intervals as it was done on [46] for modeling home's temperatures.

Deep Learning architectures are structured in several processing layers, where each layer is responsible for producing answers to the proposed problems and the weights of each layer are randomly assigned to the data in the training phase [43].

Ensemble Learning

Ensemble Learning is the process of combining multiple learning algorithms to obtain a better prediction. This method has advantages based on three reasons: statistical, computational, and representational; the first one is related to the lack of data to represent data distribution since ensemble methods reduce the risk of selecting the wrong model. The computational reason relates the fact that ensemble methods run based on a local search with several starting points increasing the performance of the search, and the last advantage comes from the fact that ensemble methods can approximate functions with the sum of several hypotheses, creating better representation for the final function.

For time-series problems and forecasting, the use of these methods can solve the problem of nonlinearity, very common in problems of prediction as in [47], where an ensemble model was used for load prediction on an HVAC system.

The main difference between ensemble learning and other machine learning techniques is the fact that ensemble methods require less tuning and knowledge about data [39]. Thus, their success comes from their diversity, making use of different algorithms to improve performance.

2.4.2 Neural Networks

Artificial Neural Networks, commonly known as Neural Networks, are computational models based on the human brain. This type of algorithm, such as the brain, is composed of interconnected neurons creating a set of connections through which the information is processed [4].

They are being increasingly applied to areas such as finance, medicine, and water systems, since they are quite efficient in prediction issues [48], and easy to adapt to classification and regression problems [36]. Moreover, the strengths of these networks are that they have a parallel construction, being able to generalize problems [13], and use models for multiple cases [48]. The parallel construction also allows the identification of distinct patterns with minimum preprocessing [49]. Considering its use, *S. O. Haykin* defined a group of relevant characteristics of Neural Networks [38]:

1. Non-linearity: Neurons can be linear or nonlinear allowing the resolution of all types of problems;
2. Input-Output Mapping: Neural Networks are able to map the inputs of the training values to their output values by modifying their weights and controlling the network until the difference between the desired response and the current response is stable;
3. Adaptivity: Neural Networks are trained for a given application but are easily reconstructed for a different one;
4. Evident response: A degree of confidence can be obtained for classification problems;
5. Contextual information: Neural Networks deal with contextual information, where neurons are influenced by each other;
6. Fault Tolerance: Neural Networks implemented in hardware are computationally robust since damaged connections only affect the network performance;
7. Very Large Scale Integration (VLSI) Implementation: Neural Networks are easy to implement in hardware;
8. Standardized Analysis and Design: The same notations can be used for all application domains;
9. Neurobiological Analogy: Since Neural Networks are based on the human brain, they have great potential because they can learn based on the information processed by the neurons.

Neural Networks are frequently used as ensemble methods being used as a combination of several networks to model uncertainties and to improve forecasting accuracy and robustness. Consequently, these methods have a great capacity for model time-series data, being flexible and capable of creating relations between inputs [50].

Neuron

The processing unit of a neural network is a neuron, composed by input signals, a bias, an activation function, and an output. Based on the next neuron representation, Figure 2.8, it is possible to understand that a relation between the output and the inputs is created. First, each input has a synaptic weight that serves as an argument for the linear combination, performed by the adder [38]. And, finally, the result is an argument for the activation function, known as Squashing Function, that produces the output. Unlike what happens in the human brain, the weight of a neural network link is based on a real range so it may contain both positive and negative values. Thus, to solve this, a bias is incremented to the adder so that the input value can be normalized.

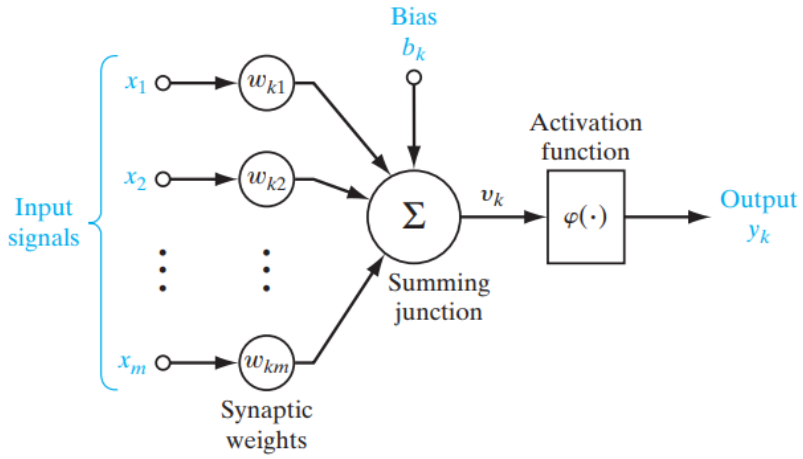

Figure 2.8: Neuron

Architecture and Implementation

A Neural Network is structured into layers and each layer contains one or more neurons. Within layers, there are three types: input layers, working as an input data consumer, hidden layers, invisible to the training set, and output layers that obtain the response value.

To build an architecture, several hidden and output layers can be defined but the number of output layers should always be lower than the input layers. This definition results in two characteristics: depth and weight. The first is the number of defined layers, while weight is related to the layer constitution [4]. The performance increase is directly proportional to the waiting time between input and the response obtained by the algorithm, therefore, the more weight the connections have, the longer the waiting time, and, also, the more complex is the network, the easier it is to solve problems. Thus, *Kumar* [48] proposed that Neural Networks could be optimized through Backpropagation, which optimizes the number of neurons despite *Maier and Dandy* [48] that identified some steps for the correct implementation of a Neural Network. Those steps were data preprocessing, description of the appropriate models and parameters to be used, defining the network size and the number of connections, optimizing the network, and validating the model.

Types of Neural Networks

Considering the connection between layers and the network structure, two types of Artificial Neural Networks can be defined:

1. Feedforward Neural Networks - network containing zero or more hidden layers [38] and where the information flow is acyclic [4], having no dependencies between output and input layers [43];
2. Recurrent Neural Networks - a bidirectional flow that allows feedback to make possible learning from sequences instead of training examples [38].

In addition to these two types, Dynamic Wavelet Neural Networks (DWNN)s can also be defined, and this type of network incorporates temporal information and storage capacity to predict future events [51]. As a result, it allows for unique identification and classification abilities. The major difference from the other two types is that DWNN is able to model the temporal evolution of a dynamic system.

Feedforward Neural Networks. The most common type of an ANN is Feedforward, admitting two subclasses: Single-Layer and Multilayer. The first one consists on a network where the input layer directly sends the data to the output layer [38]; and the second supports at least one hidden layer, making impossible to see the internal flow of information. Multilayer networks have the possibility to extract more than one result and to be analyzed in a global way. Figure 2.9 describes the architecture of a Feedforward Neural Network (FFN), where the three-layer types and the corresponding connections, are represented.

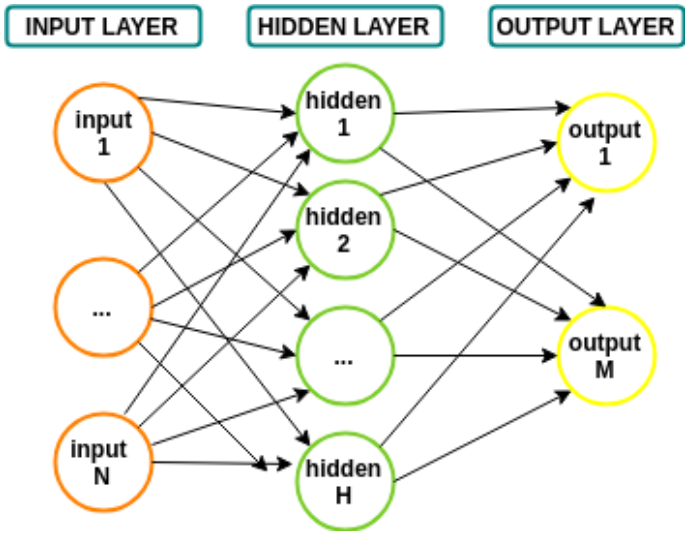

Figure 2.9: Feedforward Neural Network

Feedforward networks can be evaluated according to the connectivity of their neurons: fully connected, when all neurons connected to at least one adjacent neuron, and partially connected when there are synapses missing. This type of Neural Networks can be applied for classification, regression, clustering, and feature extraction problems, being widely used on IoT problems to predict the components states. The known types of Feedforward Networks are

Autoencoders, Deep Neural Networks, and Convolutional Neural Networks that are depicted in the next paragraphs.

Autoencoders are Feedforward Networks used for unsupervised learning. These algorithms have input and output layers connected by the hidden layers and their goal is constructing an output based on the received information [43]. The main difference is the fact that the number of nodes in the input layer is equal to the number of nodes in the output layer [13]. Besides that, an autoencoder is composed of two main components: encoder and decoder. The encoder maps the input data into a new representation to send it to the decoder, and the decoder uses the received data to reconstruct it and get the original input. Thus, these type of networks transforms the input signals into learned features [52] and is generally used for data denoising and dimensionality reduction [13].

When FNNs contain multiple hidden layers, they are called Deep Neural Networks (DNNs) [13] and are commonly used for object recognition and natural language processing [53]. Since the training can be done during runtime [13], these type of networks can be used for supervised problems such as classification and regression, being implemented on IoT applications for fault detection [54]. Figure 2.10 describes the architecture of a DNN, where the layer types and the corresponding connections, are represented.

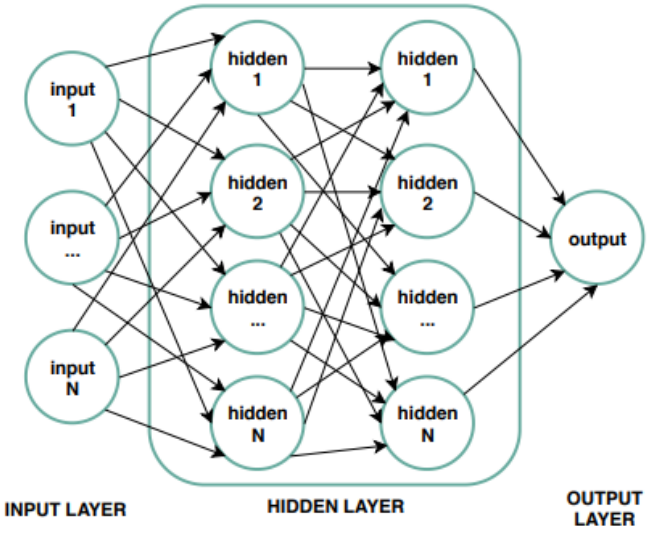

Figure 2.10: Deep Neural Network

To implement supervised learning, Deep Neural Networks are transformed into Deep Belief Networks that are Deep Neural Network (DNN)s composed of multiple hidden layers and associated with a Support Vector Regression algorithm. Deep Belief Network (DBN) can also be used for unsupervised learning, coming in useful for feature extraction [55] on time-series forecasting problems.

Unfortunately, Deep Neural Networks have the disadvantage of having connections between neurons that are very complex, which increases the difficulty of training the model and makes hard to scale the network [43], therefore, they are intensive in terms of computing.

Convolutional Neural Networks are FFNs capable of processing images as inputs [13], being widely used for image processing applications [56] as it is depicted on Figure 2.11.

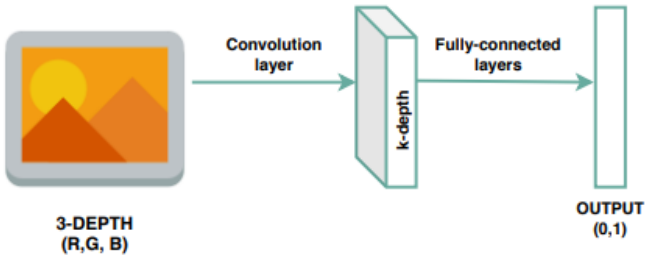

Figure 2.11: Image processing over a CNN

Although the fact that they are composed by a set of small filters, these algorithms guarantee a great depth for the input data, being computationally intense. Unfurtonally, it is necessary to accelerate the data processing through Graphics Processing Units (GPUs), Field Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASICs).

Attempts were made to implement Cloud Computing, but this method entails several problems such as wireless connections associated with high power consumption and loss of signal. Thus, the most common method is the use of GPUs since they are easy to use and have a good performance.

The training of this type of networks is done through filters, that cross the input volume, and calculate the dot product between the input and a function [43], usually the Sigmoid nonlinear function [52]. This calculation is dependent on the layers operation, which is one of three types: fully connected, pooling and convolutional. Fully connected layers are similar to the layers of a Deep Neural Network, the pooling layers reduce the representation size, decreasing the number of parameters and the probability of network overfitting [43], and convolutional layers are responsible for filter control [13]. Finally, each layer supports a 3D kernel for extracting features [56] and consists of convolution and max-pooling operations [52]. Convolutional Neural Networks are useful for identifying simple patterns relevant to find more complex patterns.

Recurrent Networks are distinguished from the previous ones due to the fact that they have at least one feedback cycle. The sending of feedback from a node to itself is called self-feedback, and the natural cycle is done by sending information from a previous node to the next one, creating a dynamic behavior [38]. This structure allows the network to learn based on the results of experiences, similar to the human brain. Based on the architecture of a Recurrent Neural Network (RNN), Figure 2.12 represents its organization and the communication between layers.

