SAPIENZA
UNIVERSITÀ DI ROMA

Artificial Intelligence for Digital Twins in Energy Systems and Turbomachinery: development of machine learning frameworks for design, optimization and maintenance

Department of Astronautical, Electrical and Energy Engineering
Doctor of Philosophy Degree in Energy and Environment (XXXV cycle)

PhD Candidate: Francesco Aldo Tucci

Advisor
Prof. Alessandro Corsini, Ph.D.

Co-Advisor
Prof. Giovanni Delibra, Ph.D.

Academic Year 2021/2022

Artificial Intelligence for Digital Twins in Energy Systems and Turbomachinery: development of machine learning frameworks for design, optimization and maintenance

PhD thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: francescoaldo.tucci@uniroma1.it

Dedicated to
my Family and Beatrice

Acknowledgments

After three years, my PhD is coming to the end. It has been a pivotal experience both in terms of professional and personal growth. Now it is time to thank the people who have supported and inspired me along this journey, marked by both joys and successes as well as doubts and fears and even a pandemic.

First and foremost, I would like to thank my supervisor, Prof. Alessandro Corsini, for guiding and trusting me throughout my research period. His valuable advice, constant presence and invaluable knowledge of energy systems, turbomachinery, CFD and Artificial Intelligence instilled in me the desire to work and study hard for being able to support the boldest ideas with belief, passion and courage. It was an honor to be his student.

I would also like to thank Prof. Giovanni Delibra, who has followed me on a daily basis since my master's thesis and whom I now consider as an older brother. He has taught me much more than programming algorithms, running CFD simulations and analyzing the results. His support really made all this possible and he will always have my sincerest gratitude.

Endless thanks also go to my now close friend, Dr. Lorenzo Tieghi, for everything he taught me about Data Analysis and Machine Learning, for his support in debugging every single piece of code I wrote, for his insightful comments, and for his daily encouragement and help. And especially for the everyday picturesque desktop wallpaper updates on my laptop.

Special thanks go to another now close friend, Dr. Valerio Barnabei, with whom, although I rarely collaborated, I shared important moments and experiences and he never let me lack his technical and moral support whenever I needed it.

I want to thank my workmates: Alessio, Davide, Eric, Erfan, Fabrizio, Isabella, Paolo and Rossana. In addition to working together, we shared pleasant and funny moments.

I also thank my family, my Mother, Father, Brother and Sister, who have always supported me; Beatrice with whom I am sharing important moments. To them I dedicate this important goal.

Last but not least, my deepest gratitude to Andrea and Marco. Without your support this journey would have been definitely more difficult and less fun.

Abstract

The expression Industry4.0 identifies a new industrial paradigm that includes the development of Cyber-Physical Systems (CPS) and Digital Twins promoting the use of Big-Data, Internet of Things (IoT) and Artificial Intelligence (AI) tools. Digital Twins aims to build a dynamic environment in which, with the help of vertical, horizontal and end-to-end integration among industrial processes, smart technologies can communicate and exchange data to analyze and solve production problems, increase productivity and provide cost, time and energy savings. Specifically in the energy systems field, the introduction of AI technologies can lead to significant improvements in both machine design and optimization and maintenance procedures. Over the past decade, data from engineering processes have grown in scale. In fact, the use of more technologically sophisticated sensors and the increase in available computing power have enabled both experimental measurements and high-resolution numerical simulations, making available an enormous amount of data on the performance of energy systems. Therefore, to build a Digital Twin model capable of exploring these unorganized data pools collected from massive and heterogeneous resources, new Artificial Intelligence and Machine Learning strategies need to be developed.

In light of the exponential growth in the use of smart technologies in manufacturing processes, this thesis aims at enhancing traditional approaches to the design, analysis, and optimization phases of turbomachinery and energy systems, which today are still predominantly based on empirical procedures or computationally intensive CFD-based optimizations. This improvement is made possible by the implementation of Digital Twins models, which, being based primarily on the use of Machine Learning that exploits performance Big-Data collected from energy systems, are acknowledged as crucial technologies to remain competitive in the dynamic energy production landscape. The introduction of Digital Twin models changes the overall structure of design and maintenance approaches and results in modern support tools that facilitate real-time informed decision making. In addition, the introduction of supervised learning algorithms facilitates the exploration of the design space by providing easy-to-run analytical models, which can also be used as cost functions in multi-objective optimization problems, avoiding the need for time-consuming numerical simulations or experimental campaings. Unsupervised learning methods can be applied, for example, to extract new insights from turbomachinery performance data and improve designers' understanding of blade-flow interaction. Alternatively, Artificial Intelligence frameworks can be developed for Condition-Based Maintenance, allowing the transition from preventive to predictive maintenance.

This thesis can be conceptually divided into two parts. The first reviews the state of the art of Cyber-Physical Systems and Digital Twins, highlighting the crucial role of Artificial Intelligence in supporting informed decision making during the design, optimization, and maintenance phases of energy systems. The second part covers the development of Machine Learning strategies to improve the classical approach to turbomachinery design and maintenance strategies for energy systems by exploiting data from numerical simulations, experimental campaigns, and sensor

datasets (SCADA). The different Machine Learning approaches adopted include clustering algorithms, regression algorithms and dimensionality reduction techniques: Autoencoder and Principal Component Analysis.

A first work shows the potential of unsupervised learning approaches (clustering algorithms) in exploring a Design of Experiment of 76 numerical simulations for turbomachinery design purposes. The second work takes advantage of a non-sequential experimental dataset, measured on a rotating turbine rig characterized by 48 blades divided into 7 sectors that share the same baseline rotor geometry but have different tip designs, to infer and dissect the causal relationship among different tip geometries and unsteady aero-thermodynamic performance via a novel Machine-Learning procedure based on dimensionality reduction techniques. The last application proposes a new anomaly detection framework for gensets in DH networks, based on SCADA data that exploits and compares the performance of regression algorithms such as XGBoost and Multi-layer Perceptron.

Contents

Chapter 1

Introduction

1.1 Background and motivations

Industry is experiencing the transformation of the digital age, and a new industrial paradigm is emerging. This new paradigm, through the introduction of smart technologies such as Big-Data, Internet of Things and Artificial Intelligence, involves rethinking industrial processes to create an environment in which these smart technologies can communicate and share data to analyze and understand industrial problems and solve them [1]. Deeply immersed in this context is the energy sector, which is now more than ever driven by the concept of increasing efficiency and reducing costs [2]. Efficiency gains are typically achieved through technological innovations, such as upgrading available technology with new and more efficient design solutions, design fine-tuning, or technology-specific optimization processes. Cost reductions, on the other hand, can be achieved without affecting the final product quality by reducing production time, reducing the possibility of failure, and adopting newly developed materials.

It is no secret that the design of energy systems, and in particular fluid-machinery components, is an expensive procedure in terms of both computational and experimental efforts because it involves several multidisciplinary activities [3]. In fact, in most of the cases the design relies on classical approaches based on empirical rules from previous studies, elements of Computational Fluid Dynamics for a detailed analysis of flow behavior and elements for laboratory tests. On operations, strategies are also still heavily anchored in classical preventive maintenance approaches and do not take advantage of new technologies to deploy predictive and condition-based maintenance frameworks [4]. However, in a world of increasingly complex and integrated problems, those approaches have proven insufficient to address the challenges of flexible and deeply customized manufacturing processes. But advances in simulation software, combined with high-performance computing and increased capacity of efficient data storage, open the way to possibilities for analysis of complex multiphysical systems that were not thought possible a decade ago [5].

In this scenario, state-of-the-art approaches for energy systems design, analysis, optimization, and maintenance have been revised through the implementation of new frameworks and strategies that are inspired by the so-called Industry4.0; this has led to:

- *Digitization of value chains*: the implementation of such networking results in flexible processes that can respond quickly to new design needs or changes in customer demand;
- *Digitization of product*: the introduction of smart data harvesting and analysis methods allows to create new design procedures and maintenance strategies using the large amount of collected information (Big-Data).
- *Acceleration through exponential technologies*: the introduction of Artificial Intelligence and its sub-fields (Machine Learning and Deep Learning) reduces costs, increases flexibility and customizes products.

Consequently, as data-driven models can quickly analyze and use data to make informed decisions in real time instead of relying on old labor-intensive empirical processes, their role in the design, production, and maintenance of energy systems will continue to grow. In particular, Machine Learning and Deep Learning tools define new programming paradigms whereby algorithms are trained rather than programmed: using huge datasets related to a specific task, users define the input data and the expected answers, while the algorithm finds the hidden structures within the examples and rules to automate the task. In other words, the inclusion of machine learning and deep learning in classical approaches changes the overall structure of the power generation process and provides modern tools for each intermediate design, operations monitoring, and maintenance stage.

The enormous success of Machine Learning is mainly due to the availability of a huge amount of data that can be used during the training phase of data-driven models; these data are collected both during the operation of machines and with experimental campaigns by means of smart sensors or through numerical simulations that take advantage of increased computational performance. As a consequence, Artificial Intelligence and its subfields have been recognized as a crucial technology to remain competitive in the modern ever-changing industrial landscape. Indeed, the availability of an ever-growing amount of detailed data allows these alternative digital tools to establish themselves as a technology that, if properly trained, can yield accurate results comparable to canonical CFD or experimental approaches, but in far less time and enhance the increase of energy systems performance while reducing costs.

To this end, the innovation provided by the Digital Twin model, and Cyber-Physical Systems in general, is extraordinary [5]: a Digital Twin, as will be described below, is able to help meet the energy industry's need for enhanced performance through constant optimization and cost reduction. The first finding will be obvious once the definition of Digital Twin is provided; while the second insight is not so well highlighted by the name Digital Twin: imagine how efficient and less costly it would be to equip a large, complex, and expensive machine with a digital counterpart that during each design, testing, optimization, and maintenance loop could be queried providing results that would facilitate real-time decision making or allow the study of the machine's behavior under certain rare conditions and thus target its development.

Closely related to the Digital Twin concept is the Digital Thread framework, which can be seen as a data-driven architecture that connects information generated from the entire product lifecycle and aims to become the primary or authoritative

data and communication platform [6]. The product development process is often fragmented into isolated teams and tools, which poses a significant risk of delays, defects, cost overruns, lack of verification and validation, etc. Reducing these risks requires end-to-end process control to detect anomalies early. Therefore, an unconstrained approach is needed that brings together the necessary metadata between different tools in a way that links the desired outcome to upstream and downstream activities. The digital thread is the best approach to reduce the risk of negative product outcomes while preserving engineering autonomy and productivity for products at all times.

Creating these tools, from a technical point of view, is far from simple, and several critical issues must be taken into account, especially in the case of fast machines involving complex physics, such as fluid machinery. In addition, different technologies, such as artificial intelligence, Big-Data analytics, Machine Learning, and cloud computing, which in most cases are themselves under development, are combined to realize the digital twin; and the infrastructure to implement the digital twin also needs to be improved to increase the effectiveness of this technology. Therefore, it is necessary to push forward research on smart technologies to implement the digital twin. Eventually, the possibility of leveraging vast data sets on which to train algorithms leads Artificial Intelligence and Machine Learning in particular to play a key role in the development of highly accurate Digital Twins [7].

1.2 Thesis outline

The following dissertation will be structured in chapters, as follows:

- In Chapter *Cyber Physical Systems, Digital Twins and Digital Thread for Energy Systems*: the concepts of Cyber-Physical Systems (CPS), Digital Twins and Digital Threads are presented. In particular, starting with the definitions, the possible applications of these technologies in the field of turbomachinery and energy systems are examined, with a particular focus on their current limitations and possible ways to overcome them;

- In Chapter *The role of Artificial Intelligence in Digital Twins*: the role that Artificial Intelligence, Machine Learning, Big Data and the Internet of Things play in the development of successful Digital Twins is described. A Digital Twin relies on bidirectional transfer and sharing of data collected during the various stages of an industrial process, leading to a holistic understanding of the same;

- In Chapter *Machine Learning Tools and Techniques*: a comprehensive description of the branches of Machine Learning is provided. The chapter discusses in detail all the supervised and unsupervised Machine Learning algorithms exploited in turbomachinery and energy systems applications;

- In Chapter *Development of Machine-Learning assisted tools and framework for CPS*: three different applications of Artificial Intelligence-assisted tools for the development of Digital Twin models in both energy systems and turbomachinery are discussed. The three works will address the design of

turbomachinery using Big Data from numerical simulations or experimental campaigns and the topic of Condition Based Maintenance of energy systems via Machine Learning frameworks;

- In Chapter *Conclusions*: a summary of the main outcomes of the dissertation are provided.

All the developed and implemented codes, the algorithms for machine learning are written in Python language, using scikit-learn [8] and Tensorflow [9] libraries. The numerical simulations needed are implemented in OpenFOAM [10] finite volume solver written in C++. Experimental data sets are provided through collaborations with international research institutes and companies such as the Von Karman Insitute for Fluid Dynamics and ENGIE.

Chapter 2

Cyber-Physical Systems, Digital Twins and Digital Thread for Energy Systems

The purpose of this chapter is, on the one hand, to briefly introduce the concepts of Cyber-Physical Systems, Digital Twins, and Digital Thread; on the other hand, to provide an extensive analysis of the improvements that these new technologies can bring to the energy industry. Today, these emerging technologies prove crucial in the design, operation, and management phase of energy systems and the machines that operate in them. In fact, they can harness the enormous amount of data collected from sensors in increasingly complex energy systems and develop tools specifically designed for predictive maintenance, design exploration and performance optimization.

2.1 Concept of Cyber-Physical Systems

Cyber-Physical Systems (CPS) are a set of modern interconnected systems that embed combined computing, communication and control capabilities into physical devices in order to monitor, control and coordinate their activities [11]. The typical structure of a CPS is illustrated in Fig. 2.1. The main component of a CPS is the communication infrastructure through which cyber systems and physical systems can be connected to exchange data. The physical unit includes various physical devices such as sensors, actuators and hardware that interact with the cybernetic system, which analyzes and processes data received from the physical systems and produces an output (recommended action) in response [12].

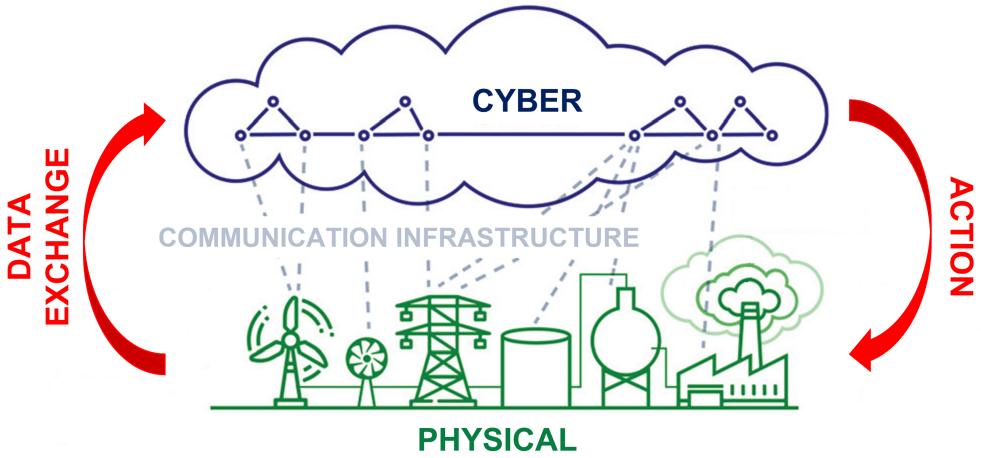


Figure 2.1. Schematic diagram of a cyber-physical system.

At present, research in the field of CPS is aimed at increasingly detailed and complex modeling of both the concept and applications, and also covers a wide variety of scientific research areas, such as bio-engineering [13], automotive [14], transportation systems [15], and eventually energy network management [16].

While there is no universal definition of CPS, the acronym CPS was first coined in 2006 [17] and refers to a wide range of next-generation multidisciplinary engineering systems that incorporate computing technologies to build goal-oriented physical devices [18].

According to Rajkumar et al. [19], who define CPS as: *"physical and engineering systems whose operations are monitored, coordinated, controlled and integrated by a computational and communication core"*, the growing interest in the design and adoption of CPS in many sectors can be traced to a combination of multiple factors that can be divided into two main categories:

- *Pulling factors*: a wide adoption of CPS in most of their application fields is still quite unfeasible since the technological foundation needed to develop them are seriously lacking;

- *Pushing factors*: rapid deployment of low-cost, high-capacity sensors of increasingly smaller size; growing availability of more powerful and smaller computing devices; widespread accessibility of wireless and broadband connections; increasing data storage and management capacity.

Therefore, the crucial aspect of CPSs is the coexistence of computing capabilities and physical processes, which is based on some fundamental subsystems that must stand together to give rise to a CPS model. In fact, a CPS model should always include the following elements [20]:

- *Hybrid system*: because of the coupling between the cyber and physical domains, through mathematical models it is possible to manage both continuous and discrete domains and to match continuous signals from physical devices with the discrete domain of computational systems;

- *Embedded system*: relies on the computational system closely integrated into a physical system;
- *Multiagent system*: the cyber part of the CPS consists of several interactive computational objects, each of which performs a specific task aimed at achieving a specific goal;
- *Real Time system*: should operate with response times comparable to the time-scales of the physical phenomena underlying the real device to which they are coupled;
- *Reliable system*: designed to work continuously despite failure or malfunction of any of its components.

Despite numerous research efforts to design and implement CPS in many industrial sectors, this technology still has to face several challenges. Using innovative Machine Learning-based strategies, researchers are trying to solve security problems of CPSs in terms of external attacks and disruptive failures. They are also trying to find new control strategies for these hybrid systems and new data storage and management solutions, given the huge amount of data obtainable from sensors.

2.2 Digital Twin definition

A digital twin (DT) is the virtual replica of a physical entity (physical twin), both of which are mutually linked by real-time data exchanges that enable a more realistic and holistic evaluation of unexpected and unpredictable scenarios [21]. Ideally, a DT will mirror the state of its physical twin in real time (and vice versa) for the entire life cycle of the product [22]. Its concept was first introduced by Grieves [23, 24], as a product life-cycle management model called the "Mirrored Spaces Model", which entails three different parts: real space and virtual space, put in communication through a network link for the exchange of data and information. As can be seen in Fig. 2.2, the proposed model shows that the connection between virtual and real space is bidirectional, and also that there is the possibility of multiple parallel virtual spaces for a single real space where alternative ideas or projects can be explored.

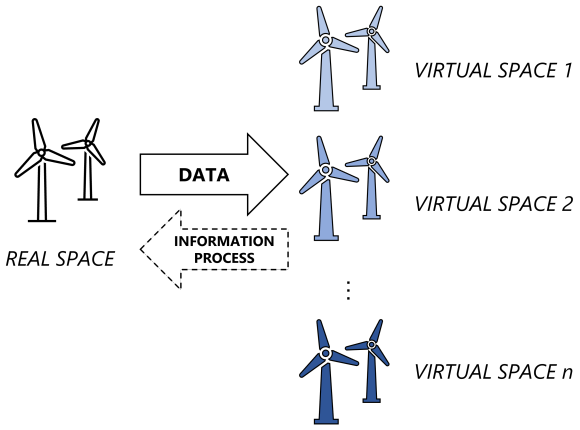


Figure 2.2. Schematic of a Mirrored Spaces Model (as proposed by Grieves [23, 24]).

In 2006, the name of the model proposed by Grieves was updated to "Information Mirroring Model" [25], while in 2010 the term Digital Twin (DT) was first introduced by NASA and defined as: *"an integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its physical twin"* [26]. Fig. 2.3 shows the time-line evolution of the concept of DT.

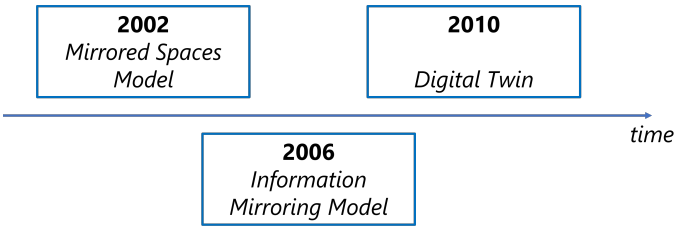


Figure 2.3. Timeline evolution of DT concept.

Besides the various definitions, DTs can be classified into different types on the basis of different criteria. For example, according to Vickers [27], there are two types of DTs based on the timing of their development during the product life-cycle:

- *Digital Twin Prototype (DTP)*: it is related to the design phase, before the prototype is created. It is a DT that contains the complete and final set of data and information to develop its physical copy. In fact, the DTP undergoes multiple tests before its physical twin is created; thus, it helps in the identification and avoidance of unpredictable and unexpected situations that are difficult to detect with traditional prototyping. Once the testing of the DTP is completed and validated, its physical twin can be produced in the real world. The greater the accuracy of the virtual model simulations, the higher the quality of the physical twin;

- *Digital Twin Instance (DTI)*: it is associated with the production/operation phases after the product is ready. In this case, data from the real space are sent to the virtual space and vice versa to monitor and predict the behavior of the system; this allows action to be taken to correct system behavior if undesirable scenarios occur. Since the link between the two systems is bidirectional, any changes made to one will be reflected in the other.

In addition to the previous characterization, DTs can also be classified according to their scopes [27]. A *Predictive DT* is useful for studying in advance the future behaviors and performance of its physical twin, while an *Interrogative DT* is used to interrogate the current or past state instead of its physical counterpart. Finally, a further differentiation of DTs can be made with respect to the focus of their applications [28]:

- *Product DT*: aimed at an efficient design of new products, provides a virtual-physical tool for analyzing the performance of a product under various conditions and making virtual changes to ensure that the developed physical product performs exactly as expected;

- *Production DT*: with the aim of improving production planning, by simulating the process with DTs, companies can create production methodologies that remain efficient under different conditions. Furthermore, by using data from product and production, companies can prevent costly downtime and even predict when maintenance is needed;
- *Performance DT*: which aims at capturing data from products in operation and analyzing them to provide useful insights for informed real-time decision making.

To conclude the overview of DTs, from a hierarchical point of view, DTs can be divided into three different manufacturing levels [29]:

- *Unit level*: relies on the geometrical, functional, and operational model of physical twins at unit level, such as material, single component and equipment;
- *System level*: linking different unit-level DTs, such as the production line, shop floor, and factory, for wider data flow and enhanced resource allocation;
- *System of systems level*: connection of different system-level DTs that integrate different phases of the product during its life-cycle, such as supply chain, design, service, maintenance, etc.

With respect to the CPSs first introduced in the previous paragraph, it is possible to conclude that a DT is a subset of CPSs; they share many characteristics, such as being driven by a computational unit that, in a closed loop, analyzes sensor data on the current and past state of the physical twin and produces outputs for real time decision making [30]. Fig. 2.4 presents an example layout for the DT architecture embedded in a more complex CPS, from which the 3 main components can be distinguished: the physical sphere, the cyber sphere connected by a bidirectional IoT network.

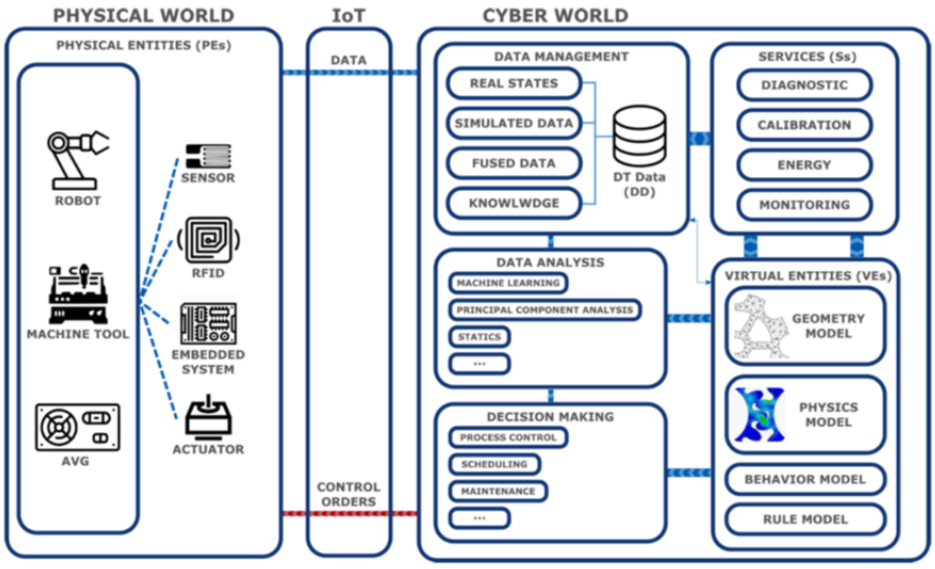


Figure 2.4. Example of a DT architecture.

2.3 Digital Thread definition

A concept that is generally coupled with that of Digital Twin is the Digital Thread. If the Digital Twin is a virtual state that represents its physical twin in real time, the digital thread relies on the storage and diffusion of all available information about the physical twin during its life-cycle [31]. In fact, according to Willcox et al. [6], the Digital Thread can be formally defined as a data-driven framework that connects data and information generated from the entire product life-cycle to make real-time and long-term decision making. Therefore, it is important not to confuse the related concepts of Digital Twin and Digital Thread: on the one hand, the Digital Twin can be intended as a high-fidelity combination of models and computational tools that simulate the performance of a product; on the other hand, the Digital Thread represents the pipeline of data and information necessary for the generation and updates of the Digital Twin [32].

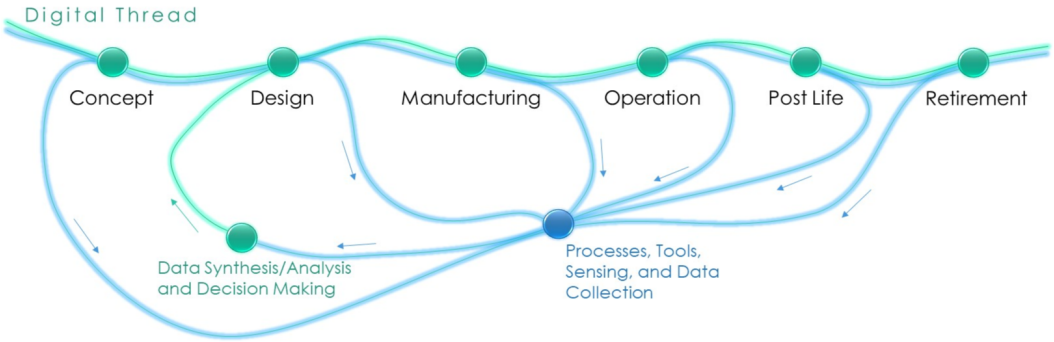


Figure 2.5. Illustration of engineering design with Digital Thread [6].

In Fig. 2.5 is illustrated a schematic example of a Digital Thread. Since multiple phases of the product life-cycle feed information into its architecture, it delivers information about the status of the product development process: performance, visible risks, and corrective actions to be considered. In this way, the information gains more value because it can be used effectively for informed choices about future designs, to reduce uncertainty in design parameters and process costs, and can also reveal more efficient strategies for operation and maintenance. Hence, unlike a static database, the key value of the Digital Thread is to enable, at any given time, an up-to-date overview of the status of product development processes even across multiple and isolated teams. Without a Digital Thread, a company would be flying blind in terms of the risks it takes in product development [6].

2.4 Digital twins in the product life-cycle perspective of turbomachinery and energy industry

Because of its multiple benefits (reduction of errors, uncertainties, inefficiencies, and expenses in any system or process), DT technology is recognized as the cornerstone of technological progress enabled by Industry 4.0. It is therefore easy to see that DT models, and all the technologies required to their implementation, are perfectly

suited to the energy sector. Looking at the entire life cycle of a product related to the energy and turbomachinery industry (a turbine, an aircraft engine, a specific component of a power generation plant, or a power grid system), it is possible to recognize the cruciality of developing a DT model that represents and supports each stage of the product life cycle, from the design phase to the disposal and recycling phase. In addition, this technology enables both horizontal design, developing a specific DT model for each stage of a product's life cycle, and vertical design, ranging from the single component, to the product, to the system of systems. After these considerations, it is possible to list the possible advantages of gradually introducing the use of DT in research and development, quality control, and predictive maintenance of the power and turbomachinery industries. Specifically, according to Xie et al [33], the life cycle of a turbomachinery product is cyclical and entails five consecutive phases (Fig. 2.6):

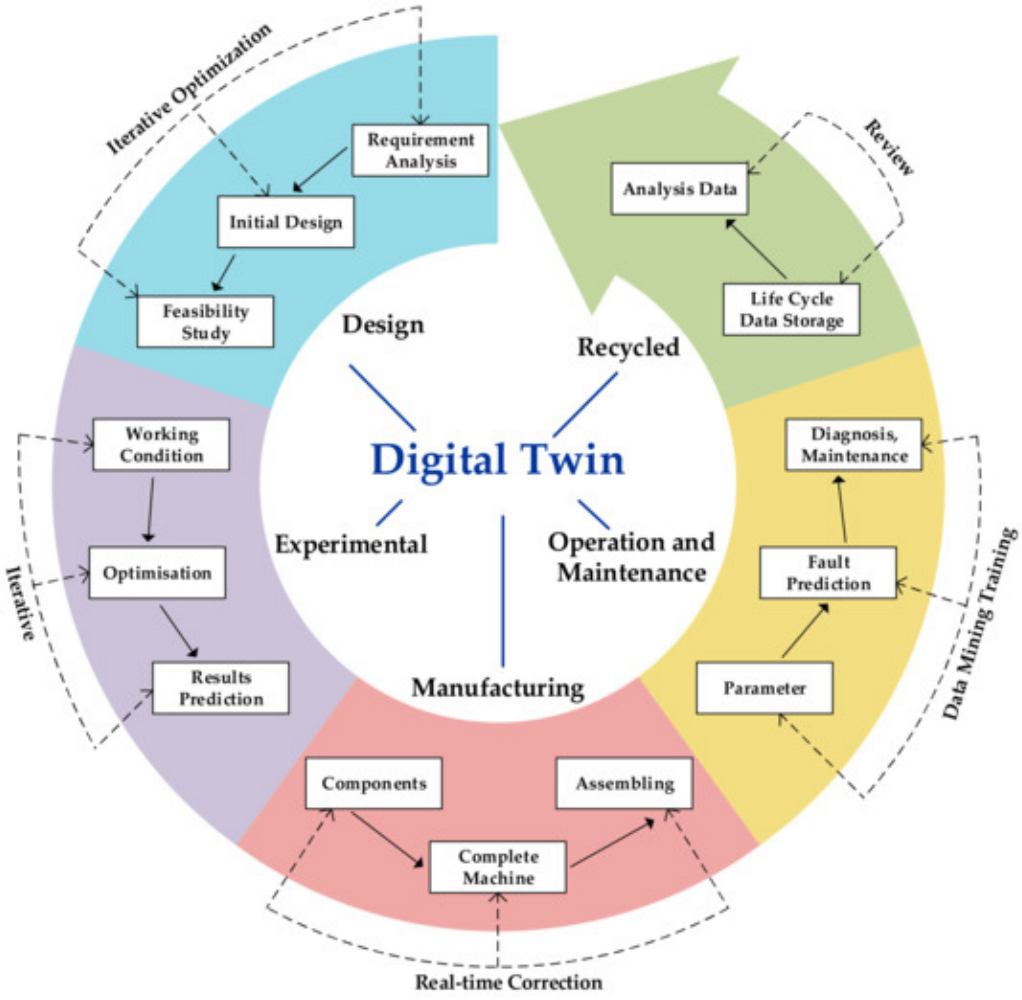


Figure 2.6. Turbomachinery product life-cycle representation with Digital Twin [33].

- *Design phase*: at this stage, the DT can be developed either from a previous product, from which data can be analyzed to make improvements or even

start a new design, or without a physical counterpart available. However, at the design stage, the DT allows designers to drastically reduce the time needed to explore a large number of different design solutions, since feedback is immediately available by evaluating the performance of virtual products. For example, Baldassarre et al. [34] by means of DT technology redesigned an existing wind turbine;

- *Experimental phase*: this phase, which is crucial for testing the prototype in terms of safety and performance reliability, is characterized by high cost and time. In fact, setting up an experimental campaign requires a huge amount of very high-quality hardware and also long run times to ensure that the data collected are as meaningful and complete as possible. At this stage, the use of DT would minimize the time needed to acquire meaningful data. In addition, virtual tests could provide higher definition data through the use of complex models. Moreover, since such tests are not bound to sensors, they can provide data even where sensors cannot be placed due to technological or operational constraints;

- *Manufacturing phase*: during the production phase, a DT assisted by a deeply interconnected network of Internet of Things (IoT)-enabled sensors could improve both the product manufacturing and assembly process and the plant layout [35];

- *Operation & maintenance phase*: at this stage, the implementation of DT in the energy and turbomachinery industry leads to a shift from reactive to proactive maintenance. As a result, extending the lifespan of the product and its components results in a drastic reduction in costs due to maintenance and failures. In fact, the DT constantly collects and processes data from the physical device and provides fault information or triggers alarms for occurring failures. In this way, it helps to prevent and reduce disruptive and costly failures with enough anticipation. For example, Seshadri et al. [36] developed an integrated Structural Health Management tool based on DT for an accurate detection and prognosis of damaged aircraft under normal and adverse conditions during flight;

- *Disposal phase*: the Digital Twin of a product at the end of its life-cycle can provide useful information for the development of a similar product. In fact, being a kind of interactive library of the operation of a previous device, it can significantly speed up the process of designing a new product of the same type and improve its performance, while reducing the cost of the design phase. It is easy to assume that for research purposes, in terms of costs and space, it is much easier to exploit a digital twin than to keep its old physical twin operational.

Chapter 3

The role of Artificial Intelligence in Digital Twins

The underlying element of the DT concept, besides the virtual representation of a real object, is the bidirectional transfer and sharing of data between the two counterparts. This bidirectional transfer of data, involving quantitative, qualitative, historical, environmental, and real-time data, enables understanding of how decisions and actions taken upstream in a process affect things downstream and also facilitates activities such as: designing, optimizing, and validating new products or processes; simulating and monitoring the state of the physical twin and increasing its performance and reliability; predicting the performance of the physical twin; and controlling the physical twin in real time. Hence, in this view, it is evident that a successful DT builds on and continues to evolve with digitization technologies such as Artificial Intelligence (AI), Machine Learning (ML), Big Data, Internet of Things (IoT), and numerical simulations.

3.1 Relationship between digitization technologies and Digital Twin

In recent years, the rise in the use of more technologically advanced sensors and the inclusion of AI implementations in industrial environments have encouraged the development of several interesting applications, above all the real-time monitoring of physical devices [37] and real-time data collection [38], which are crucial for the realization of a DT that accurately mirrors all the characteristics of its physical twin [39]. In fact, proper analysis of data collected through next-generation sensors and IoT deployments, via AI and ML tools that bidirectionally link the physical environment to its virtual image, plays a key role in developing a DT that can optimize design [40] and maintenance [41] strategies for the corresponding physical component. The reason lies in the fact that identifying potential problems in industrial processes, for example, especially when related to turbomachinery and energy systems that have to account for multi-physics and turbulent phenomena, is a very complex and complicated challenge if conducted with traditional techniques. On the other hand, such problems can be more easily, accurately and quickly extrapolated from the collected data by exploiting intelligent industrial processes

that require advanced AI frameworks and ML tools to handle the huge amount of collected data and build an efficient DT. Therefore, an intelligent DT system is based on the application of advanced AI-ML techniques to the collected data. For that purpose, a smart industrial process is realized when a DT system can identify the best process strategy and resource allocation [42], predict failures and schedule early maintenance [41], optimize process planning and control [43], and, eventually, make dynamic decisions based on physical sensor data or data generated by virtual twins. Figure 3.1 presents a schematic view of the overall relationships between digitization techniques and DTs.

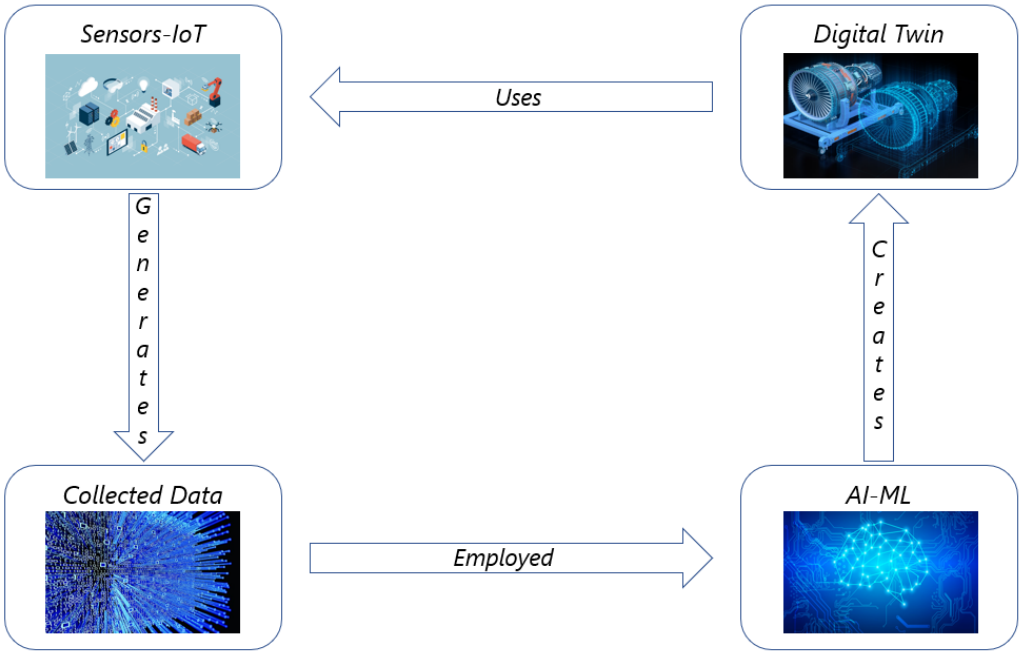


Figure 3.1. Schematic view of the relationships between digitization techniques and DTs.

To effectively explain the benefits of integrating data analysis and AI-ML models into digital twinning, a reference architecture is here introduced: the process begins with data collection via sensors (physical environment) or numerical simulations (virtual environment). The data are sent to the data analysis and decision-making layer, where the AI-ML models are used to create the DT-based system. Therefore, the virtual model is built by applying the AI-ML models to the generated data. Then, once the DT is generated, data from the physical and virtual environment are fed into other AI-ML models to achieve goals such as design optimization, performance prediction, or predictive maintenance. In addition, these results can be further used to update and improve current and future prototypes. Fig. 3.2 illustrates the entire data flow for the creation of a DT based on artificial intelligence.
However, although AI plays a key role in the development of DTs, custom selection of the best model among hundreds of available ML models is very cumbersome. Indeed, each AI approach has different levels of accuracy and efficiency and is designed for different datasets. Therefore, depending on the DTs purposes, selecting the best ML algorithm and features is challenging.

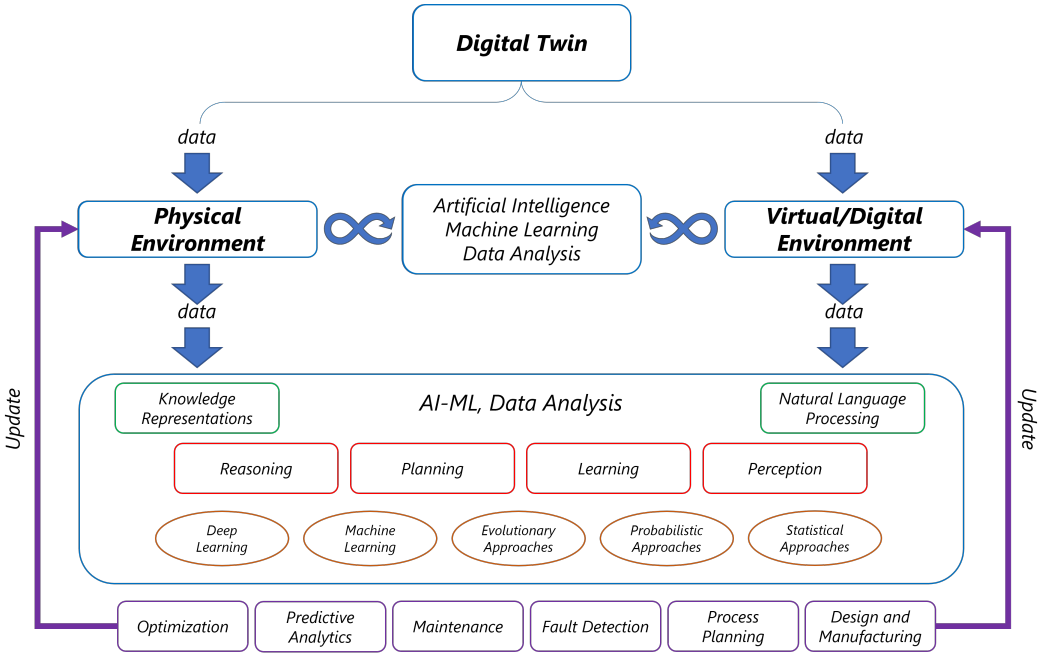


Figure 3.2. Data flow framework for DT using data analysis and AI-ML tools.