Moreover, *"RNNs can be thought of as n copies interconnected DNNs"* [13] since the output generated at time $t-1$ affects the output at time t. This dependence is generated by the fact that each neuron has an associated feedback cycle. Thus, it is necessary that each

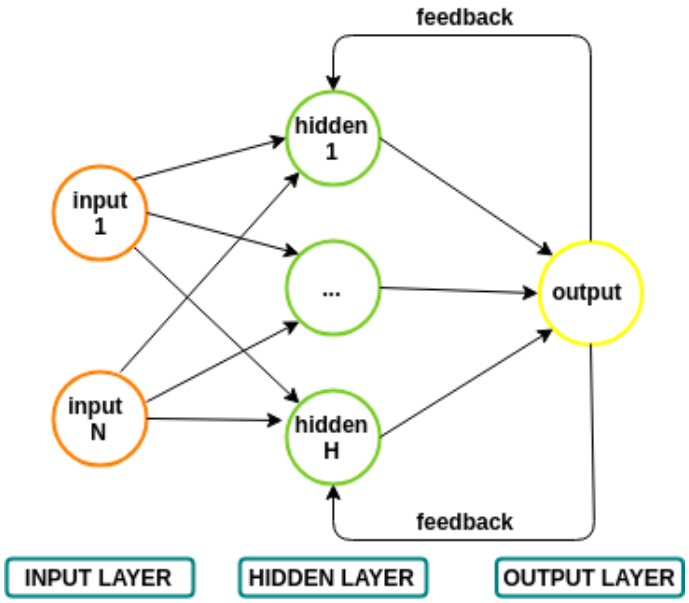

Figure 2.12: Recurrent Neural Network

neuron has internal memory to store information, entitled as Long Short-Term Memory, and a set of ports per unit for access control of memory cells. Considering that, each neuron has three different ports: forget, read and write gates. When forget gate is active the neuron writes on itself, and, when inactive, all content is forgotten; thus, this gate controls the cells state and ensures that they are not degraded [43]; the read and write ports, when enabled, provide to other neurons the ability to read and write to the port's owner.

Consequently, Recurrent Neural Networks can use their memory to process sequences of inputs making them useful for not segmented problems which require machine translation and speech recognition [41].

Reduce Neural Network Overfitting

Given the characteristics of a Neural Network, it is understood that the more data they receive, the better their results. However, larger networks are slower and can lead to overfitting by the number of created relationships. This problem raises the need to create solutions to amortize it. As a result, several methods are being tested such as stopping the training when the performance starts, which decreases the weight penalties [57]. However, these methods weren't enough, so two methods were presented as the more effective: Data Augmentation and Dropout.

Data Augmentation is known for its effectiveness on datasets that contain missing data. This method works by increasing the amount of training data and is generally used on data warping (augment of the input data to the model) [58].

This technique can be applied to any type of data, however, it comes in useful for data where additional information can help on decision-making such as sales patterns or product sales. There are several steps that should be done while implementing Data Augmentation:

- Extrapolation: relevant fields are updated;

- Tagging: common records are tagged to a group, making easier to differentiate groups;
- Aggregation: using mathematical values for relevant fields;
- Probability: based on analytical statistic, values are populated based on the event probability.

Therefore, if the dataset has a lack of data, Data Augmentation can increase its size and help on the robustness of the implemented model and simplify the training set. Figure 2.13 describes the effect of using this technique on a dataset which only contains one image of an object. Using this technique allows the creation of four more images, increasing the amount of training data and the knowledge about the input.

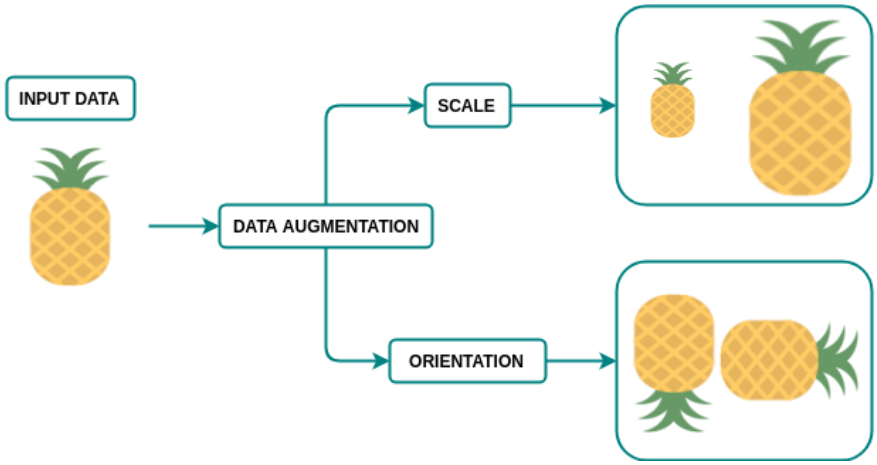

Figure 2.13: Data Augmentation effect

Dropout works by removing randomly neurons from layers during model training [58] and, consequently, preventing units from co-adapting [57]. The units are temporarily removed from the network and it provides a way to combine different neural networks. As a result, the "dropped-out neurons" contribution is temporarily removed and weight updates are not applied.

This technique can be seen as a method for regularization by adding noise to the hidden units, being able to minimize the loss function. Like any solution, it has drawbacks such as the fact that it increases the training time. However, allows a fast approximation [40], which provides higher performance, and a better generalization. Figure 2.14 represents its impact on an Artificial Neural Network (ANN).

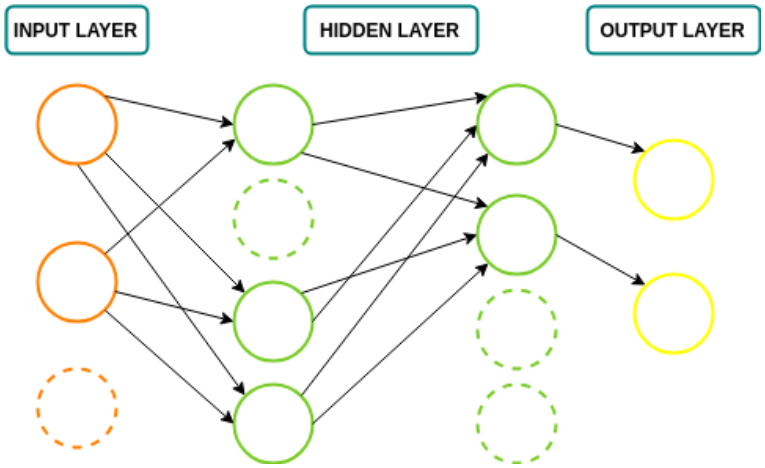

Figure 2.14: Neural Network after Dropout

2.4.3 Knowledge Discovery in Databases

Both data extraction and Data Mining algorithms are important to the good implementation of the previous learning techniques. Also, they are part of the Knowledge Discovery in Databases (KDD) process, involving the application of algorithms for extracting models from the data.

KDD includes theories, algorithms, and methods through the interconnection of several concepts such as databases, machine learning, statistics, artificial intelligence, knowledge-based systems, and data visualization [24]. So, KDD is a non-trivial process that allows the identification of new and important patterns from the data.

The identification of IoT data with KDD is done by using some steps such as selection, preprocessing, transformation, data mining, and interpretation/evaluation. The first three can be grouped into a data processing step and the last one as the decision-making. Moreover, the result of these steps has a strong impact on the mining results [25] and discovered patterns, since the number of discovered patterns results on the creation of a notation, *interestingness*, that combines the newness, usability, and simplicity of a pattern.

Therefore, to implement a process that combines both data processing and decision-making, there are several steps for manufacturing systems:

1. Have knowledge about the system and its purpose;
2. Gathering data, select it and then focus on the variables that are important to the problem;
3. Pre-processing to make the data readable;
4. Integration of the various data sources;
5. Choose the function and the Data Mining algorithm to be applied;
6. Interpret and visualize the obtained patterns;
7. Implement knowledge to obtain feedback;
8. Introduction of results in the company.

Although the previous steps can be effective by implementing them one time, there are solutions that require them to be iterated several times such as [59], where the KDD's steps were repeated to reach their goal.

Consequently, it is understood that Data Mining is an important step on the KDD process since it turns low-level data into useful knowledge.

2.5 Data Visualization

As PdM systems deal with billions of data generated every day, visualization is an important key tool that allows the analysis of massive amounts of information and data-driven decisions. Data Visualization (DV) is the graphical representation of information and data. Moreover, since the major goal of these systems is the failure identification, this block provides an accessible way to see and understand trends, outliers, and patterns in data. Considering some of the most used tools for Data Visualization, several platforms are described in Table 2.2.

Dashboard	Advantages	Disadvantages
Grafana[14]	Several time-series data storage Allow notifications and alerts	Full-text data querying not permitted
Kibana[15]	Data querying and analysis Several data representations	Work only with Elasticsearch
Graphite[16]	Highly scalable Render on demand	Specific database
Prometheus[17]	Multi-dimensional data model Flexible query language	Not viable for anomaly detection
Datadog[18]	Collaborative work Single view across cloud deployments	Critical with data increasing
Looker[19]	Collaboration features Workflows are easily managed	No analytics engine
Zoho Analytics[20]	Insightful reports Top-of-the-line security measures	Website usability of service
Sisense[21]	Intelligent Business decisions Agile analysis software	Navigation on the mobile platform

Table 2.2: Visualization platforms review

2.6 Related Work

As the main objective of Predictive Maintenance is to forecast the occurrence of faults to guarantee the availability of the applications, several approaches were performed to produce a Predictive Maintenance System that allows that. An example is [2] where predictive maintenance was applied to a set of similar appliances using dissimilarity-based representation. Although it uses dissimilarity for feature extraction, requiring that the extraction isn't influenced by external aspects, this application has the problem of analyzing the dataset with a global perspective. Since it allows the loss of singular problems, other techniques

[14] https://grafana.com/
[15] https://www.elastic.co/products/kibana
[16] https://graphiteapp.org/
[17] https://prometheus.io/
[18] https://www.datadoghq.com/
[19] https://looker.com/
[20] https://www.zoho.com/analytics/
[21] https://www.sisense.com/

such as [32] were implemented, in spite of being used with small datasets, this approach focused on singular failures of a HVAC system to find failures, and analyze their impact on the application.

Since PdM systems entail several amounts of data, the previous two implementations have the obstacle of handling small datasets making it difficult to identify if they would have great accuracy with the increase in data. To fight this struggle, PdM systems such as [60] and [10] were implemented. The first approach describes the foundation of an optimized system, combining infrastructure and process mining techniques. These methods are based on two different specifications, first process-related insights to the application layer, and, second, on infrastructure-related information regarding the technology. In the context of digital predictive maintenance services, the main result is an adapted reference model for digital enterprise architectures.

The second system [10] is based on a data-driven solution using Cloud Computing, where the main objective was predicting failures that could be monitored for an agent and then present the results. However, the dataset used wasn't balanced, resulting in the issue that some classes had better distribution than others, making impossible to verify if the obtained accuracy only reflected the distribution of those classes.

Besides that, Cloud Computing is not scalable, not guaranteeing that the growth of data couldn't interfere with the solution accuracy. Therefore, solutions that implement offline techniques can be more effective. Considering the diversity that can be obtained with Neural Networks, several PdM systems have been implemented using this type of algorithm. In [61] an analysis regarding the RUL was made and its estimation became central to the development of systems that monitor the current state of machines. Although the system studied this field in-depth, there is no universal method so RNN was implemented as a possible solution to study RUL information. Another solution based on Deep Learning algorithms is [62] where a solution was developed to monitor sound sequences captured from a microphone, analyze them and return classification results. The sound sequences are subsequently analyzed using NNs and the prototype can analyze sounds produced by a mechanical machine and classify different states, being possible to solve predictive maintenance tasks.

Moreover, an RNN was used in [63] to develop a PdM system able to analyze the oil and gas industry. The recent crude oil price fall reinforced the importance of effective maintenance management across the oil and gas industry. Effective maintenance is crucial to avoid damage and downtime for repair. This approach implemented a RNN to carry out Predictive Maintenance of Air booster compressor motor. The application of these algorithms could mitigate risk and reduce cost in the oil and gas operation.

Regarding fault detection on PdM systems, [64] aim to predict imminent faults by estimating autoregressive integrated moving average models that use real-world sensor data obtained from monitoring different machine components. The system' outputs are fused to identify the significance of an anomaly and determine how likely a fault is to occur, with alarms being issued when the fault imminence is high enough. Moreover, approaches were implemented for dynamic systems such as [65] which provides a method and system that

includes a plurality of computing modules each configured to retrieve maintenance history and create modified maintenance schedules based on the maintenance history.

Another offline approach is [3] where a solution was proposed to face the same issue as the one presented in this document. This solution focused on methods to identify outliers in the used dataset, by means of batch-processing and classification algorithms. First, the multi-type variables were translated into textual data (strings) and then classification algorithms were applied. The algorithm is responsible for representing the time series in a unified way, enabling the implementation of machine learning techniques. The combination of two methods, Piecewise Aggregate Approximation (PAA) and Symbolic Aggregate Approximation (SAX), allowed the conversion of time vectors into a set of letters that are analyzed next. The PAA algorithm divides the time series into a vector of equally sized segments where the number of segments is the same as the number of classified variables. Thus, each time series is represented by a set of multidimensional vectors, all with the same dimension.

However, while there are techniques that can analyze time-series data, being able to identify some outliers, there are still limitations such as time performance, and the fact that it does not allow forecasting of failures, making impossible to predict when and why there will be anomalies in the equipment. Moreover, there is an obligation to define segments with a specific size which requires great knowledge about data and a recurrent division into vectors that can be inefficient.

Thus, since the main goal of the project is to predict in real-time failures, it was required to develop a computational infrastructure allowing automatic data processing for a problem that can become a Big Data one due to the number of devices being analyzed.

Chapter 3

Proposed Solution

"If you define the problem correctly, you almost have the solution."

Steve Jobs

As presented in chapter 1, Big Data services can be used to develop PdM systems. They become more relevant as the number of heating appliances becomes more complex and expensive, and the number of clients increases. Moreover, the appliances are heterogeneous and may suffer from different issues, which can rise different failures. As a result, companies need to improve their maintenance procedure to reduce the costs of repairs and improve competitiveness.

In chapter 2, we presented several approaches to deal with and implement PdM systems. Although there are several possible solutions, they do not combine the acquisition, processing and predictive parts as a single platform. Thus, this work should propose an efficient Big Data Mechanism that can be instantiated for several Predictive Maintenance scenarios and environments.

Traditional PdM technologies require that companies should have a dedicated system in every boiler, which increases the cost of the heating appliance. As an alternative, the data produced could be placed in a centralized system, but this solution requires huge computational resources to be capable to process all the information and notify the occurrence of a failure during an efficient time. The main difficulty is the scalability of the solution. As the volume of data increases, it becomes difficult to process all of it. Increasing the hardware is also an unsatisfactory solution since it increases the costs for the companies.

Considering these issues, a solution that is able to handle them shall be developed. However, before the definition of a possible structure, an evaluation regarding the data shall be implemented. Therefore, this chapter first analyzes the data that is meant to be processed on section 3.1, and, finally, defines the necessary requirements and structure that allow an efficient solution.

3.1 Data

To perform a Machine Learning process it is necessary to understand the system under study and, consequently, have knowledge about the data and its characteristics. After, it is possible to determine the best method to resolve the problem and interpret the obtained results. Commonly, the final output can be translated into the original problem language to create a general understanding of all the results. So, complete knowledge about the data is required to solve the issue.

For this work, data is representative of the actual behavior of boilers, which may not always properly perform their essential function of hot water supply. Since they are running daily, standard operation is more frequent than abnormal events, resulting in an unbalanced dataset. As most machine learning methods are statistically based, an unbalanced dataset commonly results in a classification preference for the most frequent class [34] which can be helpful if the pattern identification is based on the identification of values that do not fill some characteristics. However, an unbalanced dataset also results in bias on the classification algorithms.

We are dealing with data from boilers provided by the Thermotechnology division of Bosch, a world leader in the development and production of heating and hot water systems with high energy efficiency. Fault identification and predictive maintenance allow the improvement of components and the maintenance in advance of already functioning appliances. By identifying the most common faults, it is possible to substitute the components with a higher failure rate. Moreover, a more extensive and accurate fault identification will allow a quicker and effective correction of boiler malfunctions by the warranty maintenance responsible, since each code corresponds to a specific component failure being documented with a possible solution.

The dataset is obtained from the appliances gateways that are able to identify some faults and automatically tagged them with a fault code. This identification results in labeling data with a code that can be associated with some behavior patterns. Therefore, our goal is identifying the unclassified faults, their causes, and, also, predicting the occurrence of all failures (labeled and unlabeled). These boilers, whose main function is to supply hot water to the customers, are installed in their houses. This water can be utilized as domestic water such as bathing or kitchen, denoted by *Hot Water (HW) Cycle*, or for kitchen use and heating of wall-hung devices, called *Central Heating (CH) Cycle*. There is another cycle, *Boost Cycle*, which provides a quicker water supply. The three cycles are considered normal and are the most representative patterns of data, however, they are not the goal of this study. However, one approach to outlier identification can be through the learning of normal operation cycles and then labeling as faults the behaviors that do not fit in such patterns. Thus, it is also important to consider boilers normal operation to identify which behaviors are not considered normal and are, consequently, faults.

Boilers have sensors collecting data such as temperature, number of boiler starts, number of heating requests, duration of each cycle, among others which creates a dataset composed of

continuous and discrete variables. Although there are state variables such as open or closed valves, it is possible to convert them into discrete variables since they only admit two opposite values - on/off, for example. As state variables, can be transformed into discrete values such as binary ones, the states "On" and "Off" can be coded as 1 and 0, respectively.

The appliance's software is able to identify 65 different faults, attributing to each one a distinct code. Each fault code is related to a specific boiler component failure and a possible maintenance solution. Among others, there are, for example, failures in the pump operation not allowing a correct water circulation, failures in the fan or in the gas valve opening or closing. Besides that, data loss due to connection problems or sensor malfunctions is also associated with a fault code. This variable allows an automatic identification which enables a faster and more efficient repair or substitution by the maintenance responsibilities. Moreover, it is the combination of two other parameters called display code and causes code, whose goal is, respectively, identifying the last displayed code and the last cause detected. Those detections can be relevant to verify if the number of faults identified by the appliance is recurrent or fills some pattern.

Furthermore, we can clearly classify our dataset as an unbalanced one which can influence the learning process since it can result in a classification preference for the majority class. However, this is also a reflection of more or fewer incident faults. For example, a fault that occurs frequently is a fault that is expected to occur with a higher frequency.

To verify the failure distribution, an initial analysis was conducted to identify which faults are recognized and what information can be retrieved from them. First, some knowledge about the Display Code and Cause Code was explored and is depicted in Table 3.1. Second, the failures distribution was analyzed considering the Fault Code. **G/8** proved to be the most representative, expressing 93,46% of the recognized faults (under a total of 41625).

However, the total of studied days is 417480 which demonstrates that only on 9,88% of the dataset failures were automatically identified. Thus, analysis regarding the unidentified ones shall be performed.

Furthermore, the appliances have two variables responsible for counting the number of blocking faults and locking faults. Blocking faults are related to temporary problems such as the temperature is too high and the appliance stopped working for some time, restarting when the temperature rises a reasonable value. In contrast, locking faults are related to persistent problems, which represents an unsuitable resolution if the reset can only be performed by the repairman, for example.

As a result, there are malfunctions labeled and unlabelled. So, our goal is to identify the unclassified ones, since the label observations can be used to construct the model where the initial task becomes a time-series classification problem.

Besides the variables previously described, there are also two additional variables concerning the appliance model and its software version. The connectivity gateway, is an important device installed in the boilers that allow data transmission. These two variables vary for each boiler but do not change over time, not being considered as time-series data. However, they

Display Code	Cause Code	Fault Code	Occurrences
A	1	A/1	1
	2	A/2	1
B	3	B/3	1
C	4	C/4	118
D	5	D/5	5
E	6	E/6	36
F	7	F/7	1
G	8	G/8	38901
H	9	H/9	1
I	10	I/10	1
	11	I/11	624
J	12	J/12	766
	13	J/13	726
K	14	K/14	1
	15	K/15	1
	16	K/15	78
L	17	L/16	3

Table 3.1: Fault information

have to be considered since some faults are related to a specific appliance or software version. Due to that fact, these string variables were not transformed into quantitative values.

Regarding the information that can be retrieved from the data, the dataset was created based on data that was collected during thirteen months including information from 1000 appliances, where are described, four different models. Moreover, the division between models is not similar. Table 3.2 describes the model distribution, where the *D* is the standard, representing 79,2% of all data. Also, considering that our dataset admits different water cycles, a study was performed to verify the cycles of each appliance model.

Appliance Model	Number of appliances	Data percentage	Water cycle
A	1	0,1%	Central Heating
B	63	6,3%	Central Heating
C	144	14,14%	Hot water
D	792	79,2%	Hot water

Table 3.2: Model distribution

As the main goal of this study is the identification and prediction of failures, a relation between failures and appliance model was created. Besides the number of recognized faults, analysis regarding the most critical ones was also considered. Table 3.3 describes the number of failures for each appliance model.

As it was expected, *D* has a higher number of recognized faults since it has a higher distribution on data. However, the critical faults can be the ones that are identified fewer

Appliance Model	Number of detected failures	Failure distribution
A	3	0,007%
B	485	1,17%
C	4729	11,36%
D	36045	87,46%

Table 3.3: Failure distribution within models

times, thus, analysis regarding the type of failure (Blocking/Locking) was performed and it is depicted on Table 3.4.

Display Code	Fault Type
A	Blocking
B	Locking
C	Blocking
D	Blocking
E	Blocking
F	Locking
G	Blocking
H	Locking
I	Blocking/Locking
J	Blocking/Locking
K	Locking
L	Locking

Table 3.4: Faults distribution within type

Despite the principal categorization into two types of failures, a second division was performed considering the severity of each failure which resulted in three ranges. The first stage of severity, identified as Light Failure, represents temporary faults previously classified as Blocking. Regarding Locking faults, they are representative of two ranges: Moderate and Severe. The first represents faults with a possible resolution by making use of boiler' documentation. Finally, Severe faults are described as faults that demand the intervention of a repair assistant, requiring a manual reset.

Within each range, there are various display codes. Light faults admit ten codes, Moderate seven, and finally, Severe faults represent five different codes. However, not all failures are described in the dataset, thus, Figure 3.1 depicts the distribution of each range.

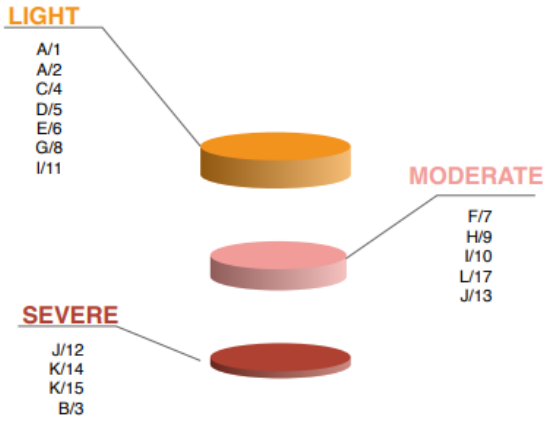

Figure 3.1: Failure distribution within range

3.1.1 Data Analysis

Regarding the acquisition time, our dataset considers thirteen months, although some of the appliances only have minimum information of one month. Besides that, despite the fact that data is collected every millisecond, only the variables for which there has been any change in the value or state are recorded. As an example, if there is no change in temperature the appliance will not send its value until the temperature is updated. Therefore, there is a need to verify if the empty space is caused by a malfunction such as connectivity, or if it is only an unchanged value.

As an example, in Table 3.5, it can be verified the issue of having missing data that does not always represent malfunctions but it can represent unchanged values. In the example, we considered 50 seconds where the three variables have been updated only when their values had changed, even if the appliance was always recording. Considering, for example, the variable "Label1", the value 23.7 has been recorded at 07:42:44.404 since there were no changes to its value in the next milliseconds. However, at 07:42:44.604 a new value was detected, 23.8. So, a new entry was recorded in the corresponding line. There are several ways to deal with this issue, one of them could be copying the last detected value until the new entry is detected, however, it could result in a fake translation of the boiler operation since if the temperature sensor broke we would have the idea of a stable temperature over time, for example. As a result, it is required to verify if there is connectivity between the appliance and the gateway, for example, to verify if it is a malfunction or only an unchanged value. This analysis and others are correctly explained next.

Furthermore, considering the dataset organization, two different techniques can be applied to analyze it: Random Access or Sequential-Access.

Since data has missing values it is required to implement a technique that is able to access an arbitrary element within a sequence or any item in data. *Random Access* allows access data organized in a matrix such as our dataset, therefore, based on the data position, we are able to implement search algorithms and then process data in parallel creating a faster

Time	Label1	Label2	Label3
07:42:44.304			1558.0
07:42:44.404	23.7	155.0	
07:42:44.504		156.0	
07:42:44.604	23.8		1559.0
07:42:44.704		155.0	1561.0

Table 3.5: Missing data

data processing. Considering that our dataset has missing values, the access should also allow going back and forward on data which is possible with the use of Random Acess. Moreover, Random Access enables easier forecasting due to the fact that it allows going through data as it is needed.

On the other hand, the main goal of *Sequential-Access* is accessing data in a sequence, not analyzing an element in specific. This approach enables the classification of a set of sequences and the identification of sequence patterns, useful to detect anomalies in data since they are represented as outliers in a sequence. Besides, this approach allows the study of individual characteristics and its similarities to an environment. Thus, it is possible the identification of normal operations and the correlation of variables that fill some characteristics.

This approach provides the application of PCA, *Principal Component Analysis*, to find correlations between variables such as the appliance mode and detected faults. Moreover, this algorithm can be applied during a period of time and then, if the correlation ends, a connectivity problem can occur, for example. Therefore, the implementation of Sequential-Access provides techniques to obtain knowledge about relations within data that can be critical for the predictive models' implementation.

Moreover, three specific behaviors are presented but are not labeled as faults, although they should be considered for our predictive models. First, data loss due to connectivity problems is a distinct fault from the inexistence of values related to sensors. However, they are both characterized as empty entries. In this specific study, value absence for all collected variables for more than four minutes is considered as data loss since it was a predefined given reference. Considering that, if some values are not updated in four minutes, a connectivity problem may happen.

Second, value absence reflected in empty entries is different from assuming that a variable has value zero. Considering for example temperature variables, a zero value is different from an empty value since a zero has a clear meaning in this case. Furthermore, since data is received within milliseconds, it is assumed that the last value has not suffered changes until a new value is recorded for this specific variable. However, as it was previously referenced, copying the last detected value can result in a fake translation of the boiler operation.

Finally, the third issue relates to the data collection process. It is common to have connection problems that lead to the first change being received after the second one. However, due to the enormous relationship between variables over time, this aspect could result in a

misinterpretation. Thus, it could lead to a fault identification, assuming some appliance fault, when in fact it is a connectivity problem.

3.1.2 Value Transcription Mechanism

Besides dealing with an unbalanced dataset, that contains some faults with greater representation than others, there is also the fact that the boilers only record data when there is a change on their values, as well as the fact that there are omissions resulting from connectivity failure.

As a result, it was necessary to create a Value Transcription Mechanism that validates not only the connectivity between the appliance and its gateway but also, that is able to resolve the missing data issue.

Since it is not possible to transcribe the previous value without verifying that at least there is connectivity between the remaining variables, a transcription technique has been developed in three steps. First, it is verified if there is connectivity between variables, being, second, verified if it ends some seconds after. If it does then a connectivity fault is raised. Finally, if the previous correlation still exists and the two first steps are validated, the value that was previously detected is transcribed.

Considering the previous example of missing data, this solution results on Table 3.6 which makes possible the implementation of Machine Learning algorithms.