3.2 Data for Holistic Understanding

By collecting and processing data, DTs can lead to three levels of knowledge and control of decision making for an industrial process: first, understanding what exists and what is happening; second, understanding and predicting future behavior; and finally, understanding how to act to improve performance. Therefore, through the combination of these three levels, the more a DT enables systemic understanding and control of decision making, the more holistic understanding and value is provided to the industrial process.

- *Understanding what exists and what is happening*: at this stage, through AI-aided descriptive modeling of the components and interconnections of real and virtual environments, a holistic representation of what is being engineered in the present moment is achieved. Indeed, data collected by increasingly intelligent sensors, integrated with data generated by DTs, enable efficient monitoring of machinery, products and other assets, providing a clear and accurate picture of what is happening now. In addition, historical operational data, provided in the form of time series, offer insights into what has happened in the past, and their combination with design data can facilitate the detection of anomalies;

- *Understanding what will happen*: additional value can be provided by predicting how the system will evolve, and what is likely to happen in the future. Learning and extrapolation from historical data allow us to learn what will happen in the future based on situations already observed in the past. Particularly important at this stage is also the simulation of the real system using the DT. The simulation is useful both for verifying the impact of system-level events

on individual processes and/or subsystems and for more reliably predicting scenarios that never occurred or occurred only rarely. ;

- *Understanding how to act and control performance*: DT represents the virtual testing environment, combining monitoring and predictive capability through the testing of different scenarios and optimization loops. DTs offer the ability to virtually simulate circumstances that would be impossible to test in the field. In this way, strategies and scenarios can be simulated to discover their impact before choosing which to implement. This holistic understanding and subsequent value creation, consequently, results in measuring risks in advance, optimizing safety levels, and assessing the impact of major changes or disruptions.

DTs, therefore, provide holistic understanding in real time, offering insights into the system, the ability to predict and investigate its future states, and ensuring its control and optimization. However, to bring added value through this new technology to energy industry sectors, such as exploring the feasibility of new turbomachinery design solutions or developing a condition-based maintenance framework for energy systems, some key points are required:

- *Data connections*: such as combining data collected during normal machine operation or experimental campaigns using IoT sensors, data sampled through numerical simulations, and historical data from both design and operation to enable synchronization between the real and virtual worlds;

- *Descriptive and predictive AI models*: to inform, monitor, predict and react accurately and promptly through data analysis and Machine Learning tools, anticipating situations and providing awareness of a system's evolution;

- *Analytic services*: to detect trends and anomalies, trigger alarm notifications and test the robustness of alternative scenarios.

3.2.1 Learn from Experimental Campaigns

Although experimental campaigns on turbomachinery and energy systems capture the fundamental assumptions of the phenomena, as no arbitrary assumptions or models embedded are in the process, they are characterized by some difficult challenges to overcome. Indeed, a single experiment cannot provide all the necessary information at once: for example, both time-resolved and time-averaged measurements are needed to fully map the behavior of turbomachinery turbulent flows. For this reason, measurements with different levels of complexity and detail are combined together at the end of an experimental campaign, and the challenge lies precisely in to choose the right measurement tool for the right job [44].

According to Dwoyer et al. [45], the process of learning through experiments can be summarized as in Fig. 3.3, where observations mean the set of measured variables, i.e., both data from the physical (real) world and obtained through experimental campaigns. Then, when enough data has been collected, theoretical ideas and models with a coherent structure are created. This process is entirely human-driven; everything is embedded in a closed loop, in which new observations challenge existing

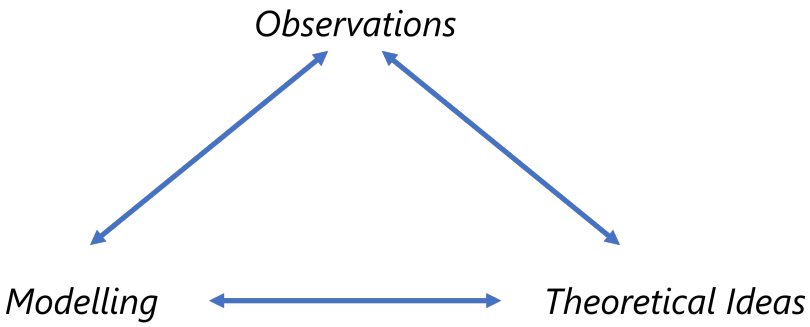


Figure 3.3. Learning by experiments.

models and the models themselves always require new heuristic tests, so the success or failure of an experimental campaign depends solely on the observer's ability to find solutions to the issues that however arise when trying to study complex phenomena. In fact, it is really difficult to successfully develop a heuristic model capable of accurately reproducing the underlying physics of complex energy systems phenomena. For example, in the field of turbomachinery, existing empirical flow models suffer mainly from inaccuracies caused by: the difficulties encountered in modeling the chaotic nature of turbulence; the limited number of observations on which tune a model that can be considered simultaneously; and finally, the limited time and computational costs required to solve the problem.

In spite of these difficult challenges, great strides have been made in the past decade due to the increasing use of big data techniques, which have moved turbomachinery flow modeling in a new direction. Therefore, the usual approach of learning by observation now becomes learning by data, and with the big data perspective, a new pattern can be developed (Fig. 3.4).

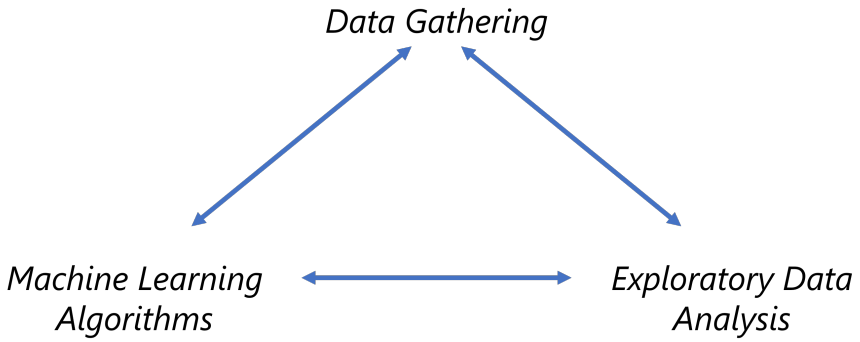


Figure 3.4. Learning by data.

The collection of raw observations is replaced by the gathering of data. These activities are somewhat identical but present some differences involving: the amount of observables that are simultaneously recorded (from a few thousand for the first to several billion for the second) and also the different types of data (e.g., images, signals, measurements) that contribute to form the organized and unorganized data

archives [46]. Data gathering is thus a dynamic process: observables constitute a constant flow of data on which models can be periodically and automatically trained, improved, and adapted. Then, the Exploratory Data Analysis (EDA) phase replaces the theoretical ideas block. Through a variety of statistical tools, in fact, by means of EDA it is possible to transform a data package of any given shape and origin into a mathematical-statistical description. Eventually, the modeling phase is the one most affected by the introduction of data analysis techniques, since Machine Learning tools provide access to a whole new family of powerful algorithms that improve and speed up the standard approach of learning by experiments, as they are able to leverage more data simultaneously and better model it through statistical tools.

3.2.2 Learn from synthetic datasets, CFD

Computational fluid dynamics (CFD) refers to the resolution of a discrete approximation of the conservation equations using numerical methods to obtain accurate knowledge of turbomachinery performance by calculating various quantities of the flow, such as fluid velocity, pressure, temperature and other derived quantities. CFD is the main tool for research, design and optimization of turbomachinery processes and prototypes in both academic and industrial settings, as it allows simulation of geometrically complex configurations without the need to build and test very expensive prototypes. Investigating almost any internal flow condition in turbomachinery, it is easy to imagine that the field of CFD is very broad. In fact, it is possible to characterize a CFD simulation according to the system of equations (e.g., compressible or incompressible NS equations, stationary or non-stationary, Newtonian or non-Newtonian fluids, etc.), the discretization adopted (e.g., finite differences, finite volumes, finite elements), the turbulence model adopted, the presence of a specific wall treatment, as well as the mesh resolution and consequently the specific solving approach: Reynolds Averaged Navier Stokes (RANS), Large Eddy Simulations (LES), or Direct Navier Stokes (DNS). But beyond this characterization, it is important to state that it is the phenomena under investigation that define the complexity of the approach, since the best accuracy corresponds to the highest computational cost.

Based on the above concepts, especially in the early stages of a product's life cycle, manufacturers look to CFD as an alternative tool to extensive and expensive preliminary experimental campaigns, as it can always improve and optimize production processes, while constantly reducing the time and cost of prototype testing; for the same reasons, they are more likely to use CFD strategies with the least expenditure of resources, while remaining consistent with physics, even if sacrificing the resolution of any scale of phenomena. Therefore, if only the overall performance of machines are needed, the fastest and widely used approaches to solve the Navier-Stokes equations are the RANS and URANS (Unsteady Reynolds Averaged Navier Stokes) models, which use SA [47], k-ε [48] or k-ω [49] as turbulence models and a wall function to the wall [50]. These approaches provide rapid results using a relatively low amount of computational resources and relatively coarse meshes, so lower-scale phenomena such as boundary layer resolution and small-scale turbulence can be modeled only by using analytical equations provided by ad-hoc solutions

derived from the observation of few canonical flows. Transient, unstable, small-scale or fluctuating features, however, can only be captured by models of higher level such as LES, with all the complexity that results. In fact, although they can reproduce a wider range of phenomena with greater fidelity, they are rarely used because they require finer discretization and consequently higher computational resources; they are used only for solving very specific problems. Finally, DNS is extremely demanding in terms of both computational effort and computational time and is practically never used during the design and production phases of turbomachines, whereas it is sometimes employed to build high-fidelity data sets.

In contrast to the above advantages of using numerical simulations instead of experimental campaigns, it should be clarified that CFD cannot completely replace experimental tests, as it is not an exact science and several aspects of flows are approximated to speed up simulations. Therefore, turbomachinery flow modeling lives in a constant trade-off between accuracy and computational costs of numerical simulations. Even the simple assumption of a steady flow in a RANS simulation can lead to seriously misleading results due to incorrect modeling of the phenomena [51]. Great efforts have been made in recent times to solve these problems, and turbulence modeling through big data-based approaches is raising great expectations in the scientific community [52], since Machine Learning can compensate for the deficiencies of less complex models (RANS) due to the intrinsic approximations of these models.

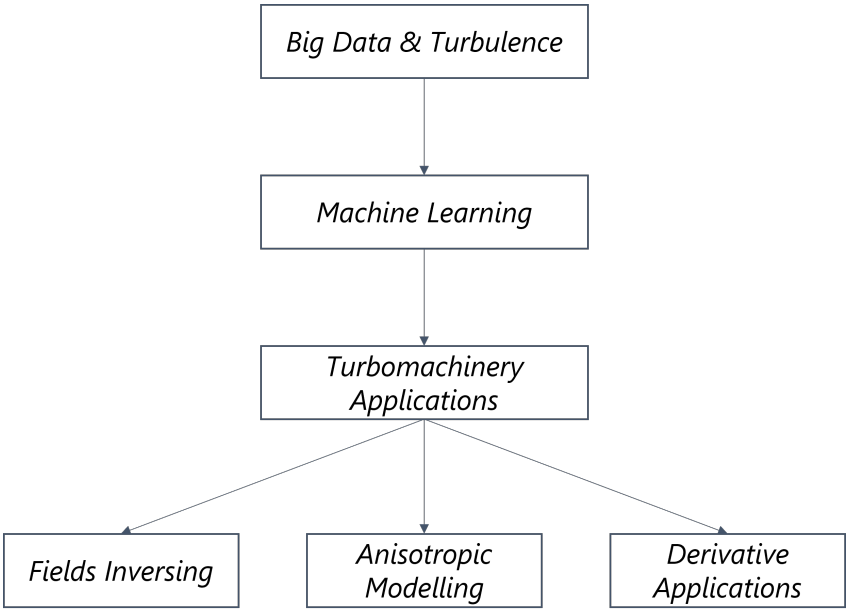


Figure 3.5. Turbulence modeling under big data perspective.

In particular, as highlighted in Fig. 3.5, starting from the large amount of available data on turbomachinery performance, ML algorithms can be successfully applied to:

- *Field inversing*: the goal of this approach is to reduce the discrepancies between LES/DNS and RANS simulations. A Machine Learning algorithm is trained on data from numerical simulations with higher degrees of accuracy or from

experiments, using flow variables as input features, to correct and improve the accuracy of RANS models in inferring the correct field without increasing the computational cost;

- *Anisotropic modeling*: that methodology involves ad-hoc data-driven corrections to the anisotropic part of the Reynolds stress tensor or to the dissipation term in the nutilda equation;

- *Derivative approach*: approaches that seek to extract information of various kinds from fluid dynamic fields with the ultimate goal of not deriving a model of turbulence, but exploiting Machine Learning algorithms to discover hidden relationships.

3.2.3 Learn from sensor datasets, SCADA

Supervisory control and data acquisition (SCADA) is a control system architecture that includes computers, networked data and communications for high-level supervision and data collection of machines and processes. The main components of a SCADA system are:

- *Supervisory computer*: collects data and sends controls back to the process in question;

- *Remote terminal units and programmable logic controllers*: Remote Terminal Units (RTU) and Programmable Logic Controllers (PLC) are used for sensors and actuators connections;

- *Communication infrastructure*: connects the RTU and PLC to the supervisory computer;

- *Human-Machine interface*: Human-Machine Interface (HMI) enables the interaction between operators and the supervisory system by letting operators set parameters within the process.

Applications of SCADA systems are found in a variety of industrial processes and, in particular, are a standard and vital element in power generation plants such as district heating networks and wind farms. The major features of the SCADA system are [53]: monitoring operational data collected and displayed in near-real time (SCADA system sends data on 5- or 10-minute intervals [54]); and, as for the reporting and controlling functions, having access to all relevant information of the energy system, the collected SCADA data enables data-driven performance analysis and, eventually, provide remote control to change the operating mode of the machines.

Numerous efforts have been made in recent years to develop efficient and cost-effective monitoring techniques for energy systems; in particular, research is now focusing on the development of AI-based frameworks for Condition-Based Monitoring (CBM) by exploiting SCADA data. In fact, since Machine Learning tools are easily adaptable to changing conditions, are able to model nonlinear phenomena and leverage historical data such as SCADA data, it is possible to capture hidden

discrepancies within those data and provide early warnings of incipient failures before catastrophic breakdowns occur.

To conclude this chapter, a summary of the state of the art of AI-ML developments in digital twinning for turbomachinery and energy systems applications is presented in Table 3.1.

Table 3.1. State-of-the-art AI-ML developments in digital twinning for turbomachinery and energy systems applications.

Paper	AI-ML Techniques	DT use-case and Application
[55]	Supervised Learning	Turbomachinery flow modeling: wall-function for rotating passages
[56]	Unsupervised Learning	Turbine rotors: complex blade tip geometries design
[57]	Unsupervised Learning	Turbomachinery flow modeling: turbine blades internal cooling channels design
[58]	Supervised Learning	Aero-engine: fault detection
[59]	Unsupervised Learning	Photovoltaic: anomaly detection
[60]	Unsupervised Learning	Wind turbine: anomaly detection
[40]	Supervised Learning	Centrifugal impeller: aerodynamic performance optimization
[61]	Supervised Learning	Aero-engine: fault prediction and maintenance
[62]	Unsupervised Learning	Turbomachinery design: axial turbomachinery performance analysis
[63]	Unsupervised Learning	Turbomachinery operation monitoring: predictive maintenance
[64]	Supervised Learning	Wind turbine: performance analysis and fault detection
[65]	Supervised Learning	Diesel engine: performance analysis and fault detection
[66]	Supervised Learning	Turbomachinery flow modelling: turbulent heat-transfer prediction
[67]	Unsupervised Learning	Turbomachinery flow modelling: modelling of mixing layer
[68]	Supervised Learning	Aero-engine: predictive maintenance
[69]	Supervised Learning	Turbomachinery design: blade profile optimization

Chapter 4

Machine Learning Tools and Techniques

The following chapter provides a definition and then complete and comprehensive overview of Machine Learning algorithms that have been used and exploited for the development of AI applications that can be incorporated into Digital Twin processes related to turbomachinery and energy systems; these AI applications will be discussed in detail later in the work.

4.1 A definition of Machine Learning

In traditional programming, coding the behavior of a program means codifying the behavior assumed by machines; in fact, an algorithm is designed, codified in a language understandable to the machine and finally executed by one or more processing units. This implies that whatever the complexity of the algorithms, machines cannot go beyond the limits set by the code, and their performance, unambiguously defined once programs are designed and coded, can only be improved by changing the type of input data.

Machine learning has emerged over the past decade as an answer to problems where a challenging task coexists with the necessity for continuous self-improvement of algorithms. In fact, Machine Learning allows machines to learn autonomously and progressively from data. In face recognition, for example, it is crucial to have a software that can correctly identify each individual in different poses and lighting conditions, getting better and better as the algorithm is used. Therefore, Machine Learning can be defined as the science of enabling computers to act in a specific way and under a specific circumstance without being explicitly programmed.

4.2 Data Treatment: the importance of Data

A successful Machine Learning algorithm relies primarily on the "quantity and quality" of data fed to it during the training phase. Every Machine Learning campaign begins with the *data mining* phase, which is essentially the process of collecting data related to the specific problem that one intends to solve. These data

can be gathered from experimental observations, numerical investigations, existing databases or IoT sensors.

Obviously, even a deeply optimized model will perform bad if the quantity and quality of training data are insufficient. In fact, if the training data come from poor quality measurements and include unrepresentative data, outliers and/or noise, it will be impossible for any Machine Learning algorithm to find the underlying structures and relevant correlations among the data to perform successfully.

A second, no less important aspect to consider is the fact that a prediction model will perform well only if the training data will not contain too many irrelevant features, since these increase the complexity of the model and reduce the ability to generalize well to unseen data. Therefore, it is important to identify and select the number of relevant features strictly necessary to describe each training sample within the dataset. By means of the *feature engineering* process, a good set of input features is determined. This stage basically involves two main aspects:

- *Feature selection*: selection of the most representative features from the whole set of features in the dataset (tools for feature selection will be introduced later);

- *Feature extraction*: creation of new, more representative features by combining existing ones (e.g., dimensionality reduction methods such as Principal Component Analysis and Projection Latent Structures; these tools will be introduced later).

For this reason, it is essential to spend a great deal of effort on enhancing the quality of the data before implementing Machine Learning applications; the success of the ML-model depends on it. This phase is known as Exploratory Data Analysis and allows the user to take a number of actions during data preprocessing, such as removing outliers, improving the shape of the distribution, or reducing irrelevant features. The data treatment flow is highlighted in Fig. 4.1.

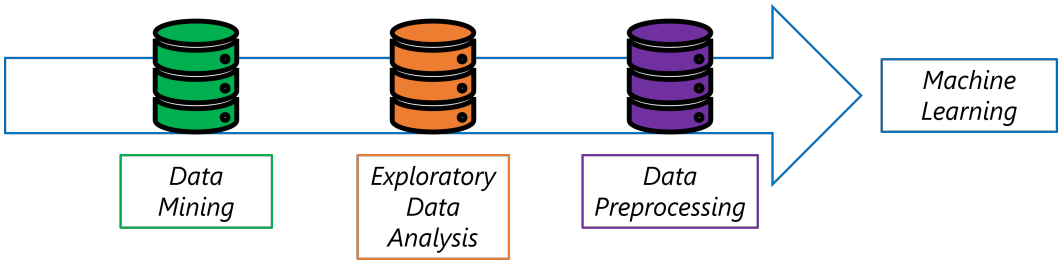


Figure 4.1. Data treatment workflow.

Typically, data treatment and preprocessing take more time than the development of the entire predictive models. However, since Machine Learning algorithms are seriously impaired by the structure of the harvested data, this preliminary stage has more influence on the quality of the final model than any hyperparameter optimization loop.

4.2.1 Normalization Techniques

Since Machine Learning techniques are built on statistical analysis and are therefore strongly influenced by the distributions and quality of the data, a preliminary and fundamental step before any deeper analysis is normalization or scaling of the data. The procedure aims to bring all features in a dataset into the same range, because Machine Learning algorithms are hampered by features with large variations in magnitude; in fact, a predictor might overlook or overestimate the influence of features with the smallest values and those with the largest values, respectively.

In general, the most commonly used normalization techniques are *min-max normalization* and *Z-score*. The first normalization strategy represents a simple scaling procedure, that transforms the original data in a new range between 0 and 1:

$$z = \frac{x - min(x)}{max(x) - min(x)} \tag{4.1}$$

where z is the scaled feature vector.
Z-score is a standardization technique that transforms the data in the following way:

$$z = \frac{x - \mu}{\sigma} \tag{4.2}$$

where μ is the mean value of the original dataset and σ represents its standard deviation. This standardization technique is particularly effective with data characterized by a Gaussian distribution and produces a new centered Gaussian distribution with standard deviation equal to 1 (also known as the *standard Gaussian distribution*).