Time	Label1	Label2	Label3
07:42:44.304	23.6	154.0	1558.0
07:42:44.404	23.7	155.0	1558.0
07:42:44.504	23.7	156.0	1558.0
07:42:44.604	23.8	156.0	1559.0
07:42:44.704	23.8	155.0	1561.0

Table 3.6: Transcription Result

However, this approach presents a significant disadvantage since data transcription can result in an incorrect translation of values.

3.2 Requirements

To develop an automatic platform that efficiently processes this type of data, and, also, deals with the previously mentioned issues, some requirements shall be met. Therefore, five requirements were established and are described on Table 3.7.

This work' goal is developing an application, implemented as a Big Data Service, that predicts appliances failures and can be used to deploy massive scenarios of PdM. To achieve it, first, the platform must be able to identify anomalies and forecast errors with a sufficiently large time window that allows possible correction methods to the appliance before the breakdown. However, as it was mentioned in chapter 2, the time window should be correctly adjusted since large ones can lead to False Negatives and small windows that can result in missing

Req#	Requirement
1	Accurate Anomaly Identification
2	Accurate Failure Forecasting
3	Efficient Data Processing
4	Ensure Scalability (vertical)
5	Storage Capacity

Table 3.7: System's requirements

important values. Thus, two requirements were defined: "Accurate Anomaly Identification" and "Accurate Failure Forecasting".

Besides, data processing shall be fast and efficient, allowing that multiple operations can be done at the same time and predictive methods can be applied to several boilers without corrupting each other. This requirement is depicted on Table 3.7 as "Efficient Data Processing".

Also, since we are dealing with a great amount of data, our system should ensure vertical scalability to support the increasing volume of data produced by IoT devices and sensors (with a special focus on heating appliances and boilers). Also, the solution shall have an efficient storage capacity since there are thousands of heating appliances that produce data every day. Therefore, these two requirements are defined as "Ensure Scalability (vertical)" and "Storage Capacity".

Moreover, requirements regarding performance and complexity shall be met. The implemented solution shall have an efficient performance, where the speedup can be analyzed and how each one of the previous requirements is fulfilled. Also, a complexity evaluation is relevant since a PdM system that is complex but does not report maintenance alerts in an efficient time window is not a sustainable solution.

Considering these requirements, the approaches described in chapter 2, are not sufficient to develop this kind of services by themselves. None of the systems reviewed previously satisfy all the requirements. In short, the independent use of those systems cannot solve the initial problem since they are not able to fulfill all the requirements.

3.3 Structure

Chapter 2 presented the necessary blocks to perform an effective PdM solution. Therefore, considering the previous requirements and the approaches presented before, a technology evaluation was made to verify their efficiency. The first part of the evaluation is related to Data Processing since it is required to have an efficient one to process the amount of available data. Since it requires continuous availability of computing and network resources, the solution should be based on Real-Time Processing. Regarding the described IoT platforms, both *SCoT* [21] and *ThingSpeak*[1] grant the requirement related to Data Processing. Thus, both platforms could be used as an infrastructure for the solution.

[1] https://thingspeak.com/

To analyze the effectiveness of the presented solutions regarding Data Persistence, some aspects such as queries were analyzed since our solution must deal with time-series data and must be able to perform several queries on it. Therefore, databases such as *TimescaleDB*[2], *Redis*[3], and *MapR*[4] were considered to be inefficient since they do not allow fast queries or even queries at all. So, analysis regarding processing was also considered. As a result, *OpenTSDB*[5] was recognized as unproductive since it is based on Hadoop (batch processing). Finally, the two databases that can fulfill the requirements related to storage capacity and data processing are *InfluxDB*[6] and *Apache Cassandra*[7].

Finally, to reply to the first two requirements, the Data Mining block shall combine three of the previous learning techniques. First, *Classification* methods shall be generated to distinguish faulty behaviors from non-faulty, being able to identify anomalies. Second, since our dataset has labeled and unlabeled data, *Clustering* tasks are required. And, finally, *Deep Learning* algorithms come in useful since they are very efficient on architectures with a large number of features.

So, as it was proved, the independent use of those technologies is not able to fulfill all the requirements. Consequently, corresponding to them, requires that an effective architecture must be implemented and, also, that it can be integrated with prediction methods.

To design the platform architecture, data acquisition, data persistence, and data processing were aspects that needed to be taken into account. Figure 3.2 describes the proposed block organization. First, *Data Source* interacts with the heating appliances obtaining data through their gateways. Since data is sent to each equipment' gateway, it is required to implement techniques that can access it and retrieve information from it. Thus, a second block should be implemented, entitled *Processing Block*. It receives data through a secure port and is responsible for the implementation of Data Processing techniques, Data Acquisition where brokers should be considered to get and send data to the necessary blocks. And, finally, it is also responsible for the Data Persistence where the information should be persisted.

Also, this block is responsible for the correct implementation of three requirements: efficient data processing, solution scalability, and storage capacity. Thus, it divides the three requirements through the three-block where the Data Processing block is responsible for the correct processing of all the information, and the Data Persistence block is answerable for the two other requirements.

Second, the Processing Block sends data through a secure port to the *Data Mining* block where the previously mentioned learning techniques are implemented. Since the main goal of a PdM system is failure prediction, the combination of Classification, Clustering, and Deep Learning algorithms can identify the incoming failures and their causes. As a result, the Data Mining block is responsible for returning the predictive results with efficient time.

[2]https://www.timescale.com/
[3]https://redis.io/
[4]https://mapr.com/
[5]http://opentsdb.net/
[6]https://www.influxdata.com/
[7]http://cassandra.apache.org/

Moreover, the DM block should guarantee the correct verification of the first two requirements where the anomalies shall be correctly identified and their forecasting shall be performed in an efficient time window. To develop prediction results and notify that maintenance is required, the Data Mining block can be structured into two parts: a predictive models generator which uses the historical data to generate the predictive results, and a connector to a dashboard.

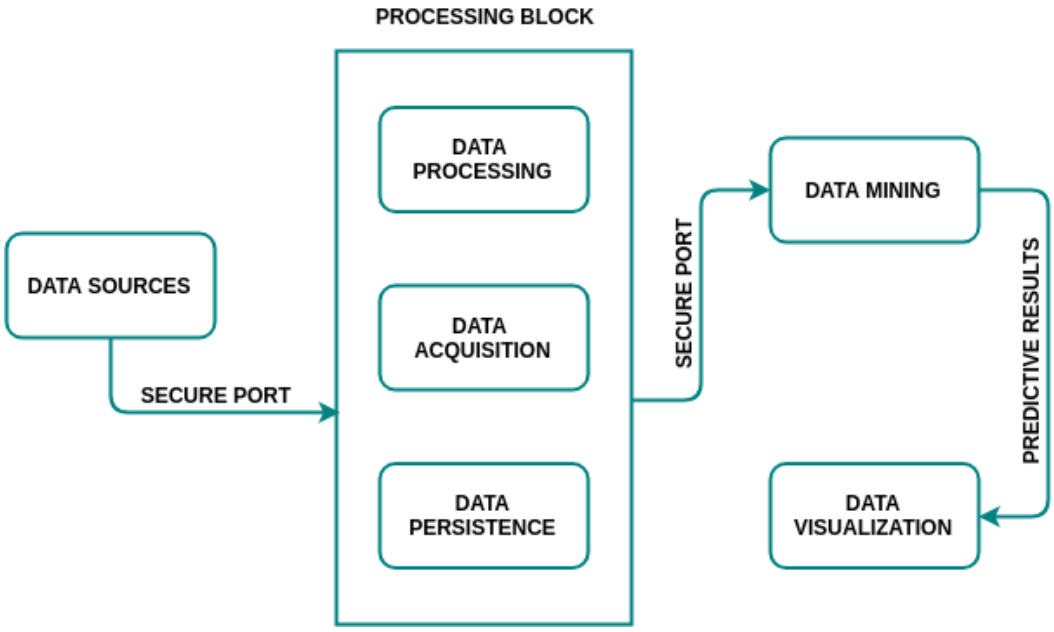

Figure 3.2: Solution architecture

The Predictive Models Generator obtains information from the storage and process it to remove useless information and prepare it to be used into the predictive models. This process is defined by three steps: first, the prediction solution is established and, based on its requirements, data is normalized and irrelevant information is removed. Finally, data is extracted to a staging area to be then used as an input of the prediction algorithm. Since this process relies on the implementation of the previously mentioned algorithms, it allows pattern detection, failure classification, and prediction.

Finally, when the predictive results are prepared, it is possible to send them through the connector to the dashboard. This dashboard is implemented on the *Data Visualization* block where maintenance alerts are represented and information about the heating appliances is shown. This information should be readjusted regarding the needs of the application user, increasing the user comfort which is an important goal of the presented solution.

Besides allowing notifications and alerting, the application shall have a user-friendly design. Moreover, since it is intended to alert for maintenance requests, it is critical that the dashboard facilitates annotations and filters showing significant events such as unknown failures. The connection between the Data Visualization and Data Mining blocks is described on Figure 3.3 where the operations of each block are described, and, also which information shall be sent to one block to the other.

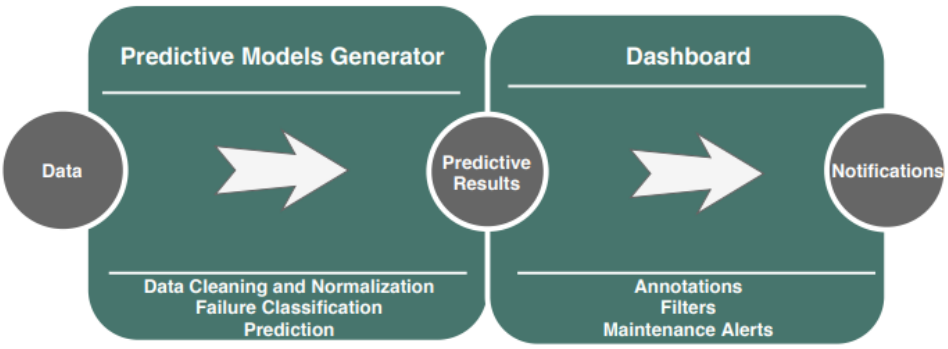

Figure 3.3: Connection between Data Mining and Data Visualization blocks

Chapter 4

Implementation

"Data by itself is useless. Data is only useful if you apply it. "

Todd Park

Considering the solution proposed before, its implementation is described in this chapter. Thus, the chapter is organized as follows: section 4.1 describes the chosen architecture for the solution implementation and section 4.2 describes the Data Mining implementation.

4.1 Architecture

As it was described in chapter 3, the solution should be remotely distributed, efficient, scalable, and fault tolerant to provide an efficient Predictive Maintenance system that can be used in the future for others Big Data problems. Figure 4.1 depicts the proposed architecture of the *SCoTv2* platform, considering that the Data Mining techniques are a vertical module of it and this platform will be used as the Processing Block of our solution, previously mentioned.

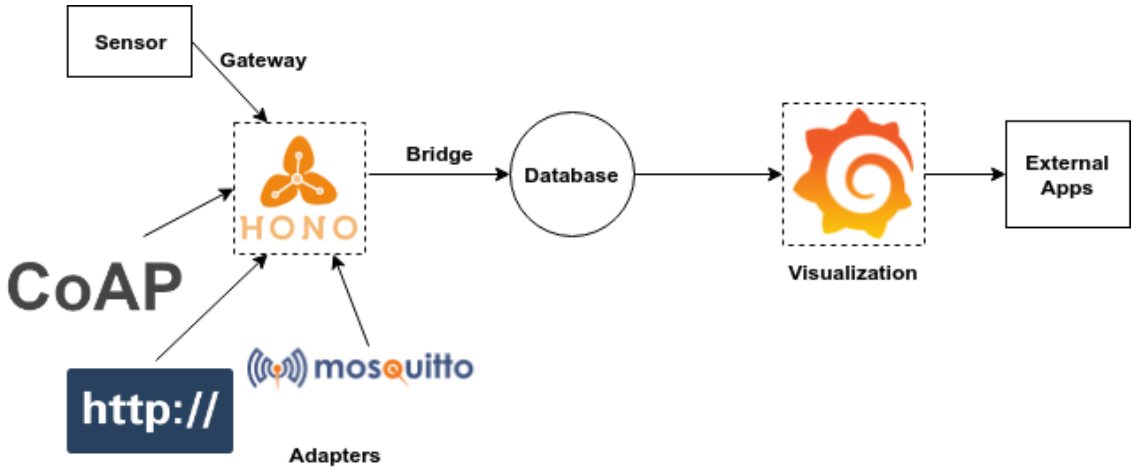

Figure 4.1: Processing Block of the proposed architecture

This block, represented as a full architecture, promotes data acquisition, data processing, and data storage being able to guarantee its persistence. Moreover, since it is a Machine to

Machine (M2M) solution, it allows the connection of several gateways and, therefore, task parallelization resulting in effective data processing.

Considering its organization, the information retrieval from the boilers gateways is implemented with the combination of three technologies, CoAP[1], MQTT[2], and HTTP. Constrained Application Protocol (CoAP) is a specialized web protocol for use with constrained nodes and in the IoT context, and MQTT is an M2M protocol that allows connections between remote locations where a small code footprint is required. Finally, the HyperText Transfer Protocol (HTTP) allows communication between clients and servers. Consequently, their combination results in an adequate data acquisition since they are able to promote valuable communication between boilers and the remaining blocks.

Furthermore, the data shall be sent to the remaining architecture. This process is implemented with Eclipse Hono[3] which provides remote service interfaces for connecting a considerable number of IoT devices and can interact with a backend regardless of the device communication protocol. As a result, this connection promotes fast and suitable data processing.

Regarding data storage, data is sent through a bridge to a database. Being important to implement a Time Series Database as we are dealing with time-series data, several solutions can be implemented such as Apache Cassandra, PostgreSQL or InfluxDB. The first one is a highly scalable columnar database that provides high availability without compromising performance. It is fault-tolerant on cloud infrastructure, rather relevant on platforms that administer critical data such as the one presented.

Also, as it is intended to predict failures and alert the boiler user that maintenance is required, the system shall have a visualization block. It is achieved with the use of Grafana[4], an open-source platform that has a vast diversity of database connectors granting a dynamic dashboard that can have combined sources. Moreover, it is imperative that it enables annotations and filters showing meaningful events such as unknown failures.

Finally, the Processing Block is connected to External Applications such as the Data Mining Block. This block is responsible for the Machine Learning techniques (depicted on Figure 4.2). The Data Mining processes and elements are described in the next section. Regarding the technologies that are implemented on this block, all the algorithms are developed on Python[5] language, which interacts with Neo4j[6] to develop the Prediction solution. To connect these technologies, py2neo[7] is used since it is a client library for working with Neo4j from within Python applications. The prediction algorithm relies on sklearn[8], a simple and efficient toolkit for data mining and data analysis. Moreover, Keras[9] is used for the Deep

[1] https://coap.technology/
[2] http://mqtt.org/
[3] https://www.eclipse.org/hono/
[4] https://grafana.com/
[5] https://www.python.org/
[6] https://neo4j.com/
[7] https://py2neo.org/
[8] https://scikit-learn.org/stable/
[9] https://keras.io/

Learning implementation since it allows easy and fast prototyping, and it runs well over CPU ou GPU. Also, this API is capable of running on top of TensorFlow[10] which helps the development and train of ML models.

Figure 4.2: Data Mining Technologies Description

After the implementation of the algorithms, it is possible to obtain the predictive results that can be, after, sent to the Data Visualization block. This block intends to notify the user that maintenance is required and failure was predicted. Therefore, a possible solution for dashboard implementation is Grafana[11]. Grafana is an open-source platform that has a great variety of database connectors granting a dynamic dashboard that can combined sources.

4.2 Automatic Fault Identification in Time-Series data

The first step in implementing an automatic temporal data fault identification solution was the analysis of the dataset, which contains device operational data. Since real data is considered, it is more frequent to find normal operation than failure states, which presents both advantages and limitations. The immediate conclusion is that we must operate over an unbalanced dataset as the frequency of outliers is lower than the normal states (standard operation), creating the issue that some classes have a bigger distribution than others, creating bias in the classification algorithms which have a preference on the most distributed ones.

Considering all the issues present on data, an evaluation regarding machine learning algorithms was made. As the project main goal is predicting failures, five methods were chosen to be analyzed considering their relevance on prediction scenarios.

First, Support Vector Machines are non-probabilistic classifiers applied to establish hyperplanes between training classes to find the largest margin between data, being effective on Predictive Maintenance systems [7] such as the one presented. As the dataset contains multivariate variables and it is pretended to find failures/outliers, an SVM can be implemented considering that the more distant points can be an outlier. However, despite being one of the most powerful out-of-the-box classifiers, its scalability should be considered as it is computationally complex.

[10] https://www.tensorflow.org/
[11] https://grafana.com/

To reduce the scalability issue, an extension for regression problems can be developed, defining a Support Vector Regression. Able to perform time-series data prediction [40], this algorithm requires fewer data to obtain results.

As Regression generates a function that approximates the input result to produce continuous predictions, and Classification aims to distinguish faulty behaviors from non-faulty, these two learning functions can be merged and a Classification and Regression Tree (CART) can be produced. CART uses an input space split into sections to assign classification and regression to each of them [36]. Therefore, the classification methods focus on predict the class of each section and the regression methods on predicting a constant of the assigned region. Finally, the two processes can be consolidated into a single predictive process and expressed as a decision-making process. Although they are easy to be interpreted by humans and possible to be scaled for large datasets, such as ours, their performance is totally dependent on a suitable training set.

Furthermore, the combination of learning methods can be an effective solution to obtain a better prediction, known as Ensemble Learning. Those methods come in useful since they run based on a local search with several starting points, increasing the algorithm performance. Since prediction problems are usually associated with nonlinearity, these methods are relevant [47] as they approximate functions based on a sum of several hypotheses.

Besides, problems of complex scenarios such as Smart Homes are being solved with the implementation of a Neural Network [41]. Quite efficient in prediction and easy to be adapted to classification and regression problems, an ANN can be applied as an Ensemble Method [50]. Also, used as a combination of several networks, it improves the forecasting accuracy and robustness, useful on time-series data problems.

Considering that the solution shall be able to deal with a large amount of data and, also, the fact that the dataset is unbalanced, a combination of learning techniques shall be performed. Thus, by the previous analysis, the solution shall rely on a Neural Network. Moreover, since the dataset contains normal operation states, an Neural Network designed for classification comes in useful as the main goal is identifying the faulty behaviors and predict them.

4.2.1 Parameter Selection

Before implementing a ML algorithm, some data normalization shall be implemented such as Parameter Selection that guarantees that the defined algorithm relies only on the relevant parameters. Considering the presented dataset and the previous data analysis, it is possible to exclude variables with no labeling influence, such as those with no value variations from label to label. This is possible through data visualization of normal operation cycles and identified faults.

However, there are also variables with no saved values since some specific sensors only exist in some appliances, not considered in this study. As our dataset has 91 parameters but not all of them are relevant for the predictive models it was needed to implement a strategy that could identify the most relevant ones. As a result, a Random Forest Classifier (RFC) was

implemented to distinguish the most influencing parameters since it allows the classification of important labels and it has a good performance. Furthermore, we were able to reduce our dataset to 23 variables, around 25%, since they were the ones that represented a bigger covariance, thus, higher importance on the data.

Consequently, to verify the impact of Parameter Selection in the training time of the Random Forest Classifier, we implemented 10 trials and then recorded the training time before and after the Parameter Selection. The obtained results are represented in Table 4.1, which shows the average, the median values, and the corresponding speed up for a different number of boilers, therefore, a different data size, allowing us to verify if the training time increases with the increase of data.

Data size		Average	Median	Speed up
81,12GB	Before FS	3.41s	3.35s	6.16 %
	After FS	3.20s	3.17s	
173,40GB	Before FS	4.38s	4.33s	4.11%
	After FS	4.20s	4.19s	
280,01GB	Before FS	5.08s	4.99s	10.83%
	After FS	4.53s	4.32s	

Table 4.1: Effect of Parameter Selection on RFC

By the analysis of the results, we concluded that Parameter Selection was able to reduce the training time of the Random Forest Classifier due to the fact that our dataset has been minimized and irrelevant parameters were discarded. Also, to understand the correlation between the selected parameters a heatmap was created as is depicted on Figure 4.3.

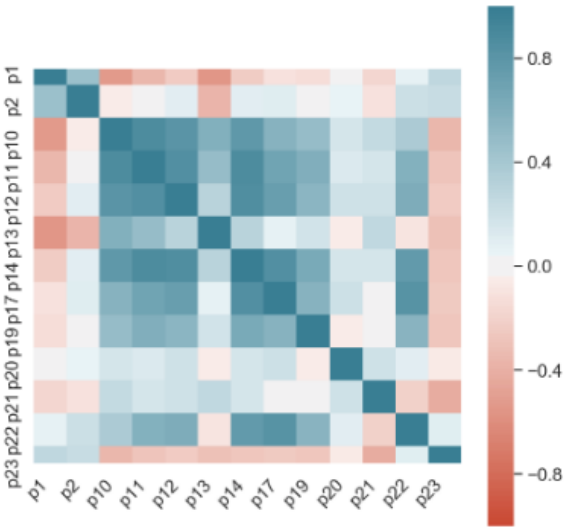

Figure 4.3: Selected Parameters Correlation Heatmap

4.2.2 Failures Classification

Furthermore, before the prediction implementation, a simple classification was performed to understand how the dataset is organized and which parameters represent a better utility.

Since a complete classification report can give us significant knowledge about the label's distribution within the dataset, and how it influences the results, a Random Forest Classifier was implemented to classify failures on the dataset and is depicted on Table 4.2. This algorithm was used since it has a good performance and is simple to understand how the unbalanced dataset has an impact on the results.

Data size	Average	Precision	Recall	F1-score
81.12GB	Micro	31%	31%	31%
	Macro	24%	34%	26%
	Weighted	23%	31%	24%
173.40GB	Micro	32%	32%	32%
	Macro	24%	34%	26%
	Weighted	24%	32%	25%
280.01GB	Micro	35%	35%	35%
	Macro	25%	34%	26%
	Weighted	26%	35%	27%

Table 4.2: Random Forest Classifier results

As micro and macro averages have near values, it is possible to derive that the dataset has its labels equally populated. As an outcome, it produced a weighted average with a similar value (27.4%).

It is also crucial to acquire if all the selected labels are effective for the classification method. So, the Recall was also retrieved and had a 33.11% average, which shows that not all of the chosen labels are important. Moreover, an F1-score with an average of less than 50% can represent that the algorithm is picking labels that can be irrelevant elements.

However, the low precision can be related to the fact that the models are developed using an unbalanced dataset. Therefore, a solution to face this issue should be performed which is explained next.

4.2.3 Prediction Implementation

Seeing that we are dealing with stream data, i.e. measurements of various aspects recorded on a time, our data can be classified as a collection of temporal objects. Also, an event registered at $t2$ has a temporal relation with another one detected at $t1$, so an effective solution can be through the implementation of a Recurrent Neural Network.

This network has at least one feedback cycle that transfers the information generated at time $t1$ as input for time $t2$. This dependence creates a dynamic behavior [38] since the produced output is used as an input. Moreover, as this recurrence requires memory to store information, RNNs come in useful to process sequences of inputs known as not segmented problems.

Furthermore, Simple Recurrent Neural Networks have a relevant problem - they are not able to capture long dependences on sequences. Known as Long Short-Term Memory (see Figure 4.4), this internal memory is able to store information and process it through the connected layers. As it was explained before, our solution shall be able to handle an

unbalanced dataset. There are several solutions such as collect more data or resample the dataset, however, since the used dataset has a considerable size these solutions are not effective.

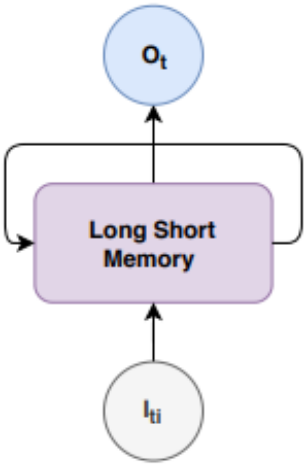

Figure 4.4: Long Short-Term Memory

Moreover, a possible solution could divide all dataset into small datasets and perform prediction on each part. However, this solution requires a precise division which results in time spent, being an ineffective resolution.

Since it was possible to identify the sequencing process, it was possible to obtain a state machine and, calculating the states transition probabilities by the Naive Bayes probability implementation. Finally, a Markov Chain was constructed and will be used as input from the Neural Network (see Figure 4.5). This solution, implemented as a Model-Based Approach, has important advantages since it is able to continuously control a process to find the best solution without blocking normal operations.

Also, combining the approach with Machine Learning algorithms such as Neural Networks, enables error detection and is dependent on the quality and quantity of the data. To guarantee the quality of data it is necessary to reduce the impact of an unbalanced dataset, thus, by assigning each process to a state, it is possible to reduce the unbalancing previously detected since a resample is made without requiring new data. The Markov Chain development is explained in the next sections.

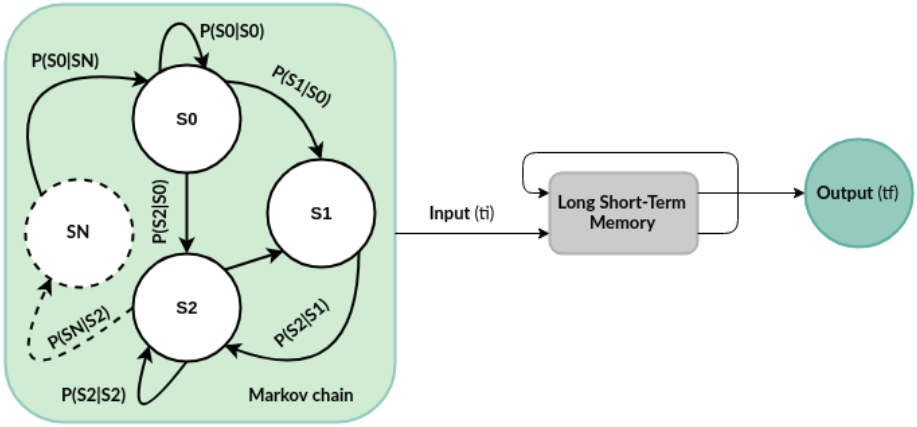

Figure 4.5: Markov Chain as an LSTM input

Data Normalization

Since it is intended to predict failures in one-week (5 days) time advance, it is required that the constructed Markov Chain has at least information from one completed month. Thus, an analysis regarding the time information was performed and 46 appliances were excluded as they did not have information from one completed month. Also, this analysis resulted in a reduction of 8.80% of our dataset, and it was also possible to verify that only seven appliances have information from ten different months.

Considering the actual models' distribution, an impact analysis was performed and the final distribution is described on Table 4.3. Despite the data reduction, *D* remains as the most representative model.

Appliance Model	Number of appliances	Data percentage
A	1	0.10%
B	62	6.50%
C	139	14.57%
D	752	78.83%

Table 4.3: Final model distribution

Moreover, this reduction had an impact on the detected failures. Thus, the final distribution within the severity ranges is depicted on Figure 4.6. As it can be seen, Light Faults have now four different group codes instead of six, Moderate Faults have three despite the previous five, and Severe Faults admit two groups against the previous three.