To focus on the topic of interest of this thesis: when processing fluid dynamic datasets, it is very effective to use another normalization strategy called *local normalization* [70]:

$$z = \frac{x}{|x| + |y|} \tag{4.3}$$

where z is still the normalized feature, x represents the original feature and y represents a normalization factor chosen according to the local properties of the flow field.

4.2.2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) represents an approach to data analysis that through statistical and graphical techniques maximizes insights within the dataset, uncovers hidden structures, detects and removes anomalies and outliers, and extracts the most relevant features.

4.2.2.1 Removing Outliers: Box and Whiskers Plot

During the exploratory data analysis phase, a very important role is played by Box and Whiskers plots (or Boxplots), which are standardized statistical tools for visualizing the distribution of data in graphical form using five numbers: *minimum*, *lower quartile* (Q_1), *median*, *upper quartile* (Q_3) and *maximum* [71]. Figure 4.2 shows the sketch of a boxplot, where the median represents the middle value of the i-th column of the dataset, and the lower and upper quartiles are the 25^{th} and 75^{th}

percentiles, respectively. As shown in the illustration, the lower and upper quartiles define the extreme limits of the box that contains 50% of samples and is referred to as the *Inter Quartile Range* (IQR), while the median is identified by the line drawn inside the box. The whiskers range from the extreme sides of the IQR box to the minimum and maximum values, which are:

$$\begin{aligned} min &= Q_1 - 1.5 \cdot IQR \\ max &= Q_3 + 1.5 \cdot IQR \end{aligned} \tag{4.4}$$

At this point it is important to note that the minimum and maximum are not the lowest and the biggest values in the data set, but all samples smaller than the minimum or larger than the maximum are considered *outliers*. In a boxplot the outliers are drawn as individual points below the minimum value or above the maximum value.

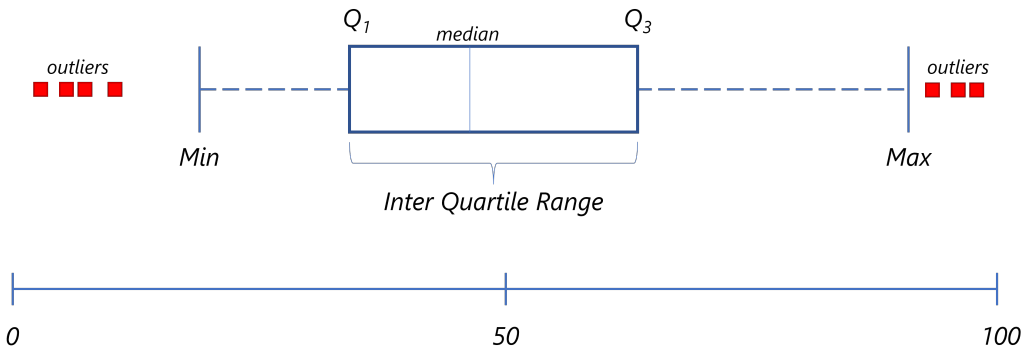


Figure 4.2. Example of a Boxplot.

Boxplots are thus generally employed to identify outliers, in other words, those samples due to errors that occurred during data generation and with values too far out of range to be considered simple anomalies or borderline configurations. In addition, boxplots allow the distribution of data across their quartiles to be visualized much more quickly and compactly than histograms and density plots, making them useful when comparing the distribution of multiple features.

Eventually, it is worth noting that the removal of outliers from a dataset results in a change in the shape of their statistical distribution. In fact, in statistics by means of the *Kurtosis* indicator one can describe the extension of the tails of distributions: long tails denote the presence of outliers which affect the symmetry of the data. Removing outliers means therefore reducing the size of the tail and improving the simmetry of the data.

4.2.2.2 Data Distribution Shape Improvement

In data science when dealing with data analysis or developing Machine Learning tools, it is very important to pay attention to the statistical distribution of the available data. In fact, most statistical methods on which these tools are based assume a Gaussian distribution (also called normal distribution) of the data: the

well-known "bell-shaped" curve, as data with this distribution make parametric methods more powerful and improve their performance.

In general, there are many reasons why the dataset may not exhibit a Gaussian trend; for example, some of the most common reasons are: the size of the dataset is too small, or the presence of extreme values that generate long tails in the data distribution. However, even if the shape of the data is not Gaussian, there are several techniques that, once applied, help in reducing the distribution asymmetry (or *skewness*), making a distorted (or skewed) data sample more normal.

The *skewness* quantifies the lack of symmetry in the statistical shape of the data and thus provides a measure of the degree to which their distribution is skewed with respect to the Gaussian distribution. A data distribution is said to be symmetric when its skewness is equal to 0; therefore, in such a configuration the mean and median coincide. A non-symmetric data distribution is characterized by a sharper trend of data on one side than on the other side and can be positively skewed (also known as right-skewed) or negatively skewed (also known as left-skewed). A positively skewed distribution has a long tail extending toward the right side of the distribution with the mean and median greater than the mode, which is identified by the highest histogram within the distribution: the most observed value. The negatively skewed distributions show an opposite trend, with mean and median lower than the mode, while the tail of the distribution is longer on the left side of the distribution. Figure 4.3 shows the three different categories. Moreover, a distribution is called *highly skewed* when it is characterized by a skewness value greater than +1 or less than -1; whereas, when this value is between +0.5 and -0.5, the distribution is called *moderately skewed*.

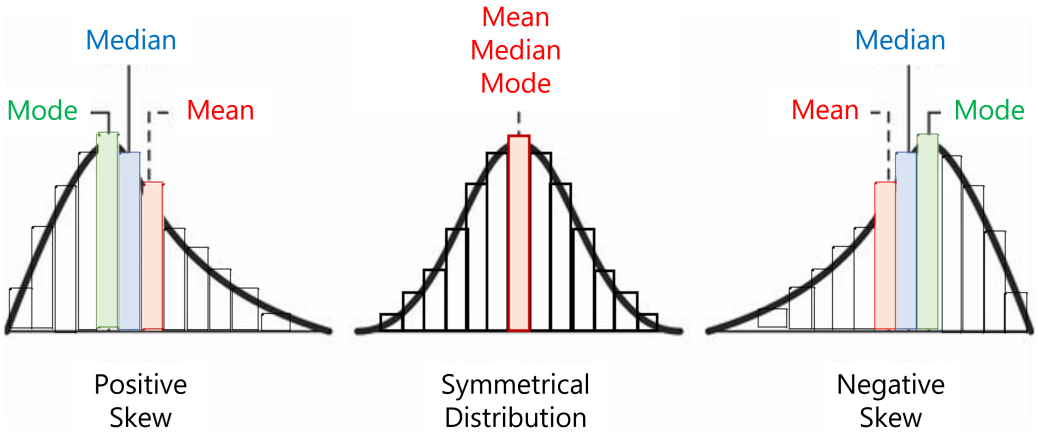


Figure 4.3. Skewness.

In general, treating a skewed distribution with statistical methods that generally assume an approximately normal distribution of the data can lead to problems such as obtaining misleading results. To overcome this issue and enhance the effectiveness of statistical analysis, a viable solution involves transforming the data to make the skewed distribution more like a Gaussian distribution. This is achieved through *transformation functions* that replace each data value with a new number, which is a function of the original value.

When dealing with positively skewed data, the most commonly used transformation functions are: *square root*, *cube root* and *logarithm*. One of the transformations that has a great effect on the shape of the distribution is the *cube root* transformation: $x' = x^{\frac{1}{3}}$. This can be applied to both positive and negative values; however, it is less powerful than the logarithmic transformation. The *logarithm* transformation: $x' = log(x)$, among all the possible transformation functions is the most powerful strategy for improving the skewness of distributions, but it can be applied only to strictly positive values. Finally, the *square root* transformation: $x' = x^{\frac{1}{2}}$, can also be applied only to positive values.

The *quadratic* transformation can be used, on the other hand, to improve data with negative skewed trend. In fact, the quadratic transformation: $x' = x^2$, moderately reduces the left skewness.

The focus for now has been on conventional transformation functions, which are computationally simple and inexpensive, but present some limitations in terms of applicability range and effectiveness of non-normality reduction. *Power* transformations, a new family of transformation functions that elevate numbers to a specific exponent, were therefore developed to overcome those limitations. Underlying the new strategy is the idea of using a potential continuum of transformations that provides a wide range of opportunities to tailor a transformation to the target dataset [72].

The *Box-Cox* transformation [73] is one of these new functions and can be applied regardless of whether a distribution is positively or negatively skewed. Recently, another transformation function has been added to the new family of power transformations: the *Yeo-Johnson* transformation [74], which can be applied without limitations on x and exploits different properties of the Box-Cox transformation.

4.2.2.3 Data Dimension: Feature Selection

An additional necessary step before building a Machine Learning model is *feature selection*. This process has a huge effect on the model's prediction capability, as databases are characterized by a plethora of features detailing each sample included in the data, but generally all features may not be necessary for building the prediction model and, in some cases, they also impair its prediction performance. It is worth mentioning that the feature selection procedure does not result in the generation of new features, but merely selects the most significant features based on a specific criterion: the *correlation* between features.

Correlation is a statistical technique that quantifies relationships between quantitative and continuous variables; it is expressed by a *correlation coefficient* that gauges the strength and direction of linear dependence between each pair of features in a dataset. The correlation coefficient, also known as Pearson correlation [75], is expressed by the ratio between the covariance (cov) of a pair of variables and the product of their standard deviations (σ):

$$\rho_{XY} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{4.5}$$

The range of the correlation coefficient value is between -1 and +1. Therefore, directly correlated features have a correlation coefficient +1, inversely correlated

features have a correlation coefficient -1, and finally, uncorrelated features have values close to zero.

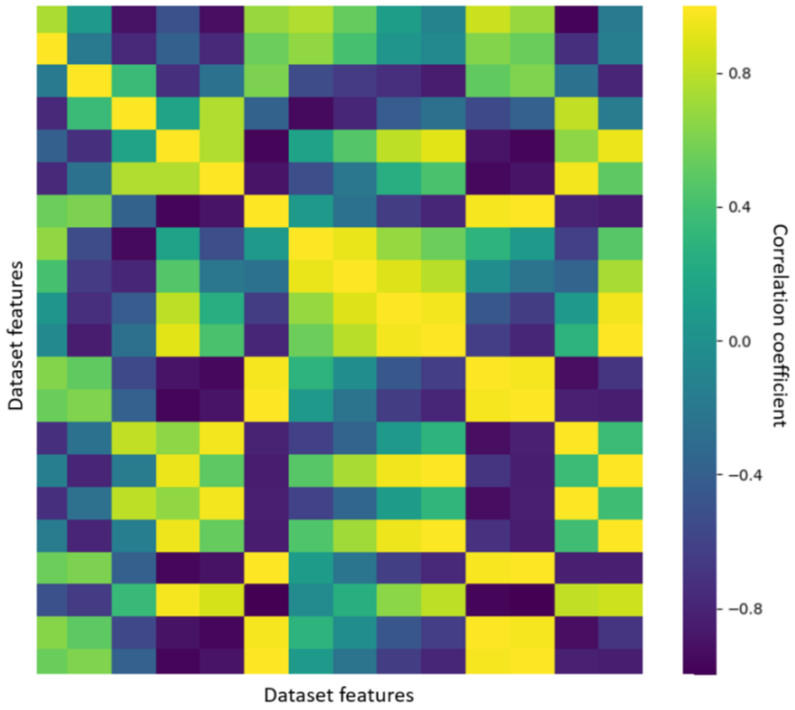


Figure 4.4. Heat map representation of correlation matrix.

Correlations between features in a data set can be quickly and graphically analyzed through the correlation matrix, an example of which is shown in Figure 4.4. This is a graphical representation in which each element of the matrix (i.e., heat map) is represented using a color spectrum to show the correlation value between each pair of features.

Another feature selection technique is based on the *Predictive Power Score* (PPS) [76], which is an asymmetric, data type-independent index that helps identify linear or nonlinear relationships between features in a dataset. The spectrum of PPS values varies between 0 and 1. Through this index we can understand how useful one feature is in predicting the values of another feature in the dataset. In general, a PPS score close to 1 is considered optimal, and it implies that a given column A is very likely to predict the values of column B; whereas if the PPS score is close to 0, then column A may not be useful in predicting the values of column B. The PPS index is computed by considering a single input feature (x_i) per time that tries to predict the target variable (y_i) via a Decision Tree algorithm with the mean absolute

error (MAE) as evaluation metric and is expressed by the formula:

$$PPS = 1 - \frac{MAE^{a_i,b_i}_{model}}{MAE^{b_i}_{naive}} \tag{4.6}$$

where $\mathrm{MAE}^{a_i,b_i}_{model}$ is the mean absolute error of the regression model that predicts $\mathrm{y_i}$ starting from a candidate $\mathrm{x_i}$ and $\mathrm{MAE}^{b_i}_{naive}$ is relative to a naive model that always predicts the median of $\mathrm{y_i}$.

In general, it is difficult to establish a specific minimum threshold for PPS applicability; Table 4.1 summarizes its guideline levels.

Table 4.1. PPS Levels ranking.

PPS Value	**Predictive Power**
PPS == 0	No predictive power
PPS < 0.2	Weak predictive power
PPS > 0.2	Strong predictive power
PPS > 0.8	Deterministic relathionship between features
PPS == 1	Perfect predictive power

4.2.3 Synthetic Data for Machine Learning

AI models require large and accurately labeled datasets for efficient analysis, training and performance; if there is a lack of data, reliable design will not be possible. However, it may be unrealistic due to cost, sensitivity and processing time to collect and label large datasets with thousands or even millions of objects. Therefore, the lack of both qualitative and quantitative data in Machine Learning could be a significant problem. But *synthetic data* can be a good alternative to rely on for training Machine Learning models properly.

Synthetic data are artificial data that mimic real-world observations and are used to train machine learning models when real data are difficult or expensive to obtain. Basically, by creating synthetic data, one recreates something that exists in the real world, obtains its characteristics but does not represent it directly, in other words, a mash-up. Therefore, the generation of synthetic data is totally different from the processes of data augmentation and randomization. In fact, the former is essentially the process of adding slightly modified copies of existing elements to the dataset, while the latter merely moves elements within the data pool instead of creating new ones.

Regarding their conformation, there are two types of synthetic data: partial and full. The *partial type* is a dataset that includes synthetic data and real data from existing observations or measurements. The *full type*, on the other hand, refers to datasets with only synthetic data.

Synthetic data offer several important advantages. First, they offer the ability to customize data to fit conditions that real data do not allow, thus enabling the generation of large training datasets while improving the quality and quantity of available data. Hence, synthetic data can basically serve any goal of any project that

requires computer simulation to predict or analyze real events. There are several key reasons for using synthetic data:

- *Data quality*: in addition to being complicated and expensive to collect, real-world data is often full of errors, containing inaccuracies or representing a bias that may affect the quality of a Machine Learning algorithm. Synthetic data ensures higher data quality, balance, and variety. Artificially-generated data can automatically fill in missing values and apply labels, enabling more accurate predictions;

- *Scalability*: machine learning requires massive amounts of data. It is often difficult to obtain relevant data on the necessary scale for training and testing a predictive model. Synthetic data helps fill in the gaps, supplementing real-world data to achieve a larger scale of inputs;

- *Easy labeling and control*: synthetic data is often simpler to generate and use. When collecting real-world data, it is often necessary to ensure privacy, filter out errors, or convert data from disparate formats. Synthetic data eliminates inaccuracies and duplicates and ensures all data has a uniform format and labeling.

Therefore, synthetic data should accurately represent the original data they are enriching, and also represent an opportunity to securely use sensitive data sets for training or testing purposes, as relevant information can be extracted from these data without impacting privacy compliance. Typical use cases for synthetic data include:

- *Testing*: synthetic test data is easier to generate than rule-based test data and provides flexibility, scalability, and realism. This data is essential for data-driven testing and development;

- *AI/ML model training*: AI model training increasingly relies on synthetic data. Data synthesis can augment real data and upsample rarer events or patterns, enabling the algorithm to train more effectively. Synthetic training data typically performs better than real-world data and is crucial for building high-quality AI models;

- *Governance*: synthetic data helps remove biases present in real-world data. Synthetic data is also useful for stress-testing an AI model with data points that rarely occur in the real world. Synthetic data is essential for explainable AI and provides insights into how models behave.

Although synthetic data offer attractive advantages, they are not easily implemented. In fact, generating synthetic data requires a deep understanding of the phenomenon to be represented and how real data works, as well as the use of sophisticated tools for generating and analyzing data sets. The major challenges related to the generation of synthetic data are:

- *Realism*: synthetic data must accurately reflect the original real-world data. Unless the synthetic data are sufficiently accurate, they will not reflect the

crucial patterns for the training or testing project. Consequently, modeling efforts based on unrealistic data cannot generate useful insights;

- *Bias*: both real-world and synthetic data may contain an inherent or historical bias. If the synthetic data accurately mimics the original, it can reproduce the same biases in the newly generated data. Data scientists must adjust the ML models to account for bias and ensure the synthetic dataset is more representative.

To generate synthetic data, it is necessary to create a robust model that reproduces realistic synthetic data points based on the probabilities that certain data points occur in the real dataset. There are mainly two ways to obtain synthetic data: generative models or conventional ways, i.e., special tools and software along with purchasing data from third parties. Here are some state-of-the-art techniques used to generate synthetic data:

- *Variational Autoencoders (VAE)*: also called VAEs, focus on learning dependencies in the data set. They reconstruct data points from the set in a similar way but also generate new variations. The application of variational autoencoders covers generating different types of complex data such as handwriting, faces, images, and tabular data;

- *Generative Adversarial Networks (GAN)*: the task of the generator is to synthesize new data, while the goal of the discriminator is to check whether it is false or real data. Both work against each other, hence the name "adversarial." The discriminator is trained on real data to differentiate generated data from real data. The generator identifies more realistic data points that the discriminator will classify as real. The process continues until the generator can synthesize data elements that the discriminator cannot differentiate from the real input data;

- *Conventional methods*: these methods include obtaining synthetic data by generating it with software or a tool, or by collaborating with a third party offering such services. Tools and software, which can be found free of charge, can meet testing needs, but may not be sufficient for excellent performance. In addition, choosing this method requires the presence of IT resources in the company. On the other hand, working with companies that provide synthetic data prevents the company from having its own IT staff. In addition, third parties usually specialize in a precise type of synthetic data generation, making them more experienced in the specific field.

4.3 Supervised versus Unsupervised Learning

Machine Learning algorithms can be classified mainly into two different categories: *supervised learning* and *unsupervised learning*.

In supervised learning, a dataset is given and it is already known what the correct output should look like, having the idea that there is a relationship between the input and the output. Supervised learning problems are classified into *regression*

and *classification* problems. A regression problem tries to predict outcomes within a continuous output, which means mapping the input variables onto a continuous function. A classification problem, on the other hand, tries to predict outcomes within a discrete output, in other words, it attempts to map input variables to discrete categories.

On the contrary, unsupervised learning algorithms are able to highlight hidden structures within the data or find correlations between features used to describe each sample without being explicitly programmed to achieve a specific goal. Unsupervised learning algorithms are therefore the cornerstone of data analysis and are sometimes a necessary first step before running a supervised learning algorithm. Dimensionality reduction and clustering are the main branches of unsupervised learning. Figure 4.5 shows a schematic representation of the Machine Learning algorithms classification.

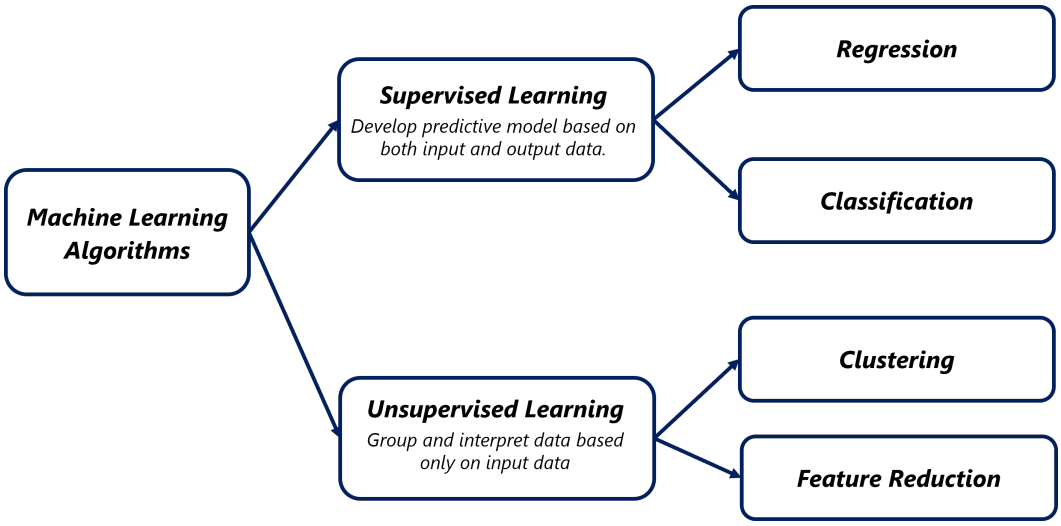


Figure 4.5. Machine Learning algorithms classification.