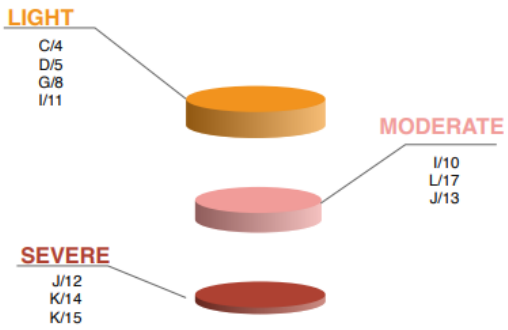

Figure 4.6: Final Failure Distribution within ranges

Sequencing Process Translation

After retrieving this knowledge from data, it was possible to create a sequencing process, depicted on Figure 4.7 that makes possible to understand how the failures are generated, which variables produce them, and from which severity range are they.

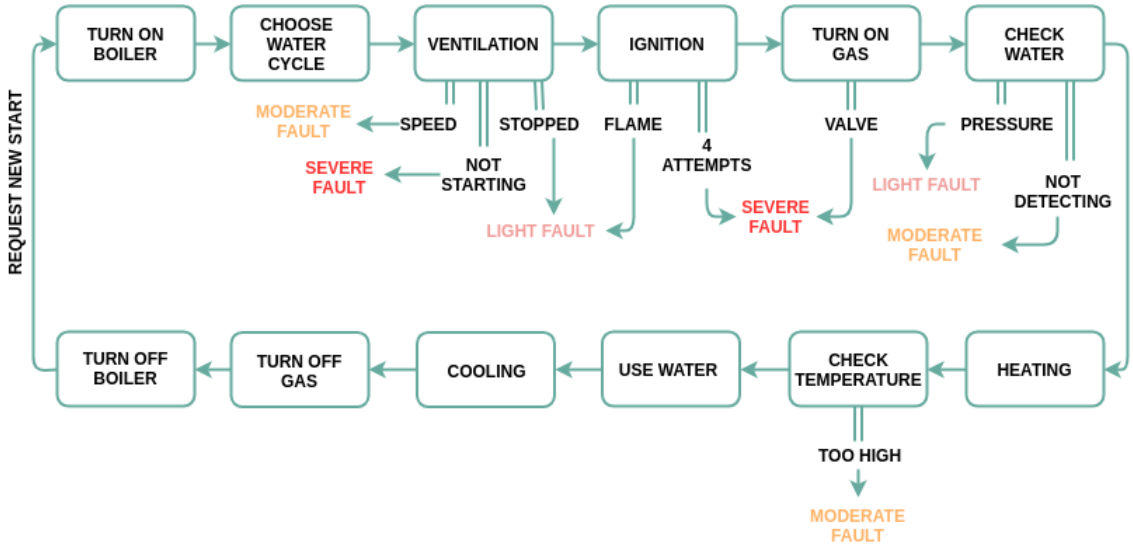

Figure 4.7: Sequencing Process

Moreover, it is possible to understand how the variables change and how are they correlated, it is possible to verify that the appliances in study start their function by an order. This operation is defined as "Turn on boiler" and is followed by the choice of the corresponding water cycle identified by the "Choose Water Cycle" process. After choosing the function to perform, the appliance starts the "Ventilation" process where three problems can arise: speed, start and stop suddenly. If this process runs without any kind of failure, the process of "Ignition" starts. In this process two problems can occur: the flame does not start correctly or after four attempts there is no flame.

The "Turn on gas" process starts after the flame has lit correctly, and in this process, it

is possible to fail the gas valve which invalidates the rest of the sequence. After the gas is switched on, the water heating process starts with "Check Water" to check the water pressure. If the water is correctly detected, the water is heated in order to respond to the initial request of the user of the equipment.

Since the appliance supports the water temperature checking feature, this process starts and is identified as "Check Temperature". If the temperature is higher than a previously defined maximum, an error is generated. If not, the user can perform its task ("Use Water") and after finishing it, the water is cooled, the gas is turned off ("Turn off gas") and the equipment is turned off ("Turn off boiler"). As soon as there is a new request to use water, the sequence repeats itself.

Moreover, this sequencing allowed the identification of four important failure causes which are Boiler Identification, Ignition, Temperature, and Usage. Also, information regarding the failure is essential for its characterization. Thus, an analysis regarding parameters distribution within these groups was performed and is depicted on Table 4.4.

Failure Cause	Number of parameters
Boiler Identification	4
Failure Identification	5
Ignition	4
Temperature	7
Usage	3

Table 4.4: Parameters distribution within Failures Group Causes

After, it is required to translate the sequencing process into a state machine. By its analysis, two different states were defined: transition and error. The first group relates to twelve stages and the second to nine. This definition, combined with the previously selected parameters, and their groups, allows the sequencing process translation to a Markov Chain as the transaction probability is implemented by querying data and calculating the Naive Bayes probability.

This probability is quite easy to implement and fast to predict classes even in multi-class problems, such as this case. Moreover, it has a good performance with categorical inputs (textual variables). As an example, Figure 4.8 depicts the calculation of the transaction between the ventilation and ignition states, respectively.

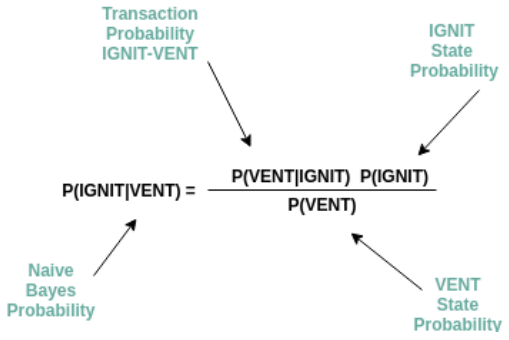

Figure 4.8: Naive Bayes Probability Example

The state machine implementation is done through a native graph database, Neo4j, a graph platform that manages operations such as creating, reading, updating, and deleting on a graph data model. In this approach, each graph represents information from one month where each node describes one state, and each relationship represents how two transition states are associated. This structure allows modeling all kinds of scenarios, thus, different failures. Figure 4.9 depicts the graph of one-month information from one appliance where the red nodes are the error states and blue ones are the transition states. Moreover, it is also possible to verify which parameters have an influence on the transaction between states.

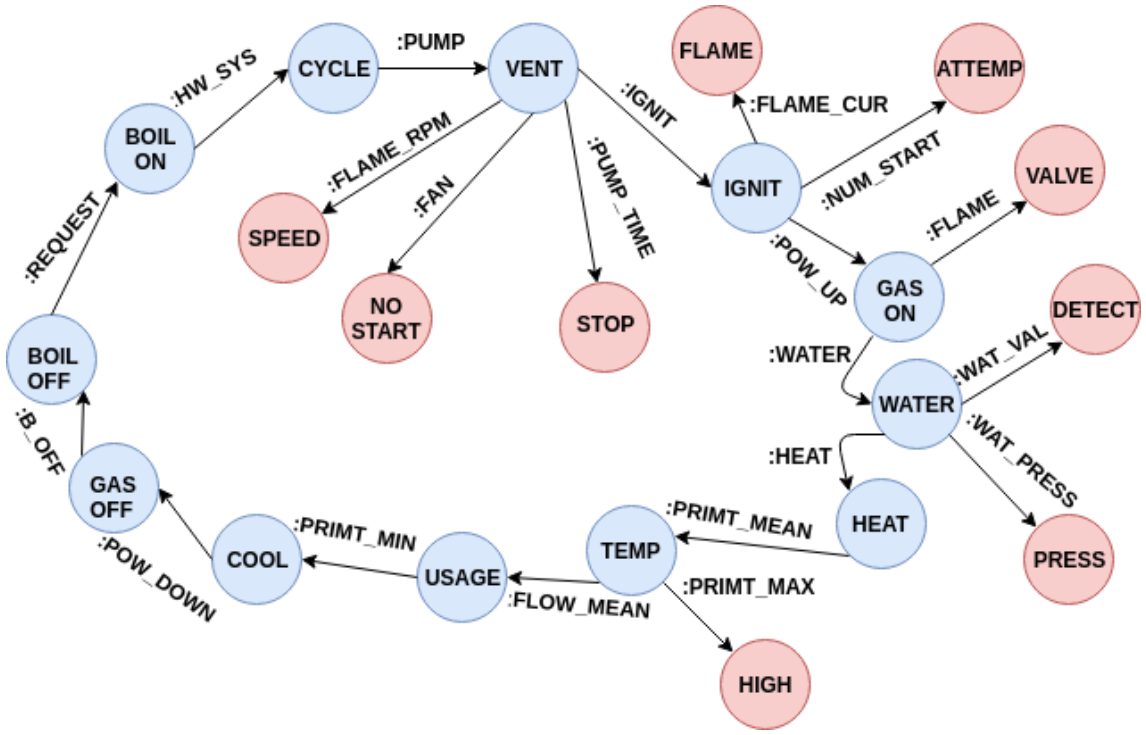

Figure 4.9: One month state machine

As the dataset contains 954 appliances with complete months, which results in 5949 months, the graph database will have the same number of graphs. Despite the two previous mentioned states, the final result of the RNN will be a combination of a severity range and a failure cause within the previous five. After, the failure will be identified by matching the two results and identifying the code that corresponds to the range and failure cause. If we could summarize the processing done to the data from its reception to the implementation of the prediction algorithms, the workflow representation would be the same as shown in the figure Figure 4.10 where the darker blocks represent simpler results obtained during the process.

Moreover, preparing data for the LSTM implementation involves framing the dataset as a supervised problem and normalizing the variables. Also, LSTM shall be fitted to work with multivariate input data. Fitting requires split the prepared dataset into train and test sets which is described in the next section.

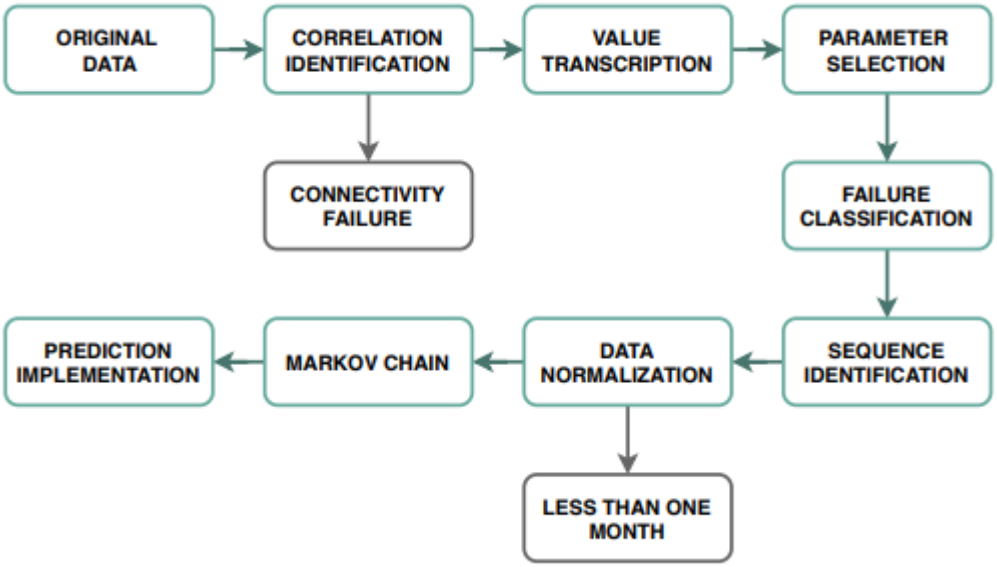

Figure 4.10: Data Workflow

4.2.4 Dataset division

Since data used usually comes from multiple datasets, a well-performed learning algorithm relies on a well-divided dataset. Particularly, the initial dataset is split into three sets applied in different stages of the model.

First, the training set is used to train the implemented model; second, the testing set evaluates the model performance; and, finally, the validation set works as an independent metric for a better evaluation.

Furthermore, considering the two main division techniques: cross and split validation, and the fact that this solution works with time-series data, a split validation was performed. Since it relies on removing a part of the training data and using it to get predictions from the trained model on the rest of the data, Cross-validation could not be used. As our dataset contains time-series data, this division could, also, result in the analysis of an event that has not happened already. Moreover, is not certain which data points will end up in the validation set and the result might be completely different. However, if the divison is made based on an unique appliance, this technique can be implemented.

Besides, reducing the training data can imply losing important patterns, which increases error induced by bias. So, a split-validation was implemented. This technique randomly splits up the dataset into a training and test set and evaluates the model. Also, the model performance is measured during the testing phase. Comparing to the previous method, split-validation is done in a single iteration, opposite to cross-validation that requires a number of iterations using different subsets for testing and training purposes.

Furthermore, divide time-series data requires that the training set comes before the test set, so a consideration regarding date shall be considered. Also, as the dataset has a considerable size, the division implemented was 70% for training, 20% for testing, and, the

remaining 10% for validation. As the completed months were 5949, the predictive models were trained with around 4165 months, tested with 1190, and validated with 594 months.

Chapter 5

Experimental Results

"You don't get any medal for trying something, you get medals for results."
Bill Parcells

Considering the requirements defined in Table 3.7, our main results are about ensuring scalability, guaranteeing an efficient data processing and verify the effectiveness of the predictive models. Also, an impact evaluation was made to verify if our solution had an effective impact and a comparison with other relevant machine learning algorithms was performed. Thus, this section is organized as follows: section 5.1 relates the effectiveness of our solution on fulfill the proposed requirements, section 5.2 reports the predictive results, section 5.3 refers to the impact evaluation, and, finally section 5.4 compares the implemented solution with other ML algorithms.

5.1 Requirements evaluation

Considering the requirements defined in chapter 3, some tests were performed to verify their validity. As the first two requirements relate to Anomaly Detection and Prediction, an evaluation was made regarding the classification and prediction results.

Besides, the solution should guarantee efficient data processing and storage since the volume of data that is produced by the IoT devices increases every day. Also, vertical scalability shall be assured.

5.1.1 Anomaly Identification and Failure Forecasting

Predictive Maintenance Systems intend to recognize incoming failures before they occur, ensuring that important operations are not disrupted as actions can be taken to prevent them. Therefore, the first two requirements are related to correctly identifying anomalies and forecast them with a sufficiently large time window. Also, the defined time window shall be adjusted since large ones can lead to False Positives and the small ones to False Negatives since important values can be missed.

By analyzing the Classification results (see Table 4.2) we can attest that were able to identify anomalies with an average of 35%. However, as it was previously mentioned, dealing with an unbalanced dataset creates a bias on the classification algorithms.

Also, as PdM Systems intend to predict failures, an evaluation regarding the forecasting shall be done. As a large time window can result on False Positives, the decided time window to be analyzed was 1 month to predict 1 week which results on analyzing time information four times larger than the one that will be forecasted. Therefore, we can attest that an efficient time window is guaranteed. Regarding the prediction results, they will be analyzed and evaluated on section 5.2 presented next.

5.1.2 Data Processing, Scalability, and Storage Capacity

The last three requirements intend that the platform shall be fast and efficient enough to allow multiple operations at the same period, ensure vertical scalability to support the increase of data, and, also, to have efficient storage able to store thousands of information from several heating appliances.

Therefore, to verify if the processing time was influenced by the data increasing, an evaluation regarding the time that is needed to process some of the data was made. This evaluation was made using a Kafka cluster since the connection with the Processing Block (see Figure 4.1) was not fully established. This cluster processed the data of the Data Persistence block, in order to verify the time necessary to transmit the data. First, it was used a node with a 32-bit CPU and 4GB RAM, when data exceed the 200GB it was necessary to duplicate the number of cluster nodes. The results are represented in Table 5.1.

Data size	Average	Standard Deviation
81.12GB	289.99s	7.78s
173.40GB	424.80s	15.09s
280.01GB	497.29s	26.09s

Table 5.1: Data Processing Times

The time average shows us that the time processing increases near the 100s when data duplicates. Despite the fact that it shall be reduced, it is possible to attest that the proposed system allows scalability since the time increasing is supportable. Moreover, the standard deviation of the results demonstrates that they follow a range of values near the average representing a low error percentage.

Besides guaranteeing vertical scalability, the results demonstrate that sustainable data processing is possible as the increasing time ratio is supportable. And, also, that an efficient storage capacity is ensured.

5.2 Predictive Results

As we are working with a Recurrent Neural Network, our dataset shall be prepared for the LSTM implementation. This requires farming the dataset as a supervised learning problem and normalizing the input variables.

To analyze the impact of choosing different numbers of neurons on the layers, four different tests were made. First, the LSTM was established with one neuron in the first hidden layer, after with ten, fifty, and, finally, with a hundred neurons. As the number of neurons increased, so did the number of hidden layers, being half the number of neurons per layer. Regarding the output layer, two neurons were chosen since the RNN returns two different results. The input shape will be a 1-time step with 23 features, confirmed as the ones to have a major influence on data.

Also, a Mean Absolute Error (MAE) loss function and an Adam version of Stochastic Gradient Descent (SGD) were applied. MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. Different from Root Mean Squared Error (RMSE) that also measures the average magnitude of the error, however, RMSE comes in useful when large errors are particularly undesirable which is not the case. Regarding SGD, the Adaptive Moment Estimation (Adam) version is a method that computes adaptive learning rates for each parameter which allow us to verify the effectiveness of each variable.

To obtain complete information regarding the predictive results, the error information are depicted on Table 5.2. The last two columns evaluate the Mean Square and Absolute Errors that compute the average between the predicted values and what is estimated.

Neurons	Hidden Layers	MSE	MAE
1	1	85.3	91.83
10	5	98.41	31.35
50	25	38.56	19.59
100	50	29.20	6.24

Table 5.2: Prediction Results

Also, the Hamming Loss was evaluated and it had an average of 0.095 which computes the fraction of the wrong labels to the total number of labels. This result describes an effective fraction as it is closer to zero.

Regarding the error information, it is possible to conclude that the prediction has a lower error when the number of nodes is higher. Also, the Absolute Error relates the number of errors in the measurements, thus, the increasing of neurons as a positive effect on its result. The Mean Squared Error indicates how closely a regression line is to a set of points, thus, the smaller the error, the closer is the solution to find the best fit. Since the MSE decreases with the increasing of neurons, there is also a positive effect.

Moreover, to obtain a complete classification report about the RNN implementation, the Precision, Recall, and F1-score were retrieved and are depicted on Table 5.3. With an average Precision of 82.10% and an F1-score of more that 50%, it is possible to attest that our solution is picking the correct labels and the impact of an unbalanced dataset was reduced. Also, the prediction results regarding the output demonstrated a higher value for the *Light* severity range and *Boiler Identification* was the failure cause with a higher prediction.

Average	Precision	Recall	F1-score
Micro	88%	68%	75%
Macro	73%	66%	70%
Weighted	85%	67%	77%

Table 5.3: RNN Classification Report

5.3 Impact evaluation

Considering the main goals of the presented work, the solution impact should be evaluated regarding two different aspects: were we able to increase the comfort of the boiler user? And, is the time processing efficient for this type of problem?

Considering those evaluations, the following section is divided into two subsections: *User comfort*, relates the impact of the failure identification on the user comfort, and *Performance Evaluation* reports a theoretical study of the speed up represented before.

5.3.1 User comfort

Since one of the project goals is increasing the user comfort and increase the equipment' lifetime, a study was made to evaluate if the failure classification and prediction, described in the previous sections, had a positive influence on those aspects.

Considering that the macro-average precision is 73% (see Table 5.3) and, also, that predictions have one week-time (5 days) window, it is possible to infer that anomalies were predicted and the downtime can be reduced.

Although it has, currently, a small-time window, the proposed solution can be scalable to bigger ones such as one moth-time. This scalability makes possible the prediction of more than 70% of the failures, proving that is more efficient in predicting failures with time in advance. This forecasting can emerge as an increase of the remaining lifetime of the boilers, resulting in higher user comfort.

Besides, identifying and predicting failures results in an important improvement of the number of appliances that require maintenance.

5.3.2 Performance Evaluation

To evaluate the solution performance, an estimation based on the previous speed up results was conducted. Figure 5.1 represents it and the speed up values are illustrated by the blue points and their linear regression by the red line. Besides its representation, the mathematical formulation was obtained in order to verify how the data size, denoted as x, can influence the speed up expressed as y. Moreover, to verify the improvement compared to the previous batch solution [3] a theoretical estimation of its behavior was performed, which is drawn by the blue line.

As batch processing remains on process blocks of data, the data expansion results on a time increasing. As a result, the speed up grows proportionally to the escalation of data having exponential growth.

On the opposite, the presented solution has an exponential increase (46%) at the begin where around 80GB are processed and becomes stable, as the obtained mathematical expression shown. It entails that the solution starts with exponential growth, but then it becomes to have a stabilized value, having a logarithm behavior.

Comparing the two behaviors, it is possible to conclude that the speed up in batch processing will rise to a value that requires too much time to process all data, demanding that hardware solutions shall be performed to be viable. However, the proposed solution will have a sustainable value that outlines a viable time to process all the information. Moreover, by the analysis of Figure 5.1, it is possible to ensure that the presented PdM system becomes more effective than batch processing when data size is bigger than 300GB which expresses a considerable amount of data.

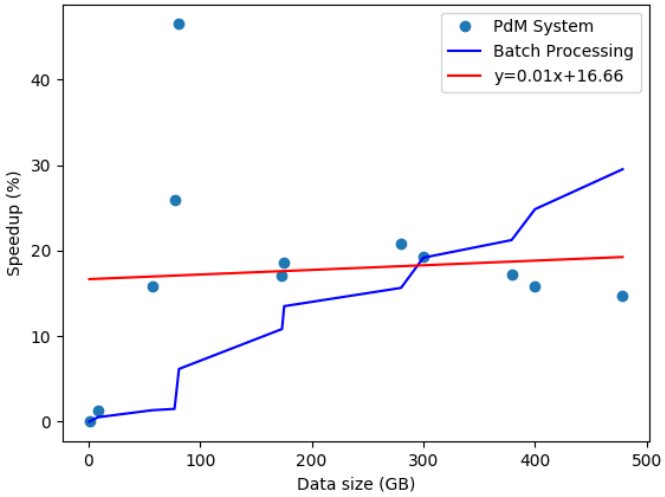

Figure 5.1: Comparison between Batch Processing and PdM System

5.4 Machine Learning Algorithms Comparison

Considering the three Machine Learning Algorithms identified as possible solutions on section 4.2, a comparison was performed to verify if the chosen method was the correct one. Therefore, this section is organized into two comparisons. First, the Recurrent Neural Network is compared to a Support Vector Machine. And, finally, a Classification and Regression Tree is contrasted with the RNN to verify which one is more effective. All the algorithms were implemented with the sklearn toolkit and evaluated regarding the same division into the train, test, and validation tests.

5.4.1 Support Vector Machine

Support Vector Machines are non-probabilistic classifiers applied to establish hyperplanes between training classes to find the largest margin between data. Considering the used dataset, this algorithm was implemented by identifying the most distant points as outliers.

Implemented with the Logistic Regressor module of the sklearn toolkit and a linear kernel, this algorithm is computationally complex. However, a complete classification report was obtained and is depicted on Table 5.4.

Average	Precision	Recall	F1-score
Micro	87%	67%	80%
Macro	72%	65%	67%
Weighted	86%	66%	78%

Table 5.4: Support Vector Machine Results

Considering the previous results, it is possible to verify that this algorithm identified the outliers and predicted them with a Precision average of 81.67%, which is an efficient result. Also, a Recall average of 66% indicates that the selected labels for classification were efficient. Moreover, an F1-score of more than 50% represents that the picked labels were relevant elements of the dataset.

5.4.2 Classification and Regression Tree

Classification and Regression Trees split the input into sections to assign classification and regression to each of them. Finally, the two processes can be consolidated into a single predictive process and expressed as a decision-making process.

Although their performance is totally dependent on a suitable training set, these trees are possible to be scaled to large datasets such as the one presented. Implemented with the Decision Tree Classifier and Regressor modules of the sklearn toolkit, this algorithm was implemented with 5 layers depth and a minimum number of rows per node equals to 10. The obtained results are depicted on Table 5.5.

Average	Precision	Recall	F1-score
Micro	75%	67%	70%
Macro	67%	56%	60%
Weighted	60%	49%	61%

Table 5.5: Classification and Regression Tree Results

Analysing the previous table, it is possible to verify that the CART algorithm predicted failures with a Precision average of 67.33%. Also, a Recall average of 57.33% and an F1-score of more than 50% indicates that the algorithm selected the relevant labels.

5.4.3 Predictive Results Comparison

To compare the three methods, a table was created and is depicted on Table 5.6. By its analysis, it is possible to attest that the solution presented in this dissertation had a similar precision to the SVM algorithm. The average time of each algorithm is also represented since it is relevant to attest the speed of train, test, and prediction.

Although the RNN had a similar precision than the SVM algorithm, the second method requires more time to be trained and tested. Being a computationally complex algorithm,

Method	Precision	Average Time
RNN	82.10%	164s
SVM	81.67%	287s
CART	67.33%	365s

Table 5.6: Comparison of Machine Learning Algorithms

this method needed more 123s than the RNN solution to obtain its predictions. Moreover, the RNN solution does not imply that the dataset should be balanced, on the opposite of the SVM solution that as a classification method, can be influenced by the most populated classes. Therefore, it is possible to attest that the proposed solution in this dissertation was proved to be the most efficient one.

Chapter 6

Conclusion and Future Work

"When human judgment and big data intersect there are some funny things that happen."

Nate Silver

The presented work proposes a Predictive Maintenance Mechanism devised for heating, ventilation, and air-conditioning monitoring and optimization. By analyzing a dataset that contains boilers life cycle provided by Thermotechnology division of Bosch, it was possible to understand that data was given as multivariate time-series information. Also, some faults were already identified and documented which creates the need to identify them as outliers. Moreover, this identification requires that this problem should be interpreted as a classification and prediction one.

The main difficulty was to find an algorithm able to analyze time-series data that has missing data. Also, processing the data to better express the boilers cycles and models became an important step since this work deals with an unbalanced dataset. Therefore, to reduce its impact, a novel Model-Based Prediction Methodology which relies on the combination of a Markov Chain and a Recurrent Neural Network was implemented.

This implementation required a second normalization since not all of the studied boilers had the necessary information to retrieve one-month data, therefore, predicting one-week in advance. This normalization resulted in an 8.80% dataset reduction, which made possible the implementation of the Prediction Methodology.

After creating a graph that represents one month of information, it was possible to send it as an input of a Recurrent Neural Network. As this solution works with long dependencies, an LSTM implementation was required which required data preparation. Also, as data was organized as multivariate, the dataset was framed as a supervised problem and variables were normalized. Finally, the RNN outputs a combination of a severity range and a failure cause within five possibilities.

To verify the effectiveness of the application some aspects were evaluated such as the required time to process data and the results of the predictive methods. Moreover, an analysis

regarding the solution requirements was implemented. The experimentation has shown that the system shall have adjustments regarding the data processing and the models precision. First, time processing increases based on data expansion, second the anomaly classification is below 50%. However, the low accuracy can be related to the fact that the models are developed using an unbalanced dataset. Regarding the time processing, an increase of 100s with the data duplication is supportable.

Therefore, to reduce this impact a Model-Based method was implemented by considering the sequencing process of the boilers and creating a Markov Chain based on it. With a prediction hamming loss of 0.0952, we can attest that the proposed solution has a positive result. Moreover, an evaluation regarding the error information was performed and it was demonstrated that a higher number of neurons results in a lower error value.