The work reported in this thesis concerns the design, optimization and application of several Machine Learning algorithms, coded in the Python language with extensive use of the scikit-learn and Tensorflow libraries, aimed at the development of digital twin models for solving turbomachinery and energy system problems through regression and clustering tools. The definition and theory of these algorithms are given below.

4.3.1 Classification

As aforementioned, classification problems fall into the supervised learning category and aim to predict a categorical label (i.e., discrete value) of new observations, based on the learning gained during the training phase of the model. It can be formally defined this way, consider the training dataset as a subset $S \in D$, where D is a domain expressed as:

$$D = ((x_1, y_1), (x_2, y_2), ..., (x_n, y_n)) \tag{4.7}$$

where $x \in R^{\mathrm{n}}$ and $y = (1, 2, ..., k)$ with $k \geq 2$ number of categories. The labeled sample dataset will be used to generate and train a machine learning model in the

form $f: x \to y$, which will later be employed to make predictions about unseen data $(x_i, y_i) \notin S$ since:

$$y_i = f(x_i) \tag{4.8}$$

where f is the prediction model previously trained, x_i is the input feature and y_i is the output or target feature. Therefore, during the training phase, a function is created that, when exploited in the prediction phase, is able to map the data described by the feature set to the correct category they belong to.

The measure of the distance between the real value of observation y_i and the predicted value $f(x_i)$ evaluate the quality of the classification task and can be expressed by the following formula:

$$E = \frac{1}{|U|} \sum_{(x_i, y_i) \subseteq U} f(x_i) \neq y_i \tag{4.9}$$

where U is the set of unseen data, defined as $U = D - S$, the variable x_i is referred to input features and the variable y_i are output variables (or labels) of classification problem.

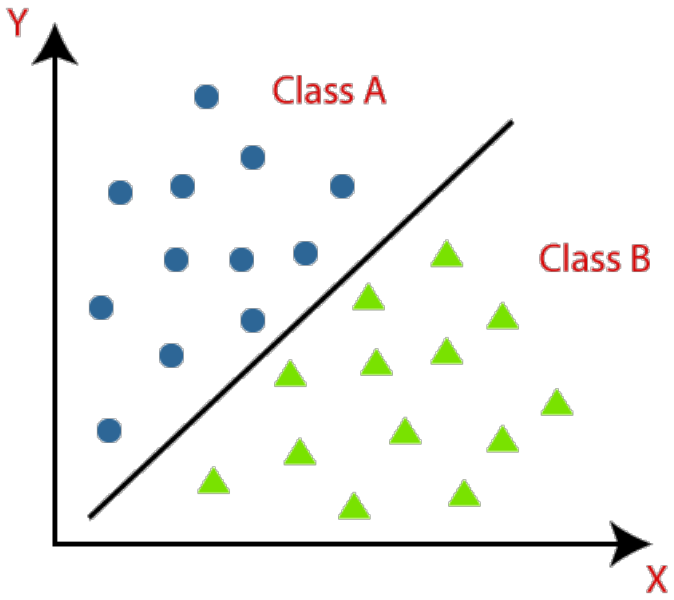


Figure 4.6. Classification problem.

Data classification is a very common problem in machine learning, and several machine learning algorithms can perform this task. Classification methods range from the simplest linear method, in which classification is based on the value of a linear combination of features, to more complex artificial neural networks. Figure 4.6 shows a classification application whose purpose is to divide the set of observations into different categories.

4.3.2 Regression

Regression problems represent a second branch of supervised learning. The objective of a regression algorithm is to determine the relationship between a group of

independent input features and one or more dependent output variables (i.e., target variables). In other words, they estimate the function f that maps x to y, but since all statistical models must account for a random error term, they try to approximate the real function that generated the observations: $y = f(x)) + \varepsilon$. Therefore, the estimate function becomes $\widehat{f}$ and the prediction is:

$$\widehat{y} = \widehat{f}(x_i) \tag{4.10}$$

In Figure 3.2, the line represents the true value of the samples , while the points in the cloud are the values predicted by the trained model. Therefore, in a regression problem the model tries to predict the continuous value of a variable based on past observations (in contrast, in classification problems the variables are categorical).

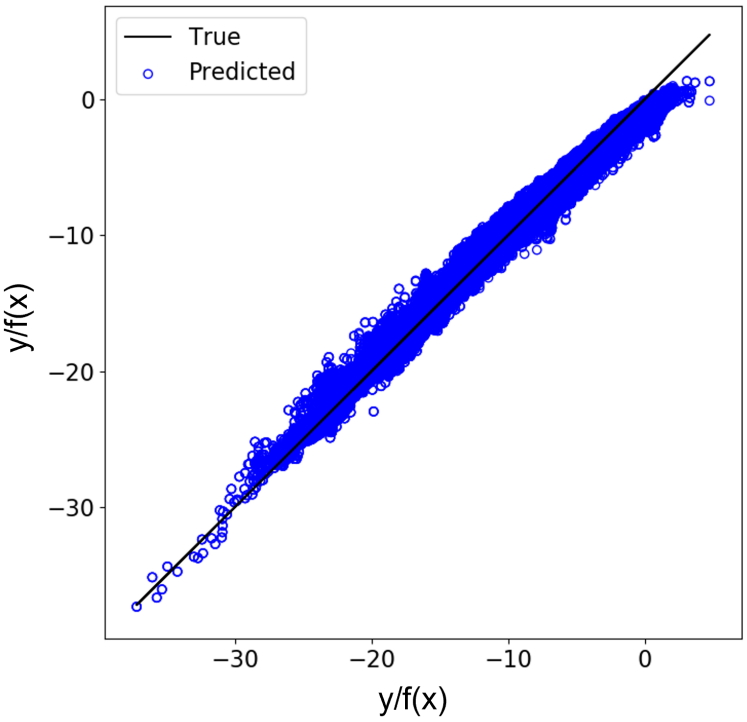


Figure 4.7. Regression problem.

The vertical distance of each cloud point from the corresponding line point estimates the prediction error for the *i-th* observation: $y_i - f(x_i)$

To estimate the best possible model, the quality of the regression analysis must be measured, and obviously since we are dealing with a regression problem such index must be minimized. In the literature several performance indicators are available, but in general they are variations of the well-known mean square error for unseen data:

$$\begin{aligned} E[(y - \widehat{y})^2] &= E[f(x) + \epsilon - \widehat{f}(x)]^2 \\ E[(y - \widehat{y})^2] &= (f(x) - \widehat{f}(x))^2 + Var(\epsilon) \end{aligned} \tag{4.11}$$

where the squared difference in the left part of the equation is the reducible error that the regression must minimize, while in the right part is the irreducible error, or noise within the original model. The reducible error can be further decomposed into a *bias* and *variance* terms [77], with the term bias denoting the error in approximating a real-world problem with a simplified model $\widehat{f}$ and the term variance quantifying the amount of variation in the response of the regression model if it had been modeled on a different dataset.

4.3.2.1 Least Square Method

In regression analysis to solve overdetermined systems, a standard procedure is the *Least Squares Method* (LSM) [78], which can be thought of as a method for determining the line of best fit to a given data set. It then involves adjusting the coefficients of a model function, also known as a response surface, so that it best fits the data. But this procedure can be easily generalized, and instead of looking for the line of best fit, one can also find the best fit given by any finite linear combination of specified functions.

The theory behind the model regards the real function determining the observations as being representable by a polynomial approximation developed using a least-squares approach [79]. The regression model is a function $f(x, \beta)$, where: $\beta = [\beta_1, ..., \beta_p]^T$ is the vector of the p regressors to be tuned, $x = [x_1, ..., x_k]^T$ is the vector of k input parameters and f is a vector function of p elements consisting of powers and crossproducts of power of x up to a certain degree $d \geq 1$. In general, the most widely used LSM models are the first-degree model (d = 1) and the second-degree model (d = 2):

$$\begin{aligned} y &= \beta_0 + \sum_{i=1}^{k}(\beta_i x_i) + \epsilon \\ y &= \beta_0 + \sum_{i=1}^{k}(\beta_i x_i) + \sum_{j<1}\sum(\beta_{ij} x_i x_j) + \sum_{i=1}^{k}(\beta_{ii} x_i^2) + \epsilon \end{aligned} \tag{4.12}$$

when designing a quadratic model, the minimum number of regressors needed to determine all second order polynomial coefficients is equal to $(k + 1)(k + 2) = 2$, with k being the number of factors.

This approach can be used to estimate the relationship between y and x which can be used to address a regression problem and to derive the optimal settings of x that result in a maximum or minimum response in a certain region of interest. However, LSM becomes difficult to apply when a large amount of data needs to be handled since it is more prone to errors.

4.3.2.2 Ensemble Method

One of the most commonly used supervised learning algorithms, employed for both regression and classification, is the *decision tree* algorithm, which is based on binary recursive partitioning [80]. The result predicted by the model is function of a series of questions and conditions; specifically, taking the Figure 4.8 as a reference, each node in the tree represents a feature (or regressor), each split represents a decision

to be made, and each leaf is an outcome. To assess the goodness of a split, the algorithm must estimate a *metric of impurity*: $i(t)$, where t is a potential node to be evaluated.

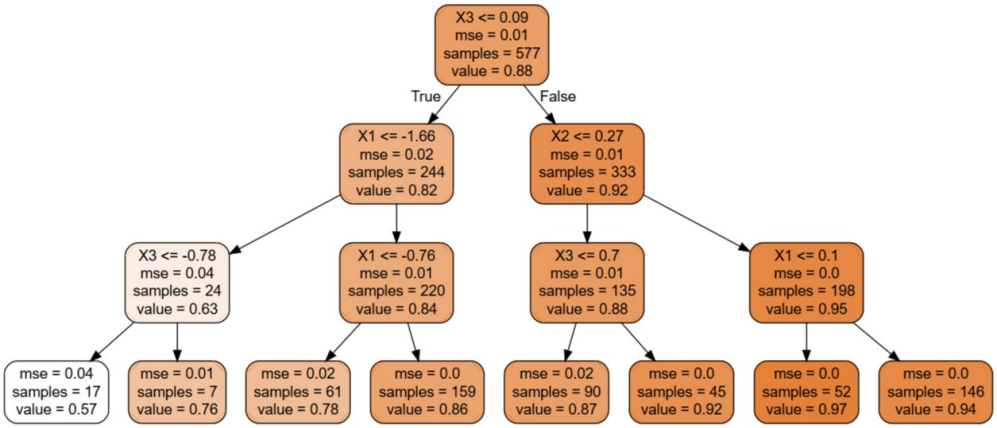


Figure 4.8. Decision tree example.

When dealing with classification problem, each node must compute the probability of incorrectly classifying the data, in other words the *Gini Impurity* defined as:

$$i(t) = G_t = \sum_{i=1}^{C}(p(i))(1 - p(i)) \tag{4.13}$$

where $p(i)$ is the probability of incorrectly classifying the *i-th* sample and C are the different classes.

Otherwise, in regression problems, the impurity index is quantified using the *mean square error (mse)* defined as:

$$i(t) = mse_t = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}(x_i))^2 \tag{4.14}$$

where y_i are the real values, $f(x_i)$ are the predicted values and n are the samples.

The process of dividing the dataset into smaller and smaller subsets continues until no further gains can be achieved or until a predefined rule, such as maximum depth of the tree, is met.

The main limitation of using such a simple model concerns the tendency of the predictive model to overfit: decision trees, in fact, are so adaptable that small variations in the training data result in large variation in the learned parameters; therefore, such a model has high *variance*. On the other hand, a more robust model that is less sensitive to noise is said to have high *bias*, meaning that it makes assumptions following preconceived ideas about the data. In either scenario, the final model cannot generalize well to new data [81]. Therefore, in order to improve the stability and accuracy of these algorithms, ensemble methods have been developed, which using multiple learning algorithms (decision trees), achieve better predictive performance compared to those achieved by only one of the ensemble learning algorithms. Examples of ensemble methods include: Random Forest and Gradient Boosting.

4.3.2.3 Random Forest

One type of ensemble machine learning algorithms is the *Random Forest* (RF), based on a statistical technique called *bagging*. This statistical technique constructs multiple weak learners, decision trees in the case of the random forest, and combines their predictions to obtain a more precise response.

The algorithm generates M subsets of data, each of which has $m \leq n$ samples randomly selected within the original training dataset and are subsequently used to build a tree ϕ_m where $m = 1, ..., M$. Therefore, by averaging all M predictions, one obtains the final bagging prediction $\widehat{f}_b$ for a new sample x_i:

$$\widehat{f}_b(x_i) = \frac{1}{M} \sum_{m=1}^{M} \widehat{f}_{b_{\phi_m}}(x_i) \tag{4.15}$$

Moreover, to achieve better performance, in addition to bagging a second random source is also introduced into the random forest algorithm, that during the splitting of a tree randomizes the partition procedure by considering only a random subset of features Φ_m:

$$\widehat{f}_R F(x_i, \Phi) = \frac{1}{M} \sum_{m=1}^{M} \widehat{f}_{b_{\phi_m}}(x_i, \Phi_m) \tag{4.16}$$

A scheme of that method is shown in Figure 4.9

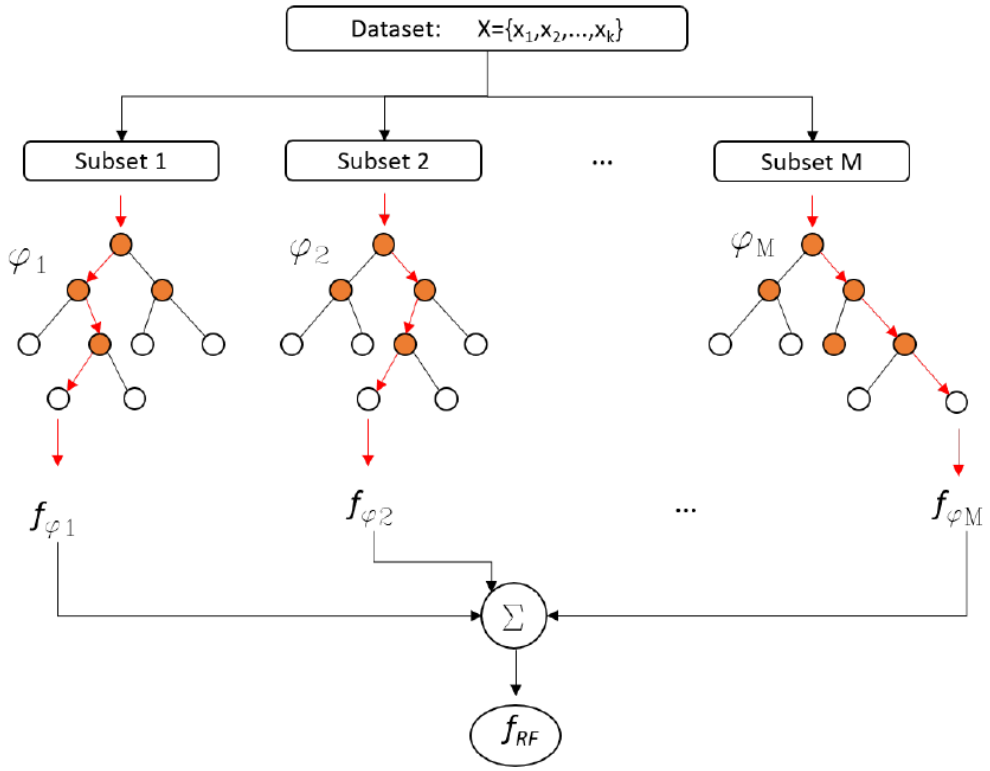


Figure 4.9. Random forest flow chart.

The theoretical basis of the random forest algorithm merging strategy is based on two concepts:

- *Random sampling* of training instances during the building of trees;
- *Random subsets*: of features (or predictors) for splitting nodes.

In a random forest, as each tree is trained and learns from different subsets of randomly selected samples, it decreases the overall variance of the forest without consequently increasing its bias, since deviations in the original training dataset do not affect the final response obtained from the aggregation [82]. For example, if each tree trained on its own subset of data overfits them, then all models have errors, but these are random and uncorrelated. Averaging the responses will reduce the overall error committed by the random forest.

The random forest algorithm is driven by three *hyperparameters*: the minimum node size that is used to define the terminal nodes (leaves) of each tree in the forest, the total number of trees grown $M \in (500 : 1000)$, in fact the accuracy decreases asymptotically by using a larger number of trees [83], and finally the number of features considered randomly for each split. Therefore, being governed by only three hyperparameters, the algorithm is easily implementable and, despite its simplicity, can achieve great performance for both classification and regression. In addition, the model effectively minimizes the risk of overfitting with respect to a single decision tree, and the user can easily evaluate the influence of each feature on the prediction. However, it becomes computationally expensive once it is applied to problems involving a large number of learners and is unable to extrapolate data that lie outside its training interval.

4.3.2.4 Gradient Boosting

Gradient Boosting (GB) [84] is a second ensemble method, which predicts its outcome by combining the results of weak learners. It differs from the RF algorithm in the way trees are constructed; in fact, in a boosting algorithm, trees are constructed in a sequential manner and each new tree is boosted to correct prediction errors made by the previous learner, as shown in Figure 4.10.

Boosting focuses on the most difficult to predict samples. In fact, each new tree is trained using all samples from the original dataset weighted taking into account the success of the previous learner, which means that no partitioning or feature selection is applied. Misclassified samples increase in weight and subsequent trees focus on them during the training phase.

Consequently, the model needs a loss function L to estimate the prediction error committed by each learner and enhance the performance of the one that follows. Even in this case, the most widely used loss function is the aforementioned mean square error:

$$L = mse = \frac{1}{2}(y - \widehat{f}(x))^2 \tag{4.17}$$

The reason for choosing this loss function for gradient boosting is that the user differentiates it with respect to expected value $\widehat{f}(x)$ and the result is an easy-to-manage negative residual $-y_i + \widehat{f}(x_i)$. After the inputs are provided, the initial step

is initializing the model with a constant value $f_0(x)$:

$$f_0(x) = argmin \sum_{i=1}^{n} L(y_i, \widehat{y}_i) = argmin \sum_{i=1}^{n} (y - \widehat{f}(x))^2 \tag{4.18}$$

with $i = 1, ..., n$ representing the number of samples in D.

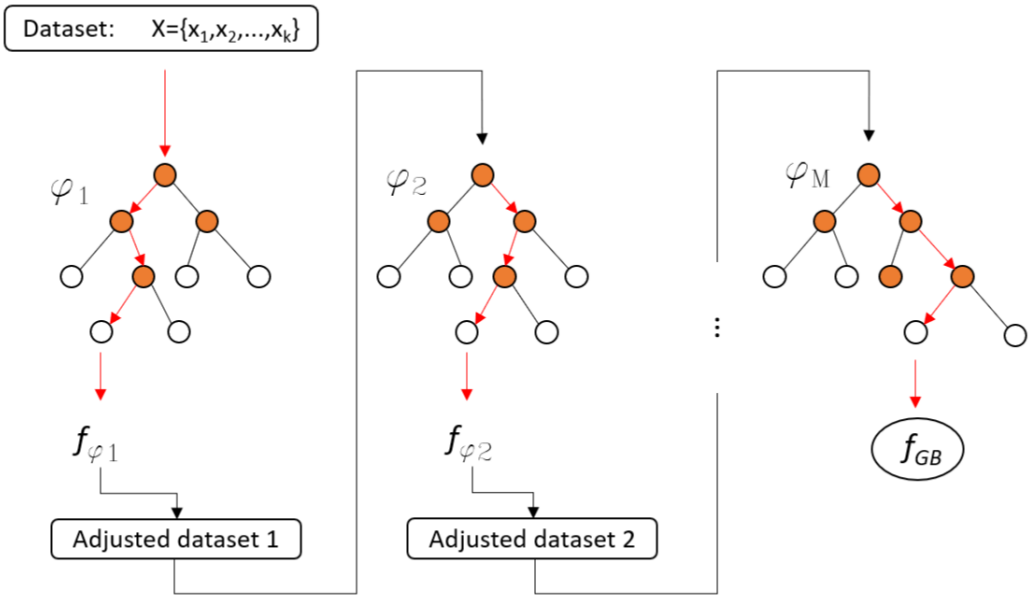


Figure 4.10. Gradient Boosting flow chart.