Although the classification results were not perfect, an impact evaluation was conducted to understand how the results can respond to the problem that was meant to be solved and guarantee some of the project' objectives. This evaluation has demonstrated that user comfort has risen since the solution is able to predict failures with more than 70% precision. Besides, the necessary time to process data was also considered and an evaluation of the solution speed up was executed, being able to prove that a sustainable speedup is a guarantee.

Besides, to verify if the implemented solution was the most effective compared to other possible solutions, a comparison between the most typical Machine Learning algorithms used on PdM systems was made. After being compared with two methods, Support Vector Machines and Classification and Regression Trees, the presented solution has been demonstrated as the most adequate one.

Considering the previous results, it is possible to attest that most of the solution requirements were implemented and a viable Predictive Maintenance Mechanism was implemented.

Moreover, this work resulted in two publications in international conferences and an important contribution related to the dataset normalization.

6.1 Future Work

Regarding Future Work that would be possible and useful, this application would be improved by balancing the dataset before generating the classification models and readjust the data normalization techniques. Moreover, this balancing could improve the predictive technique resulting in a better failure prediction. It would also be prudent to readjust the task scheduler to decrease the time spent on the process of all data, resulting in a more efficient data processing.

Besides, online learning could be performed to maintain the predictive models always updated, and consequently, have the models adapted to the actual operating state of the boilers, not only on historical data. And, finally, a visualization of the prediction results would be prudent since the users could receive notifications when failures are predicted to happen.

References

[1] P. Lade, R. Ghosh, and S. Srinivasan, "Manufacturing analytics and industrial internet of things", *IEEE Intelligent Systems*, vol. 32, no. 3, pp. 74–79, 2017.

[2] R. Satta, S. Cavallari, E. Pomponi, D. Grasselli, D. Picheo, and C. Annis, "A dissimilarity-based approach to predictive maintenance with application to hvac systems", *arXiv preprint arXiv:1701.03633*, 2017.

[3] J. Ribeiro, M. Antunes, D. Gomes, and R. L. Aguiar, "Outlier Identification in Multivariate Time Series: Boilers Case Study", in *Proceedings of International Conference on Time Series and Forecasting (ITISE)*, 2018.

[4] C. R. de Sá, A. K. Shekar, H. Ferreira, and C. Soares, "Building Robust Prediction Models for Defective Sensor Data Using Artificial Neural Networks", *Advances in Intelligent Systems and Computing*, vol. 950, pp. 142–153, 2020, ISSN: 21945357. DOI: 10.1007/978-3-030-20055-8_14.

[5] J. Lee, J. Ni, D. Djurdjanovic, H. Qiu, and H. Liao, "Intelligent prognostics tools and e-maintenance. Computers in Industry", *Computers in Industry*, vol. 57, no. 6, pp. 476–489, 2006.

[6] C. Okoh, R. Roy, J. Mehnen, and L. Redding, "Overview of remaining useful life prediction techniques in through-life engineering services", *Procedia CIRP*, vol. 16, pp. 158–163, 2014.

[7] C. Groba, S. Cech, F. Rosenthal, and A. Gossling, "Architecture of a predictive maintenance framework", in *6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM'07)*, IEEE, 2007, pp. 59–64.

[8] C. M. Tan and N. Raghavan, "A framework to practical predictive maintenance modeling for multi-state systems", *Reliability Engineering & System Safety*, vol. 93, no. 8, pp. 1138–1150, 2008.

[9] R. Dhall and V. Solanki, "An IoT Based Predictive Connected Car Maintenance.", *International Journal of Interactive Multimedia & Artificial Intelligence*, vol. 4, no. 3, 2017.

[10] 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)*, IEEE, 2017, pp. 70–77.

[11] J. Lindström, H. Larsson, M. Jonsson, and E. Lejon, "Towards intelligent and sustainable production: Combining and integrating online predictive maintenance and continuous quality control", *Procedia CIRP*, vol. 63, pp. 443–448, 2017.

[12] J. Pan, R. Jain, S. Paul, T. Vu, A. Saifullah, and M. Sha, "An internet of things framework for smart energy in buildings: Designs, prototype, and experiments", *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 527–537, 2015.

[13] S. Suursalu, "Predictive maintenance using machine learning methods in petrochemical refineries", 2017.

[14] D. Chelidze, "Multimode damage tracking and failure prognosis in electromechanical systems", in *Component and Systems Diagnostics, Prognostics, and Health Management II*, International Society for Optics and Photonics, vol. 4733, 2002, pp. 1–13.

[15] J. B. Ali, B. Chebel-Morello, L. Saidi, S. Malinowski, and F. Fnaiech, "Accurate bearing remaining useful life prediction based on weibull distribution and artificial neural network", *Mechanical Systems and Signal Processing*, vol. 56, pp. 150–172, 2017.

[16] S. Kang, E. Kim, J. Shim, W. Chang, and S. Cho, "Product failure prediction with missing data", in *International Journal of Production Research*, vol. 56, 2018, pp. 4849–4859. DOI: 10.1080/00207543.2017.1407883.

[17] M.-A. Zöller, M. Baum, and M. F. Huber, "Framework for mining event correlations and time lags in large event sequences", in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, IEEE, 2017, pp. 805–810.

[18] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang, "Automated it system failure prediction: A deep learning approach", in *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, 2016, pp. 1291–1300.

[19] R. Sipos, D. Fradkin, F. Moerchen, and Z. Wang, "Log-based predictive maintenance", in *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, ACM, 2014, pp. 1867–1876.

[20] V. Gurusamy, S. Kannan, and K. Nandhini, "The real time big data processing framework: Advantages and limitations", *vol*, vol. 5, pp. 305–312, 2017.

[21] M. Antunes, J. P. Barraca, D. Gomes, P. Oliveira, and R. L. Aguiar, "Smart cloud of things: an evolved iot platform for telco providers", *Journal of Ambient Wireless Communications and Smart Environments (AMBIENTCOM)*, vol. 1, no. 1, pp. 1–24, 2015.

[22] F. Chen, P. Deng, J. Wan, D. Zhang, A. V. Vasilakos, and X. Rong, "Data mining for the internet of things: literature review and challenges", *International Journal of Distributed Sensor Networks*, vol. 11, no. 8, p. 431 047, 2015.

[23] A. Bey-Temsamani, M. Engels, A. Motten, S. Vandenplas, and A. P. Ompusunggu, "A practical approach to combine data mining and prognostics for improved predictive maintenance", *Data Min. Case Stud*, vol. 36, 2009.

[24] A. K. Choudhary, J. A. Harding, and M. K. Tiwari, "Data mining in manufacturing: a review based on the kind of knowledge", *Journal of Intelligent Manufacturing*, vol. 20, no. 5, p. 501, 2009.

[25] C.-W. Tsai, C.-F. Lai, M.-C. Chiang, and L. T. Yang, "Data mining for internet of things: A survey", *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 77–97, 2014.

[26] G. Chandrashekar and F. Sahin, "A survey on feature selection methods", *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[27] A. K. Shekar, T. Bocklisch, P. I. Sánchez, C. N. Straehle, and E. Müller, "Including multi-feature interactions and redundancy for feature ranking in mixed datasets", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2017, pp. 239–255.

[28] R. Adhikari, "A neural network based linear ensemble framework for time series forecasting", *Neurocomputing*, vol. 157, pp. 231–242, 2015.

[29] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah, "A review on time series forecasting techniques for building energy consumption", *Renewable and Sustainable Energy Reviews*, vol. 74, pp. 902–924, 2017.

[30] C. Yang, W. Shen, Q. Chen, and B. Gunay, "A practical solution for HVAC prognostics: Failure mode and effects analysis in building maintenance", *Journal of Building Engineering*, vol. 15, pp. 26–32, 2018.

[31] M. R. Brownie, G. A. Romanowich, R. K. Alexander, and N. J. Vandermause, *Hvac system with equipment failure prediction*, US Patent App. 15/483,667, Oct. 2017.

[32] C. Yang, Q. Chen, W. Shen, and B. Gunay, "Toward failure mode and effect analysis for heating, ventilation and air-conditioning", in *2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, IEEE, 2017, pp. 408–413.

[33] H. Yoshida, A. Kawaguchi, F. Yamashita, and K. Tsuruya, "The utility of a network–based clustering method for dimension reduction of imaging and non-imaging biomarkers predictive of Alzheimer's disease", *Scientific reports*, vol. 8, no. 1, p. 2807, 2018.

[34] 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.

[35] S. Duffuaa, M. Ben-Daya, K. Al-Sultan, and A. Andijani, "A generic conceptual simulation model for maintenance systems", *Journal of Quality in Maintenance Engineering*, vol. 7, no. 3, pp. 207–219, 2001.

[36] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey", *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[37] P. Ramos, J. M. S. Oliveira, and P. Silva, "Predictive maintenance of production equipment based on neural network autoregression and ARIMA", in *21st International EurOMA Conference-Operations Management in an Innovation Economy*, 2014.

[38] S. S. Haykin, S. S. Haykin, S. S. Haykin, K. Elektroingenieur, and S. S. Haykin, *Neural networks and learning machines*. Pearson education Upper Saddle River, 2009, vol. 3.

[39] J. Heinermann and O. Kramer, "Machine learning ensembles for wind power prediction", *Renewable Energy*, vol. 89, pp. 671–679, 2016.

[40] D. Warde-Farley, I. J. Goodfellow, A. Courville, and Y. Bengio, "An empirical analysis of dropout in piecewise linear networks", 2013.

[41] C. Yang, Q. Chen, Y. Yang, and N. Jiang, "Developing predictive models for time to failure estimation", in *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, IEEE, 2016, pp. 133–138.

[42] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning", 2017.

[43] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey", *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[44] G. Chen, R. Xu, and S. N. Srihari, "Sequential labeling with online deep learning: Exploring model initialization", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2016, pp. 772–788.

[45] D. Lachut, N. Banerjee, and S. Rollins, "Predictability of energy use in homes", in *International Green Computing Conference*, IEEE, 2014, pp. 1–10.

[46] A. González-Vidal, A. P. Ramallo-González, and A. Skarmeta, "Empirical study of massive set-point behavioral data: Towards a cloud-based artificial intelligence that democratizes thermostats", in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, IEEE, 2018, pp. 211–218.

[47] J. Zhuang, Y. Chen, X. Shi, and D. Wei, "Building cooling load prediction based on time series method and neural networks", *International Journal of Grid and Distributed Computing*, vol. 8, no. 4, pp. 1386–1390, 2015.

[48] H. R. Maier and G. C. Dandy, "Neural networks for the prediction and forecasting of water resources variables: A review of modelling issues and applications", *Environmental modelling & software*, vol. 15, no. 1, pp. 101–124, 2000.

[49] A. M. Bahaa-Eldin, H. K. Mohamead, and S. S. Deraz, "Increasing Server Availability for Overall System Security: A Preventive Maintenance Approach Based on Failure Prediction", *arXiv preprint arXiv:1401.5686*, 2014.

[50] N. Kourentzes, D. K. Barrow, and S. F. Crone, "Neural network ensemble operators for time series forecasting", *Expert Systems with Applications*, vol. 41, no. 9, pp. 4235–4244, 2014.

[51] P. Wang and G. Vachtsevanos, "Fault prognostics using dynamic wavelet neural networks", vol. 15, no. 4, pp. 349–365, 2001.

[52] K. Wang, Y. Zhao, Q. Xiong, M. Fan, G. Sun, L. Ma, and T. Liu, "Research on healthy anomaly detection model based on deep learning from multiple time-series physiological signals", *Scientific Programming*, vol. 2016, 2016.

[53] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation", 2014.

[54] J. Keuper and F.-J. Preundt, "Distributed training of deep neural networks: Theoretical and practical limits of parallel scalability", in *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments*, IEEE Press, 2016, pp. 19–26.

[55] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble deep learning for regression and time series forecasting", in *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, IEEE, 2014, pp. 1–6.

[56] M. Motamedi, D. Fong, and S. Ghiasi, "Fast and energy-efficient cnn inference on iot devices", 2016.

[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[58] J. Wang and L. Perez, "The effectiveness of data augmentation in image classification using deep learning", *Convolutional Neural Networks Vis. Recognit*, 2017.

[59] A. Bergmann, *Data mining for manufacturing: Preventive maintenance, failure prediction, quality control*, 2012.

[60] M. Boer, M. Friedrich, M. Krämer, P. Noack, J. N. Weiss, and A. Zimmermann, "Towards resilient enterprise architecture for predictive maintenance", in *Innovation in Medicine and Healthcare Systems, and Multimedia*, Springer, 2019, pp. 381–391.

[61] A. Rivas, J. M. Fraile, P. Chamoso, A. González-Briones, I. Sittón, and J. M. Corchado, "A predictive maintenance model using recurrent neural networks", in *International Workshop on Soft Computing Models in Industrial and Environmental Applications*, Springer, 2019, pp. 261–270.

[62] M. A. der Mauer, T. Behrens, M. Derakhshanmanesh, C. Hansen, and S. Muderack, "Applying sound-based analysis at porsche production: Towards predictive maintenance of production machines using deep learning and internet-of-things technology", in *Digitalization Cases*, Springer, 2019, pp. 79–97.

[63] T. Abbasi, K. H. Lim, and K. San Yam, "Predictive maintenance of oil and gas equipment using recurrent neural network", in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 495, 2019, p. 012 067.

[64] M. Fernandes, A. Canito, J. M. Corchado, and G. Marreiros, "Fault detection mechanism of a predictive maintenance system based on autoregressive integrated moving average models", in *International Symposium on Distributed Computing and Artificial Intelligence*, Springer, 2019, pp. 171–180.

[65] D. E. Johnson, S. Gray, and K. Uppuluri, *Maintenance optimization system through predictive analysis and usage intensity*, US Patent App. 15/810,223, May 2019.