Thereafter, trees are generated cyclically, and for each new tree the algorithm calculates:

$$r_{i,m} = -\left[\frac{\partial L(y_i, \widehat{f}(x_i))}{\partial \widehat{f}(x_i)}\right] \tag{4.19}$$

where $m = 1, ..., M$ is the number of trees generated in the gradient boosting model and $\widehat{f}(x) = \widehat{f}_{m-1}(x)$. The last equation can be redrafted as:

$$r_{i,m} = (y_i - \widehat{f}(x_i)) \tag{4.20}$$

with r representing a residual for each sample.
The algorithm fits a regression tree to the $r_{i,m}$ values and create a terminal region $R_{j,m}$ for $j = 1, ..., J_m$, whit $R_{j,m}$ representing the leaves and j is the index of each leaf in the tree. As mentioned above, the output values of each leaf are determined considering the prediction of the previous tree $\widehat{f}_{m-1}(x_i)$:

$$\widehat{f}_{jm}(x_i) = argmin \sum_{x_i \in R_{jm}} L(y_i, \widehat{f}_{m-1}(x_i) + \widehat{f}) \tag{4.21}$$

and substituting the effective loss function

$$\widehat{f}_{jm}(x_i) = argmin \sum_{x_i \in R_{jm}} \frac{1}{2}(y_i - (\widehat{f}_{m-1}(x_i) + \widehat{f}))^2 \tag{4.22}$$

the goal of gradient boosting is to find the value of $\widehat{f}$ that minimizes this equation, that is, the average values of the finite residuals in the leaf $R_{j,m}$. The final prediction for each sample becomes:

$$\widehat{f}_{GB}(x) = \widehat{f}_{m-1}(x) + \nu \sum_{j=1}^{J_m} \widehat{f}_{jm} I \tag{4.23}$$

Hence, the result of the algorithm is based on the value of the previous prediction to which the output value $\widehat{f}_{jm}$ is added for all R_{jm} leaves in which a sample is found. ν is the *learning rate* of the model ($\nu \in (0, 1)$). A small learning rate reduces the effect of each tree on the final prediction, improving accuracy in the long run.

Finally, if we want to compare the performance of a random forest with that of gradient boosting, it can be said that although the latter is characterized by a training process with higher computational costs, especially for tuning the hyperparameters needed to avoid overfitting, it is able to outperform the performance of the random forest in terms of prediction ability [85].

4.3.2.5 Anatomy of Multi-layer Perceptron

The feed forward neural network [86] consists of the following elements: the *layers* which are made up using neurons, the *activation functions* that result in the output of each neuron within the network, the *loss function* which describes the feedback signal used for learning and the *optimizer* that determines how learning proceeds.

The *neurons* receive input, change its status according to that input and produce output also depending on the activation function. The single neuron is composed of three functional operators: an input vector $x \in R^k$ is multiplied by weight vector $w \in R^k$ and the resulting vector is summed to the scalar bias b to form the scalar net input z.

Therefore, in a multi-layer perceptron, since each neuron is described by the formula: $a = f(z) = f(\sum_{i=1}^{k} x_i \cdot w_i + b)$, each layer can be parametrized by a *weight matrix* $W_{R \times K}$, with R the dimension of input vector x, that is the number of features describing the problem, and S the specific layer neuron number, and a vector bias b of length S.

Multilayer networks, a sketch of which is provided in Figure 4.11, are characterized by the fact that the output of one layer becomes the input for the next layer; this process, referred to as the back-propagation algorithm, is described by the following equations:

$$a^{m+1} = f^{m+1}(W^{m+1} a^m + b^{m+1}) \tag{4.24}$$

with $m = 0, 1, ..., M$ where M is the number of layers in the network. Neurons in the input layer receive the input features: $a^0 = x$ that become the starting point for subsequent layers. Hence, output layer neurons produce the response the response of the network:

$$y = a = a^M \tag{4.25}$$

In the back propagation algorithm, each artificial neurons have input parameters with associated weights and biases. The weighted sum of these parameters, z, is

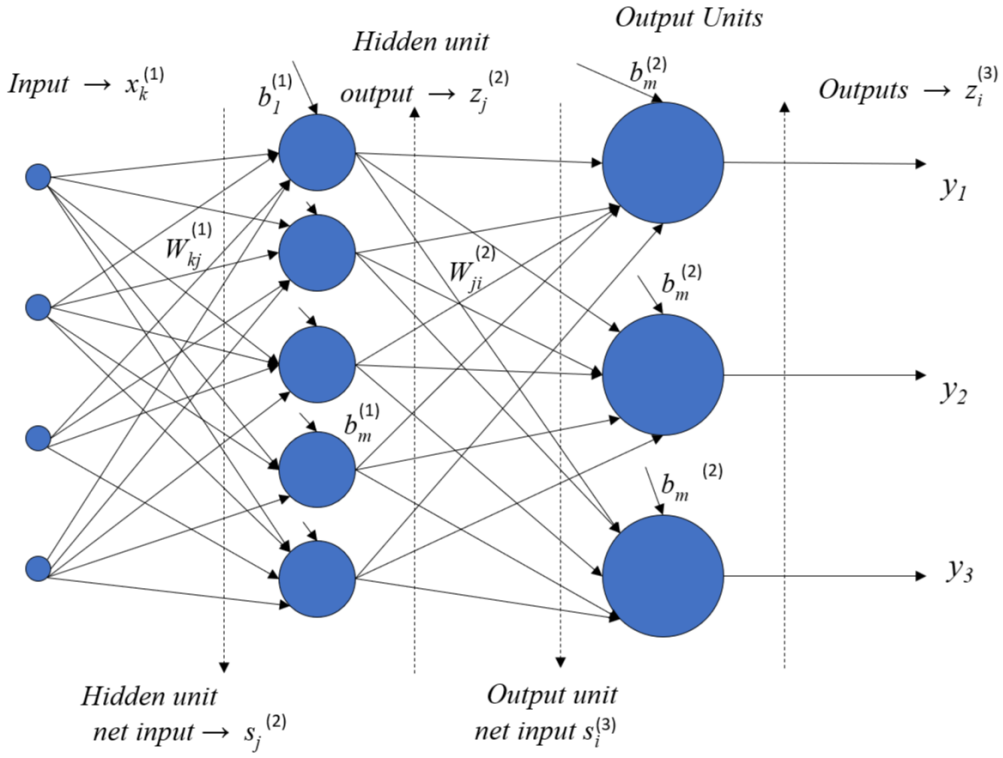


Figure 4.11. Example of a three-layer feed forward neural network.

the input value for the activation function, which applies a transformation *f(z)* to establish whether or not a neuron should be activated to transmit the value to the next level.

One of the simplest activation functions is the *linear function*: $a = f(z) = z$, which is not confined between any range and provides in output the weighted sum of the inputs. In this case, since activation is linear, it is unnecessary to have more than one hidden layer; in fact, this function is generally used only in the output layer to allow extrapolation of the neural network since the function is everywhere defined.

Another activation function is the *sigmoid function*: $\sigma(z) = \frac{1}{1+e^{-z}}$, which is continuously differentiable and has a determined output range, $a \in (0, 1)$. When the value of z increases, the function can vanish the gradient, negatively affecting learning accuracy. This function is mainly used in the last layer or in binary classification models, since the probability varies in the output range of this function.

There is also a modified version of the sigmoid function: the activation function called *hyperbolic tangent*, defined as: $tanh = \frac{2}{1+e^{-2z}}$. With this activation function, the output values are between -1 and 1. This feature simplifies the optimization of the neural network with multiple hidden layers, the main disadvantage still remains the gradient vanishing problem.

Especially for regression problems, the most frequently used activation function is the *rectified linear unit (RelU)*, $R(z) = max(0, z)$. The results of the function are null for any *negative* z, while it returns the *value of the input for positive* z (as a

linear function). The function solves the problem of the vanishing gradient, since the derivative is 0 for negative inputs and 1 for a positive inputs. In addition, the computational cost is lower than that of Sigmoid and Tanh functions. The main limitation of this function is known as the "dying RelU problem": for $z < 0$ the gradient will be null and, consequently, the weights will not be adjusted during the optimization process. Therefore, these neurons will cease to respond to input changes, and become passive. One method to remedy this problem is to use a modified version, called *Leaky RelU*, in which the horizontal line for $z < 0$ has a small constant gradient.

The gradient descent method allows the values of the network weights to be optimized by the loss function. In fact, during the training process the goal is to minimize the distance between the predicted and actual values, and the loss function is a measure of that distance. The most frequently used loss function for regression problems is the mean square error (previously described, see Equation 4.14). However, in some cases, especially when the distribution of target values contains outliers, it is more appropriate to opt for the *Mean Absolute Error*, more robust to outliers:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i, \widehat{y_i}| \tag{4.26}$$

At this point, the role of the optimizer is to iteratively update the weight and bias coefficients to minimize the loss function. The most commonly used technique is *Gradient Descent*, which starts by assigning a random value to the *Gamma* model parameters, after which the model is run and the loss function calculated. The algorithm then computes the gradient of the loss function ($g_t = \Delta L(\Gamma_{t-1})$) and decides the direction in which to shift the weights and bias to obtain a lower loss at the next iteration:

$$\Gamma_{t+1} = \Gamma_t - \alpha g_t \tag{4.27}$$

where α is the learning rate parameter.
Stochastic Gradient Descent (SGD) reduces the computational time for performing a randomly selected parameter update for each training example. But these frequent updates result in high variance in parameter values, leading to intense fluctuation in the loss function. *Gradient Descent with momentum* speeds up the SGD and reduces its oscillations with the introduction of a momentum term γ that adds a fraction of the previously updated vector V_{t-1} to the current value of the parameter:

$$\Gamma_{t+1} = \Gamma_t - \alpha g_t - \gamma V_{t-1} \tag{4.28}$$

A different technique for accelerating optimization convergence is to use an *adaptive learning rate*. Thus, instead of updating the *Gamma* parameters all at once employing the same learning rate *alpha*, for each $Gamma(i)$ parameter the algorithm computes a different learning rate.
The currently most widely used optimizer is *Adaptive Moment Estimation (Adam)* [87], which calculates adaptive learning rates for each parameter and estimates the first $\widehat{m_t}$ and second $\widehat{v_t}$ moments of the gradient to fit the learning rate for each parameter:

$$\Gamma_{t+1} = \Gamma_t - \frac{\alpha}{\widehat{v_t}}\widehat{m_t} \tag{4.29}$$

with:

$$\widehat{m_t} = \frac{(1-\beta_1)\sum \beta_1^{t-1} g_t}{\beta_1^t}$$
$$\widehat{v_t} = \frac{(1-\beta_2)\sum \beta_2^{t-1} g_t^2}{\beta_2^t} \tag{4.30}$$

default values for the new hyperparameter introduced by this method are: $beta_1 = 0.9$ and $beta_2 = 0.999$. Over the past few years, the solver *Adam* has achieved such high levels of optimization that it is currently recommended as the best default algorithm to use for many machine learning applications [88].

4.3.3 Clustering

Clustering analysis can be considered the main application of unsupervised learning and aims to find different structures within a dataset in order to group similar samples in the same group. Among all the clustering algorithm (more than 100 different algorithms are available), a sketch of them is given in Figure 4.12, the most common are: *k-Means*, *Gaussian Mixture* and *DBSCAN*.

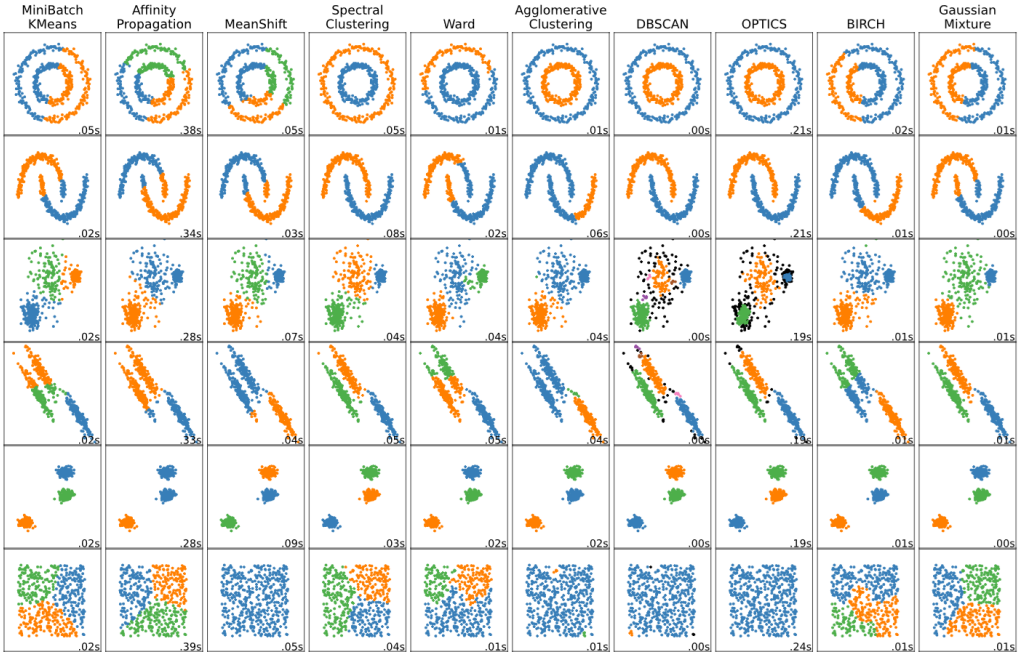


Figure 4.12. Comparison of different clustering methods.

4.3.3.1 k-Means

Although many algorithms have been developed over the years to solve clustering problems, k-Means emerges as the most popular method. The main advantage resides in its simplicity: the algorithm, starting with a random selection of initial centers, repeatedly assigns each sample point to the nearest center, which are then reassigned based on the sample allocations [89]. From a technical point of view,

although k-Means is not the best algorithm in terms of efficiency or quality, in practice it becomes so from the point of view of speed and simplicity [90].

The goal of the algorithm is to group different samples inside the same class according to their *similarity*, which is parameterized using the *squared Euclidean distance* between a pair of points x_0 and x_1:

$$d(w_0, x_1)^2 = \sum_{j=1}^{m}(x_{0j} - x_{1j})^2 = \| x_0 - x_1 \|_2^2 \tag{4.31}$$

with j indicating the sample point dimension(i.e., feature columns). As a first step, each point is assigned to the nearest randomly selected centroid using this distance. Next, the algorithm calculates the *cluster inertia* using the sum of squared errors within each cluster:

$$SSE = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{(i,j)} \| x^i - \mu^j \|_2^2 \tag{4.32}$$

where $\mu(j)$ is the centroid for cluster j and $w(i, j)$ is 1 if the *i-th* sample is inside the *j-th* cluster, otherwise is null. The objective of k-Means is to minimize the cluster inertia. The centroids will be iteratively computed by taking the average of assigned points. The hyperparameters of k-Means are: number of clusters (m), maximum iterations and initial cluster centroids. The initial number defines the number of times the algorithm will be executed with different centroid seeds. The *elbow method* can be used to select the correct number of clusters: plot the increasing number of clusters m with respect to the total variance explained. To obtain a good solution, proper initialization of the initial centroids is essential. *k-Means++* has been proposed to overcome the limitation of random selection of initial centroids, where the initial selection of centroids is based on the following steps: the first centroid is chosen uniformly at random from the data points. Then, for each sample x_i, the distance between the point and the centroid, $D(x_i)$, is calculated and a new point is randomly selected as the second centroid with the use of a weighted probability distribution, $P = f(D(x_i)^2)$. The procedure is repeated until all centroids are initialized [91].

4.3.3.2 Gaussian Mixture

A *Gaussian Mixture* model is basically a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters [92]. Mixture models can be viewed as a generalization of k-means clustering to incorporate information about the covariance structure of the data and the centers of latent Gaussians.

The Gaussian Mixture object implements the *Expectation-maximization* (EM) algorithm for fitting mixture-of-Gaussian models [93]. In fact, the main difficulty in learning Gaussian mixture models from unlabeled data is that one usually does not know which points come from which latent component (if one has access to this information, it becomes very easy to fit a separate Gaussian distribution to each set of points). *Expectation-maximization* is a well-founded statistical algorithm that gets around this problem by an iterative process. First, random components are assumed and the probability of being generated by each model component is calculated for

each point. Then, you adjust the parameters to maximize the likelihood of the data based on these assignments. Repeating this process always guarantees convergence to a local optimum.

The number of mixture components for the clustering algorithm is selected according to the *Bayesian Information Criterion* [94]. The Bayesian Information Criterion is an index for scoring and selecting between two or more models that basically works in this way: looks for contrasts in the density of the data points, picks out spatially separated regions of high-density data estimating centers and extents of these regions, and then counts the number of that regions. This index is expressed by the formula:

$$BIC = kln(n) - 2ln(\widehat{L}) \tag{4.33}$$

where n is the number of data points, k is the number of free parameters to be estimated and $\widehat{L}$ is the maximized value of the likelihood function of the model.

4.3.3.3 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) is a clustering algorithm that employs the distance between the nearest points to identify different clusters within the data. It is based on the assumption that clusters are densely populated regions in the data space which are divided by regions of lower density. Therefore, for each point within a cluster, the neighborhood identified by a fixed radius (*epsilon*) must include at least a minimum number of points (*minimum points*) [95]. Any data point that has more neighbors than the minimum points is tagged as a cluster *core point*, whereas a sample is considered a *boundary point* if it has fewer neighbors than the minimum but belongs to the neighborhood of a core point.

DBSCAN operates through the following three phases:

- *First step*: for each point xi within the data set, the distance between that point and the other samples is determined. Then, all points within the epsilon radius are considered neighbors of xi, and those that have a number of neighbors greater than or equal to the minimum points are labeled as core points;
- *Second step*: if a center point is not already part of an existing cluster, it generates a new cluster. The algorithm will find all the density connected points in order to assign them to the same cluster;
- *Third step*: the algorithm iteratively analyzes the remaining unvisited points. If a point does not belong to any cluster it is treated as an outlier.

One of the main advantages of this clustering algorithm is that the number of clusters to be generated is not required preliminarily. In addition, a second important aspect is DBSCAN's ability to process clusters of any shape and independently identify outliers.

On the negative side, DBSCAN does not work well when applied to clusters with different densities, and the user must carefully select the necessary parameters. In addition, the value of the minimum points increases as the size of the data set increases, and typically this value should be at least 3. To select the correct epsilon radius, the user can use the k-distance graph, in which the distance to the k minimum points is plotted against the value of k.

4.3.4 Dimensionality Reduction

Dimensionality reduction is one of most important sub-field of unsupervised learning. The dimensionality of a dataset can be understood as to the number of features needed to fully describe a given problem. For example, in a dataset in column-row format, dimensionality is the number of columns, whereas each row identifies a single training sample. The objective of an algorithm employed in dimensionality reduction is to minimize the number of features necessary to represent the problem, and consequently its complexity, by analyzing the hidden structures of a dataset.

It is worth noting that dimensionality reduction drastically differs from the process of feature selection. Although both methods seek to reduce the number of features in the dataset, dimensionality reduction succeeds in this task through the creation of new combinations of features (i.e., the extraction of new variables), while feature selection attempts to include and/or exclude some features that already exist in the dataset without generating new ones.

4.3.4.1 Principal Components Analysis

The *Principal Components Analysis* (PCA) consists of a multivariate statistical method that reduces the dimensionality of the data while preserving most of the variation in the data set [96]. An orthogonal transformation allows a set of n observations, possibly including k correlated variables, to be converted into a new set of values of linearly uncorrelated variables, called *principal components*, which are linear combinations of the initial k variables [97].

The first principal component represents the maximum possible variance in the data and extends in the direction along which the sample points show the greatest variation. Accordingly, the first component can be considered as the line of best fitting between sample points. The determination of the best fit is accomplished by an algorithm that minimizes the distances between the various points and the line itself, typically Lapack's Singular Value Decomposition (SVD) or, especially for redundant data sets, the Nonlinear Partial Least Squares Algorithm (NIPALS) is employed [98].

The second principal component, as can be seen from Figure 4.13, is calculated with the constraint of being orthogonal to the first one and having the largest possible variance. All subsequent principal components are then constructed using the same criteria. It is also important to note that the amount of explained variance increases with the number of principal components taken into consideration; a scree plot showing the explained variance versus the number of principal components can be used to select the appropriate number of those components. PCA is technically a decomposition of data matrix X into two matrices T and P:

$$\begin{aligned} X &= TP^{\mathrm{T}} \\ (n \times k) &= (n \times a)(a \times k) \end{aligned} \tag{4.34}$$

where T and P are the score matrix and load matrix, respectively, as well as orthogonal matrices. In addition, k is the number of features of original set, n is the number of samples and $(a<k)$ is the number of the considered principal components.

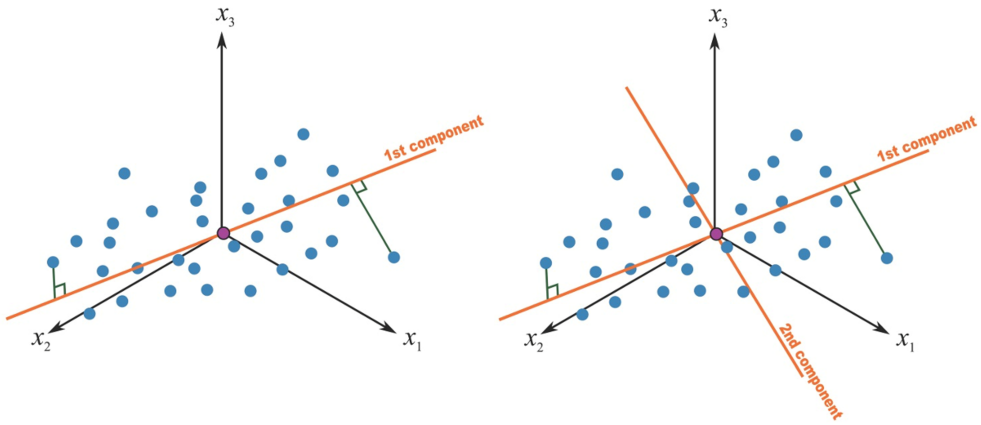


Figure 4.13. Geometric interpretation of PCA.

The loadings are the weights of each original feature in the principal component calculation; their values quantify the statistical importance of the features for the variability of the dataset. Highly correlated features have almost the same weights in the loadings vectors and appear next to each other in the loadings graphs, consisting of a graph of the loadings of one principal component versus the loadings of another component.

The scores represent the original data in a rotated coordinate system, i.e., the principal component space, and consist of the extracted features that can describe the original dataset within a new lower dimensional space. In this way, it is possible to represent the original dataset with a reduced number of new features.

4.3.4.2 Projection to Latent Structures

The *Projection on Latent Structures* (PLS) acts on the data similarly to PCA, but with one main difference: in this case, unlike PCA, the variables of the original dataset are divided into input features and output features, so the analysis aims to find relationships between these two groups. In practice, PLS will define the multidimensional direction in the space of input features that defines the maximum multidimensional variance in the space of output features [99]. The PLS in its general form builds latent vectors (i.e., orthogonal score vectors) by maximizing the covariance between these different sets of variables [100, 101]. The results for the input and output features are a load vector and a score vector, respectively:

$$\begin{aligned} t_{\mathrm{a}} &= X_{\mathrm{a}} w_{\mathrm{a}} \\ u_{\mathrm{a}} &= Y_{\mathrm{a}} c_{\mathrm{a}} \end{aligned} \tag{4.35}$$

in this set of equations t_a and w_a represent the score vector and the loading vector for input features X, whereas u_a and c_a are the score vector and the loading vector for output features Y.

As illustrated in Figure 4.14, the PLS model iteratively calculates loads and scores using Nipals' algorithm, which works as follows:

- *First step*: the algorithm regresses each column of X_a on the vector u_a. Initially, u_a is estimated by a column from Y_a. The coefficients of this regression are stored in $w_a = X'_a u_a(\frac{1}{u'_a u_a})$. The columns highly correlated with u_a have the largest weights in w_a.
- *Second step*: the weight vector is normalized: $w_a = \frac{w_a}{\sqrt{w'_a w_a}}$.
- *Third step*: each row of X_a is regressed onto w_a and the regression coefficients are stored in $t_a = X_a w_a(\frac{1}{w'_a w_a})$.
- *Fourth step*: Nipals' algorithm regresses each column of Y_a onto the score vector t_a and stores the regression coefficient in $c_a = Y'_a t_a(\frac{1}{t'_a t_a})$.
- *Fifth step*: each row of Y_a is regressed onto c_a, in this way $u_a = Y'_a c_a(\frac{1}{c'_a c_a})$ is computed for the first time.

This process is repeated until the vector u_a reaches convergence values.

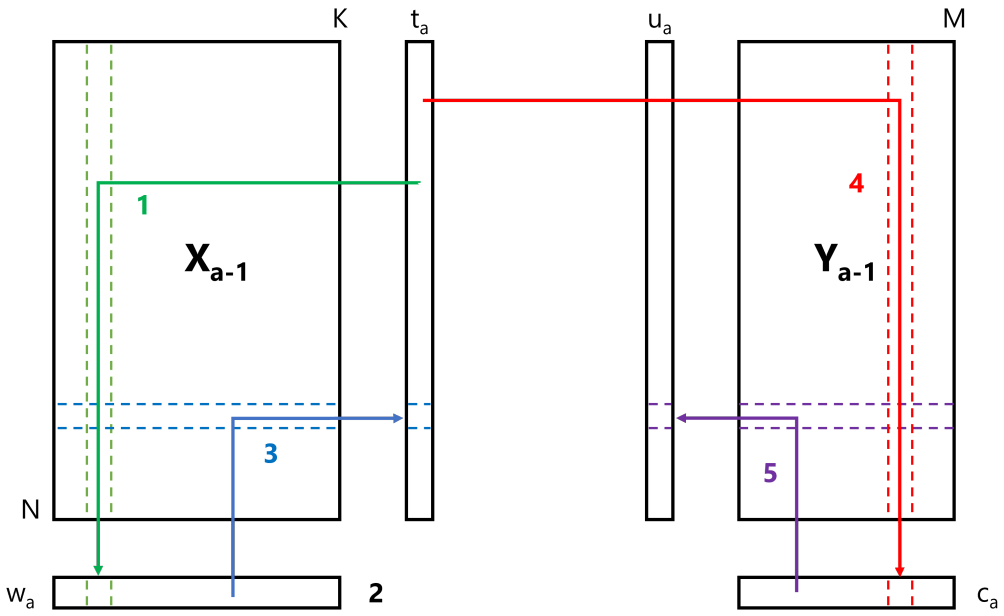


Figure 4.14. Nipals algorithm for the calculation of PLS.

After building a PLS model, the most informative result lies in the load graphs, where there is the superposition of a graph of input feature loadings to a graph of output feature loadings. This graph shows the relationship between latent variables in input feature space and output feature space, as well as the relationship between input and output variables.

4.3.4.3 Autoencoders

An *autoencoder* (AE) is a data compression algorithm (specifically, a special type of neural network) trained to copy its input to its output, otherwise to learn a compressed representation of a set of data [102]. For example, given an image

(Figure 4.15) of a handwritten digit, an AE first encodes the image into a lower dimensional *latent representation*, then decodes the latent representation back to an image. An AE learns to compress the data while minimizing the reconstruction error and consists of two different parts, namely an *Encoder* and a *Decoder*.

On one hand, the Encoder maps the input $x \in \mathbb{R}^{\mathrm{n}}$ to a *latent space* and the output ${h_{\mathrm{e}}}^{(l+1)}$ of the *l-th* layer can be written as:

$$ {h_{\mathrm{e}}}^{(l+1)} = \sigma({W_{\mathrm{e}}}^{(l)} \cdot {h_{\mathrm{e}}}^{(l)}) \tag{4.36} $$

where ${W_e}^{(l)}$ are the trainable weights of the layer and σ is a non-linear activation function. If the Encoder has L_e layers, then ${h_{\mathrm{e}}}^{(0)} = 0$ and ${h_{\mathrm{e}}}^{(L_e)} = h$, where $h \in \mathbb{R}^{\mathrm{k}}$ is the compressed representation of the input.

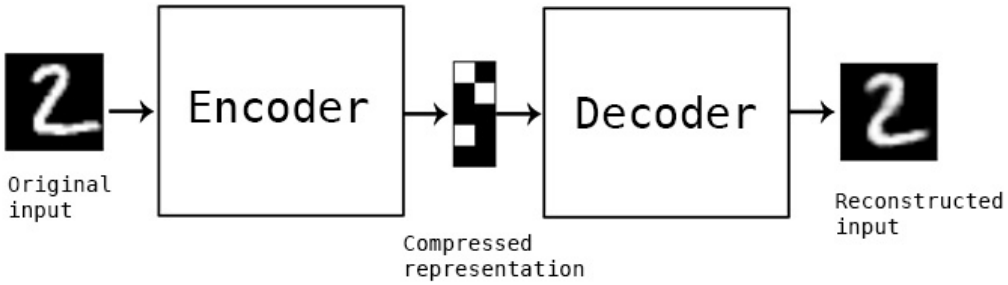


Figure 4.15. General architecture of an autoencoder.

On the other hand, the Decoder maps the compressed representation h back to its original space and therefore:

$$ {h_{\mathrm{d}}}^{(l+1)} = \sigma({W_{\mathrm{d}}}^{(l)} \cdot {h_{\mathrm{d}}}^{(l)}) \tag{4.37} $$

where ${h_d}^{(l+1)}$ is the output of the *l-th* layer, W_d is the trainable weight matrix and σ is still a non-linear activation function. Considering L_d layers, then ${h_{\mathrm{d}}}^{(0)} = h$ and $h^{(L_d)} = x'$, where x' is the reconstruction of the vector x.

Since the goal of an AE is to reconstruct the input as accurately as possible, ideally $x' \approx x$, it is trained by minimizing the reconstruction error $\mathcal{L}(x', x) = \|x' - x\|$, also referred as loss function, through the Backpropagation algorithm [103].

4.4 Model Evaluation and Optimization

The crucial stage of machine learning is the training of the model. Indeed, it is at this stage that it is necessary to assess how well the model fits the dataset and, above all, how well the trained model is able to generalize with respect to unseen data. Generalization is the ability of a trained model to produce sensible and reliable results when applied to input sets never seen before.

During the training phase, the prediction capability of the model must be incrementally improved. For this purpose, the original data set is divided into a training set and a validation set; the algorithm is trained using samples from the former and tested on the latter to assess its accuracy.

4.4.1 Training and Validation

The simplest method for evaluating a predictive machine learning model is the *holdout* method, which randomly divides the original dataset into three subsets: training set, validation set and test set. The predictive model is trained using the samples from the training set. The validation set is used for testing purposes and for tuning the hyperparameters of the model. Finally, the test set, which includes the never-seen data, is used to judge the future performance of a model. The results obtained during this testing phase define the accuracy of the model. Although this method is easily implementable and has high flexibility, it is characterized by high variability, as different training and test sets can result in significantly different models.

A second validation method used to evaluate the performance of a predictive model is the *k-fold cross-validation*, which divides the original dataset into k different folds of the same size [104]. As shown in 4.16, a single fold constitutes the validation set, while the model is trained on the remaining *k-1* folds; this procedure is repeated k times yielding k different training and testing sets.

The accuracy is, then, estimated by averaging the results obtained for all training-test pairs:

$$\widehat{y} = \frac{1}{k} \sum_{i=1}^{k} \widehat{y} \tag{4.38}$$

Figure 4.16. *k-th* cross validation.

Cross-validation using most of the data for the training phase significantly reduces bias, and since all data are also used in the testing phase, it also reduces variance. In addition, cross-exchange of test and training samples increases the efficiency and robustness of this validation strategy, but having to train and validate k different models might be computationally expensive.

The goal of a training-validation process is to obtain a model that performs optimally when applied to predict data never seen before. This means avoiding, during the training phase, *overfitting* of the data set, which is a phenomenon that causes loss of information.

4.4.2 Overfitting and Underfitting

In any machine learning problem, it is inevitable to deal with the problem of overfitting and underfitting. The key is to find the right trade-off between optimization, which refers to adjusting model hyper-parameters to achieve the best performance on the training set, and generalization, which refers to the model's ability to perform well on new data.

At the beginning of the training phase, the model is in an underfitting condition and thus still needs to improve its prediction. As shown in Figure 4.17, by iteratively reducing the loss on the training set, the loss on the test set also decreases. In this case, the model has high bias and low variance: the model is unable to correctly capture relationships and correlations among the data and has a difference in fit between the different data sets.

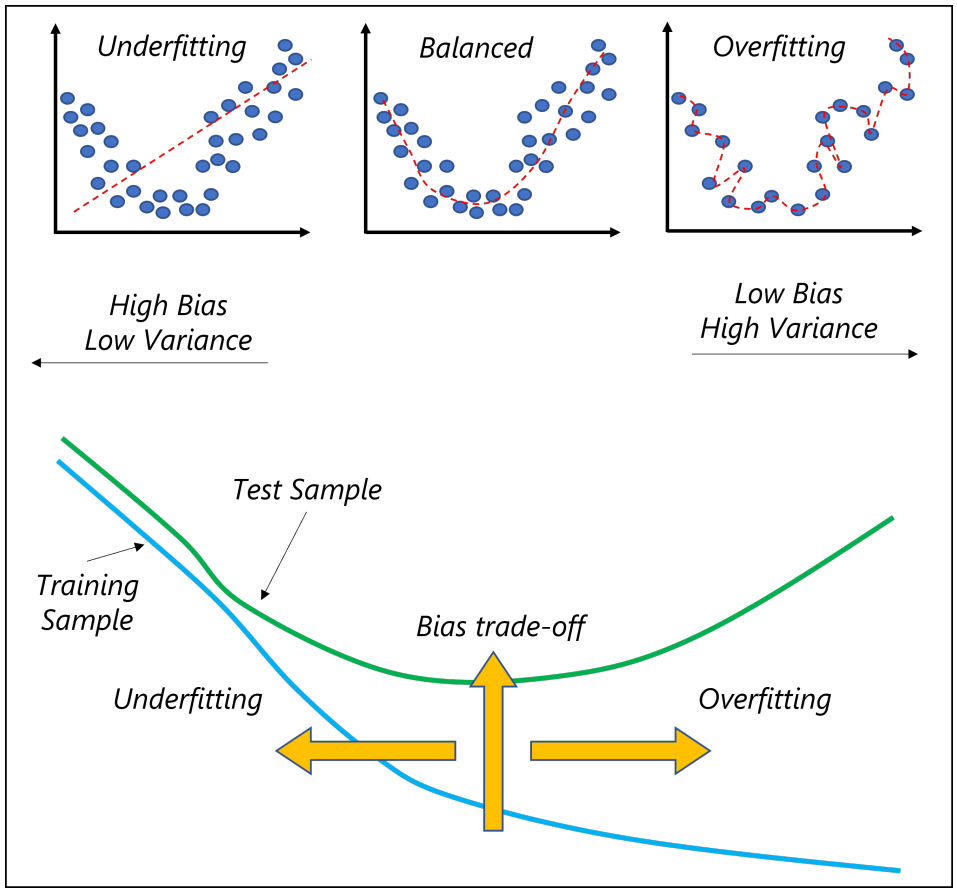


Figure 4.17. Bias-Variance trade-off.

After a few iterations, the test loss stops improving. At this stage, overfitting begins: the model learns overly specific relationships and correlations in the training set, which are misleading for the prediction of new data. Under overfitting conditions, the model has low bias and high variance: the model fits the training data perfectly, but has high prediction error when applied to the test dataset. To improve the accuracy and generalization ability of a model, it is necessary to expand the training

dataset. If this is not possible, it would be necessary to regularize the model by modulating the amount of information it is able to store according to the process called *bias-variance trade off*, which allows a model with limited memory to focus only on the most important patterns, thereby increasing its generalization capabilities.

Chapter 5

Development of Machine Learning assisted tools and framework for Digital Twin

This chapter describes three different applications of artificial intelligence-assisted tools for the development of Digital Twin models in both energy systems and turbomachinery. The three works will address issues of:

- *Turbomachinery Design via CFD*: for the identification and quantification of deflection by means of unsupervised learning for a cascade with sinusoidal leading edges;
- *Turbomachinery Design via Experimental Campaign*: for the high-fidelity compression of large experimental dataset from HPT with unsupervised learning;
- *Energy Systems Operation Monitoring*: for the condition based maintenance of gensets with a machine learning framework.

5.1 Cascade with Sinusoidal Leading Edges: Identification and Quantification of Deflection with Unsupervised Learning

One of the key issues in turbomachinery design is the identification of loss mechanisms and their quantification, both during preliminary design and in all subsequent optimization loops.

Over the years, many correlations have been proposed, accounting for different dissipative mechanisms that occur in blade-to-blade passages, such as the development of boundary layers, turbulent wake mixing, shockwaves, and secondary flows or off-design incidence. In recent years, the fan industry started the production of more complex rotor geometries, characterized by sinusoidal leading and trailing edges, mostly to extend stall margin and to reduce noise emissions. But literature still lacks a quantification of the losses introduced by the secondary motions released by serrated leading edges.

In this work a design of experiments that entails 76 cases of a 3D flow cascade with NACA 4digit profiles with sinusoidal leading edges is investigated to measure losses according to Lieblein's approach. The flow field simulated with RANS strategy was investigated using an unsupervised machine learning strategy to classify and isolate the turbulent wake downstream of the cascade with a combination of Principal Component Analysis and Gaussian Mixture clustering. Then a gradient boosting regressor was used to derive the correlation between input parameters and cascade deflection.

5.1.1 Introduction

In recent years, fan designers were pushed to more complex solutions to meet fan efficiency grades introduced by EU and US legislations. Leading and trailing edge serrations were among the possible strategies adopted to control separation dynamics and noise [105, 106].

Developed from biomimicry of humpback whales [107], leading edge serrations proved able to control the separation of the flow at the trailing edge of the blade and thus to alter the stall dynamics of the rotor aerodynamic profiles. In particular, scholars noted that the vorticity shed from the leading edge allowed the limitation of separation at the trailing edge only to the portion of the suction surface corresponding to the leading-edge throat, assumed at the point of lowest chord length. An example of this finding is shown in Figure 5.1, where the regions of recirculating flow are shown with red iso-surfaces over the suction side of the cascade profile. At an incidence i=18.3° the flow is still attached over the portion of the surface that corresponds to the leading-edge sinusoid peak.

Figure 5.1. Iso-surface of recirculating flow over WHALE4412-4-10 cascade.

Other findings show that while the amplitude of the serrations was directly

correlated to the capability of the leading-edge serration to control separation and increase lift at high angles of attack, the influence of its wavelength on aerodynamic efficiency is almost negligible.

When used in fan rotors, leading edge serrations proved to be able to alter the stall dynamics of the fan flattening the pressure curve in stalled operations, limiting the drop of pressure by altering the dynamics of the tip leakage vortex. In particular, the development of the tip leakage vortex in the blade-to-blade passage is partially blocked by the vorticity shed by the sinusoid at the leading edge of the rotor.

The acoustic performance of modified serrated leading edges was studied in different isolated airfoil and fan configurations [108, 109]. Chong et al. [110], studied a modified NACA 65(12)-10 airfoil with a serrated leading edge, concluding that the serrations can suppress the laminar separation bubble at the leading edge and the noise associated with that source. They also found that the changes induced in the growth of the turbulent boundary layer at the trailing edge suggest that a reduction in far field low-frequency noise can be achieved, whilst an increase of high-frequency noise is expected. Biedermann et al. [111] measured noise from airfoils with serrated geometries, concluding that serrations with high amplitudes and small wavelengths are the ones with better noise reduction capability.

Biedermann et al. [112] also derived a statistical-empirical model to predict the overall sound pressure level and noise reduction of a NACA 65(12)-10 airfoil with leading edge serrations.

In their works Becker et al. discussed the noise performance of axial flow fans fitted with serrated leading edges [113, 109]. An exploration of the noise performance of un-swept blades under clean and distorted inflow conditions led to the conclusion that serrations lead to better acoustic performance especially with clean inflow but affect the aerodynamic performance in the pre-stall region [113]. In [114] they compared the performance of different fans with leading edge serrations and swept blades, concluding that a forward-skewed fan with leading edge serrations had the highest efficiency and the lowest sound emission for nearly all operating points considered, achieving a sound reduction of between 3 and 7 dB at different operating conditions. Finally, in [109] the application of leading-edge serrations to flat plates was found to improve both aerodynamic and noise performance.

At the current state of the art of performance analysis of serrated leading-edge profiles there is a lack of information and correlations to extrapolate the deflection capability of a cascade as a function of the cascade parameters and sinusoid characterization of the leading edge. In the following a numerical procedure that allowed the computation of such performances in terms of deflection capability of the profile will be discussed.

5.1.2 Methodology

The proposed integrated and automated procedure allows to:

- setup a design of experiments (DoE) of modified NACA 4-digit airfoils (hereafter identified as “WHALE” airfoils) with sinusoidal leading-edges;

- compute the performance of the DoE;

- setting up an unsupervised machine-learnt algorithm to automatically identify the boundary layer and turbulent wake inside the computational domain;
- extract the wake velocity defect distribution;
- map the deflection capability of the cascade using a regression method on the DoE.

5.1.2.1 Leading edge modification

In the following, the performance of a series of modified WHALE airfoils will be discussed. A WHALE XYZZ-A-Λ airfoil is derived from a NACA XYZZ airfoil as a sinusoidal profile with the same average chord of the original and given sinusoid amplitude A and wavelength Λ , where A and Λ are given as a percentage of the chord. Thus, a WHALE 5412-4-16, Figure 5.2, is an airfoil with sinusoidal leading edge characterized by an amplitude A =4% of the chord and wavelength Λ =16% of the chord. This profile is the interpolation of NACA 5412 profiles with 1m average chord, maximum chord equal to 1.04m and minimum chord equal to 0.96m. All the profiles' trailing edges were aligned to achieve a straight trailing edge.

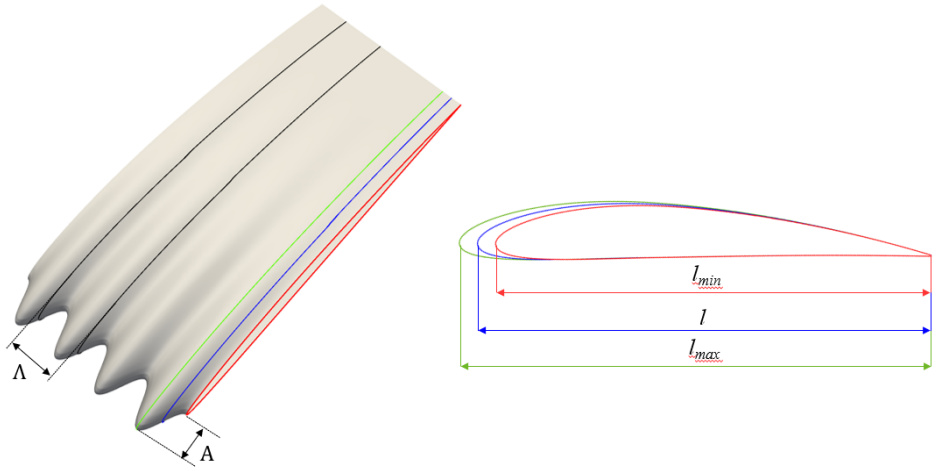


Figure 5.2. Whale 5412-4-16 airfoil geometry.

5.1.2.2 Cascade Design of Experiment

In this work the baseline NACA profiles is a priori selected as those of the 4-digit family X412, so that the distance of maximum camber from the leading edge was fixed to 40% and maximum thickness was set to 12% of the chord. The geometry of the cascade was therefore selected changing the 1st digit of the NACA airfoil, the amplitude A and wavelength Λ of the sinusoidal profile, the pitch (γ) and solidity (σ) of the cascade. Operating conditions were changed by adjusting the inlet velocity angle β_1 at a constant Reynolds number *Re=1.1M*.

The possible range of variation of each parameter is summarized in Table 5.1. The possible combinations of all the variations of input parameters are 86,000. To reduce

the number of experiments a CCDoE (Central Composite Design of Experiment) approach was selected, and the number of simulations was reduced to 76. This strategy was selected according to the same criteria adopted in [69]. Variability of input parameters is summarized in Table 5.1 while a sketch of the cascade geometry is given in Figure 5.3.

Table 5.1. Design of Experiment parameters.

	min	**max**	**step**	**unit**
airfoil parameters				
NACA 1st	1	6	1	[-]
sinusoidal leading-edge parameters				
Amplitude - A	4	16	2	[% of chord]
Wavelength - Λ	4	16	2	[% of chord]
cascade parameters				
Pitch - γ	15	60	5	[°]
Solidity - σ	0.32	1.32	0.2	[-]
β_1	-4	13	1	[°]

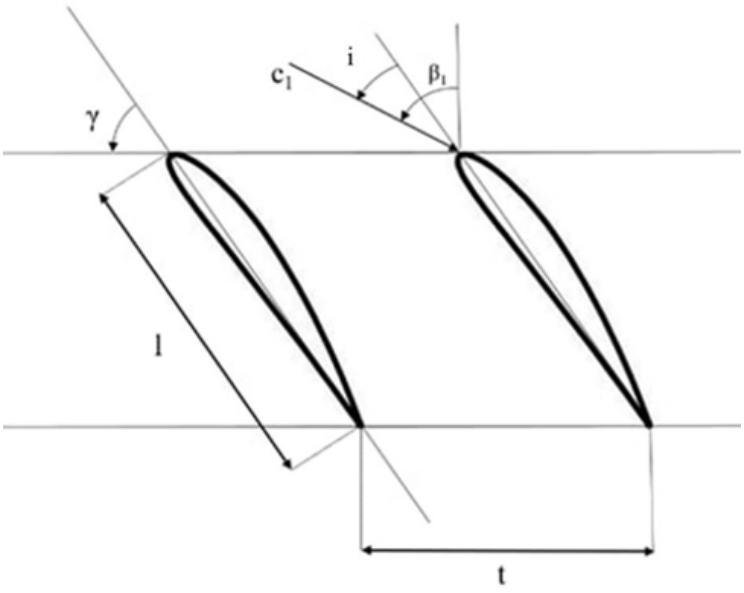


Figure 5.3. Reference cascade geometry.

5.1.2.3 Numerical setup

Numerical computations were carried out using OpenFOAM v-18.12 [10]. The steady-state solution of the Reynolds Averaged Navier-Stokes equations with the Spalart-Allmaras model [47] was computed using the *simpleFoam* solver with 2nd order discretization for steady computations of incompressible flows. Convective

terms were discretized using central differences for velocity and upwind scheme for turbulent quantities. The linearized system of equation was solved using, *GAMG solver* for pressure equation and *smoothSolver* for velocity and $\tilde{\nu}$.

The computational domain entails a blade-to-blade passage with periodic conditions in the pitch-wise direction. In the span-wise direction the domain covers half of the sinusoid at the leading edge with symmetry conditions, as shown in Figure 5.4.

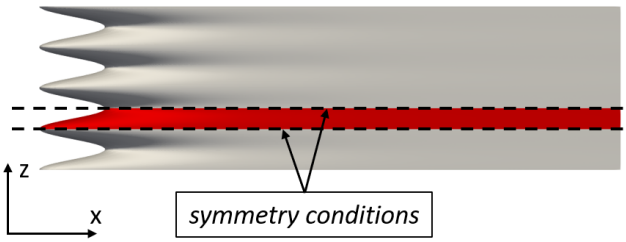


Figure 5.4. Blade geometry in spanwise direction.

In stream-wise direction the computational domain extends 10 axial chords up-stream of the leading edge and 30 axial chords downstream of the trailing edge. These dimensions were validated against grid convergence starting from the full sinusoid profile without symmetry conditions and 30 axial chords up-stream of the leading edge and 60 axial chords down-stream of the trailing edge [69]. Grid resolution depends upon wavelength, pitch and solidity and spans between 3M and 20M cells. For all the simulations the wall distance of the first cell is set to achieve a y+ $<$ 1.5, the maximum value obtained is y+ = 1.422.

Grid refinement around the airfoil was selected according to [69]. A grid-independency analysis was carried out using different refinements generated through the *snappyHexMesh* utility. A blowup of the grid around the airfoil is shown in Figure 5.5. The tolerance of convergence was set to 10^{-3} for pressure and to 10^{-5} for velocity and turbulence equations. Final tolerance of the simulation was checked against convergence of C_L, C_D and δ. Simulation time is between 15000 and 90000 seconds.

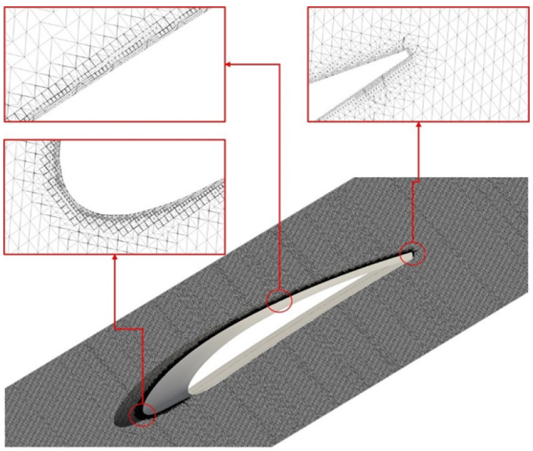


Figure 5.5. Grid around the airfoil.

5.1.2.4 Data analysis and preprocessing

Machine Learning techniques rely on statistical analysis and are therefore strongly influenced by the distributions and quality of data; in fact, they are seriously impaired if the data shows large variations in magnitude. To avoid this, a normalization of the original CFD data is required. In accordance with [70], a local normalization is performed on each feature which are defined as any observable characteristic of the flow:

$$F' = \frac{|f|}{|f| + |N|} \tag{5.1}$$

where F' is the new normalized feature, f is the original feature, and N is the normalization factor listed in Table 5.2, with U_{ref} equal to the average inlet velocity and $L_{\text{ref}} = 4U/S$, where S is the magnitude of symmetric strain rate tensor (while W is the magnitude of asymmetric strain rate tensor).

Table 5.2. Features Normalization.

Variables	**N**
U	U_{ref}
p	${U^2}_{\text{ref}}$
ν_{t}/ν	$U_{\text{ref}} \cdot L_{\text{ref}}$
$\tilde{\nu}/\nu$	$U_{\text{ref}} \cdot L_{\text{ref}}$
S, W	$U_{\text{ref}}/L_{\text{ref}}$
ω	$U_{\text{ref}}/L_{\text{ref}}$
$\nabla(U)$	$U_{\text{ref}}/L_{\text{ref}}$
$\nabla(p)$	${U^2}_{\text{ref}}/L_{\text{ref}}$
$\nabla(\nu_{\text{t}})$	U_{ref}
$\nabla(\tilde{\nu})$	U_{ref}
DF	[-]
L_{RANS}	L_{ref}

This normalization technique, based on the properties of the local flow field, has been previously tested in [115] and the positive results prove a better response of the algorithm to this approach compared to other usual normalization techniques in Machine Learning.

5.1.2.5 Unsupervised learning methodology: PCA and Gaussian Mixture clustering

For this work, an unsupervised ML method was developed, as a combination of features reduction and clustering. First, the correlation matrix (Figure 5.6) of the features was analyzed to select the most suitable to predict flow deflection. This resulted in the list of features in Table 5.2 that correspond to 28 scalars. As

the matrix shows a high number of uncorrelated features, Principal Component Analysis (PCA) was used to reduce their number by projecting them on a latent space according to dataset variance [92]. In this way, for each simulation the number of principal components NPC was selected according to the elbow chart, Figure 5.7, as the minimum NPC required to achieve a 95% of variance (for all the cases in DoE $9 \leq NPC \leq 12$).

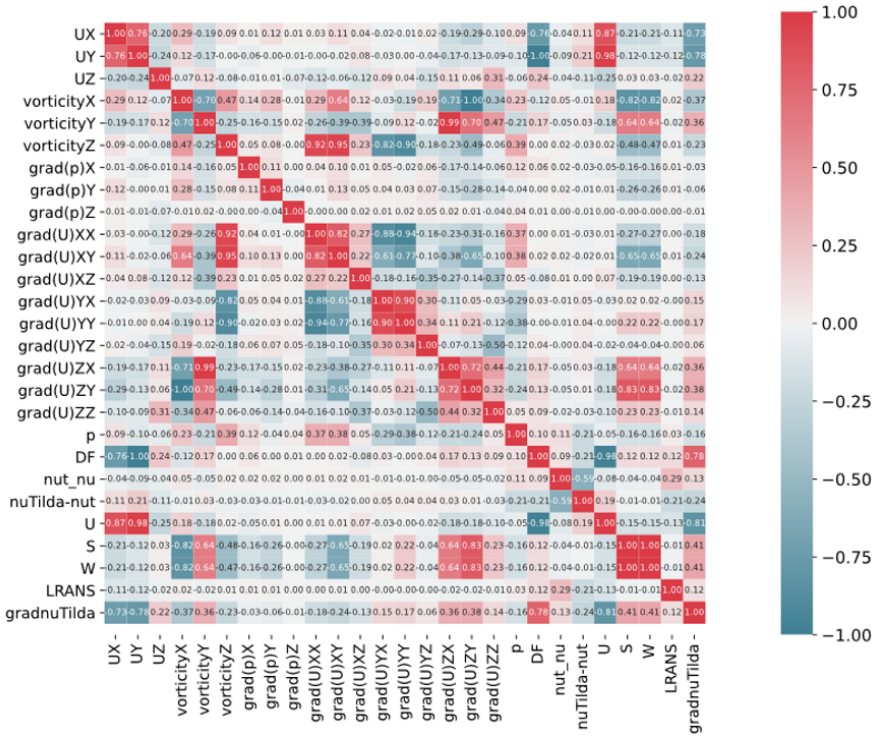


Figure 5.6. Features correlation matrix.

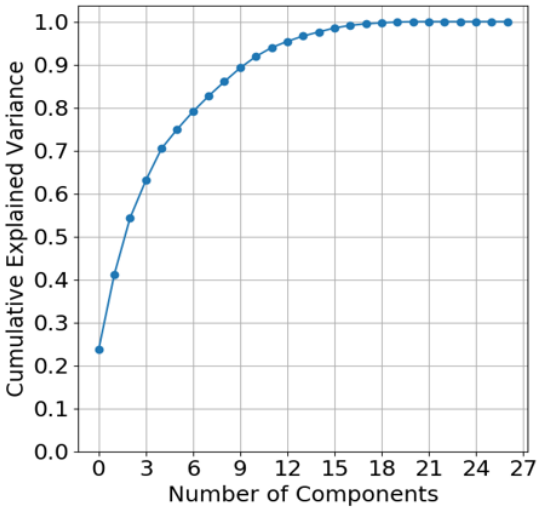


Figure 5.7. PCA elbow chart.

Vortical structures in the flow field were identified using an unsupervised clustering approach. In this case Gaussian Mixture (GM) was selected [92] among three possible choices: k-Means, GM, and DB-SCAN. Among these, k-Means is the fastest and more scalable, but even after hyperparameter tuning it was not possible to achieve stable results over the complete dataset. On the contrary, both GM and DB-SCAN performed well once tuned, yet GM was selected as it was found to be much faster and scalable on parallel computations (DB-SCAN requires more than 20 times the time required by GM and unlike GM it is not properly parallelizable).

After hyperparameter tuning, GM was run on the NPC features. The number of mixture components for the algorithm was selected according to Bayesian Information Criterion (BIC) [94]. An example of a BIC plot is given in Figure 5.8. All the cases in DoE resulted in several clusters between $8 \leq NC \leq 10$. The cluster that entails the largest number of individuals represents the freestream flow field. Such result is coherent with the scope of the work, i.e. to build an automated procedure to identify and remove it from the rest of the data. Therefore, it will be hereafter removed from visualizations for clarity's sake. The reader can assume it as NC=0.

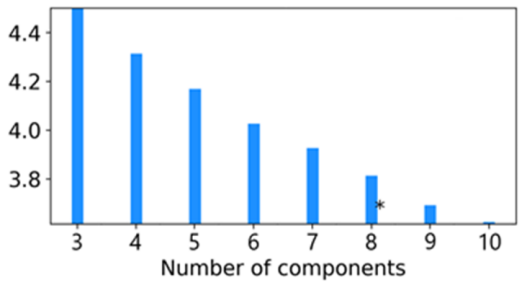


Figure 5.8. Whale 5412-4-16, BIC score per model.

5.1.3 Identification of turbulent regions with GM clustering

GM clustering was capable of recognizing the vortical structures belonging to boundary layer and turbulent wake. Figure 5.9 shows the clusters for three different cascade geometries with different leading-edge profiles and incidence. First, turbulent regions are identified as different boundary layer and wake regions. In particular one cluster belongs to pressure side boundary layer, while two clusters are associated with the turbulent wake according to the extent of separation. One cluster identifies the impinging region on the leading edge and three the different regions of the boundary layer over the suction surface. Two more clusters are due to the three-dimensional structures released by the sinusoidal leading edge. All are recognizable for the 76 cases in the DoE and partially shown in Figure 5.9.

To improve the pattern recognition, in Figure 5.10 these clusters are shown separately in 3D for one of the cases. NC 1, 2 and 3 identify different regions of the boundary layer developing over the suction surface. In particular, NC=1 corresponds to the region of the boundary layer that is more attached to the suction surface. NC=2 is the portion in the unmodified part of the airfoil and NC=3 is the portion on the modified leading edge. NC=4 shows the wake released at the trailing edge (purple structures continue downstream but are cut to have consistent views), NC=5 and 6 include the turbulent structures released by the leading-edge bumps. NC=7 and 8 are the impinging regions corresponding to the leading-edge troughs and peaks respectively and finally NC=9 the boundary layer on the pressure surface (visibility is limited from the view to stay consistent with other figures).

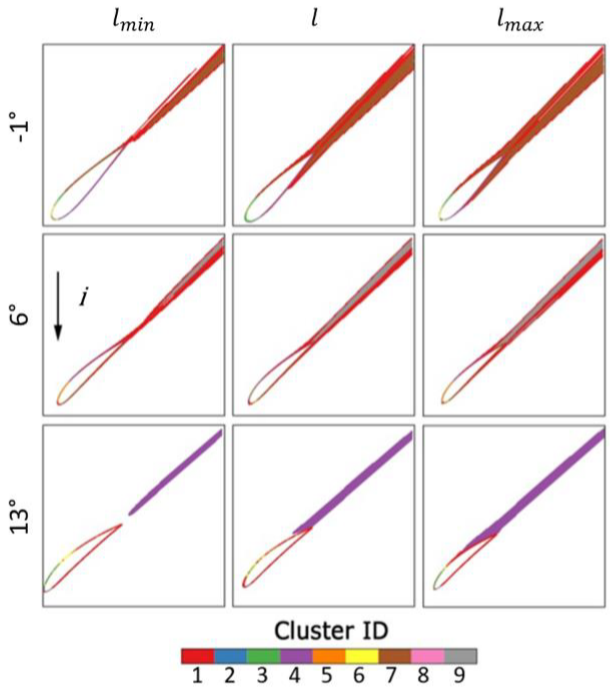


Figure 5.9. Clustering at different referent sections.

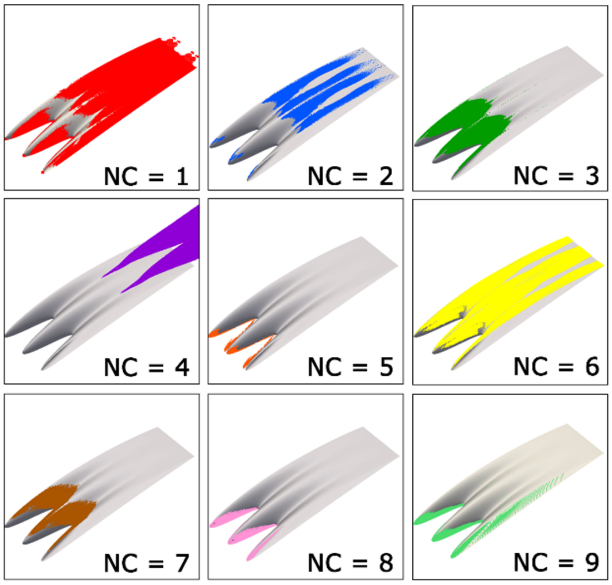


Figure 5.10. Clusters for WHALE 5412-13-13, γ=53.4°, σ=1.174, i=13°.

5.1.4 Metamodel for regression of deflection

Following the separation of boundary layer and wake, it was possible to compute losses and thus to build a meta-model that predicts the deflection of whale cascades within the range of the DoE as shown in Table 5.1. Using the following input features: NACA 1st digit (maximum airfoil camber), A (sinusoidal amplitude), Λ (sinusoidal wavelength), γ (pitch), σ (solidity), i (incidence), the meta-model was trained to predict deflection δ, using RMSE as accuracy score.

Three different regression models were tested: Random Forest [116], Support Vector Machine (SVM) [116] and XGBOOST [84]. The first two models from the sklearn Python library [117], the latter from the XGBOOST Python library [84]. To achieve a higher generalization capability of the algorithm, the hyperparameters of the model were optimized using a grid search process [118]. In so doing, the best combination of hyperparameters was chosen among an initial selection of hypothetical parameters. Such selection was aimed to obtain the maximum cross-validation accuracy. After tuning of hyperparameters the one that was more resilient to outliers proved to be XGBOOST [119]. Data from DoE were divided into a training set with 70% of data and a test set with 30%. The algorithm shows a high accuracy in both the training and test datasets, equal to 97.6 and 93.6% respectively (Figure 5.11).

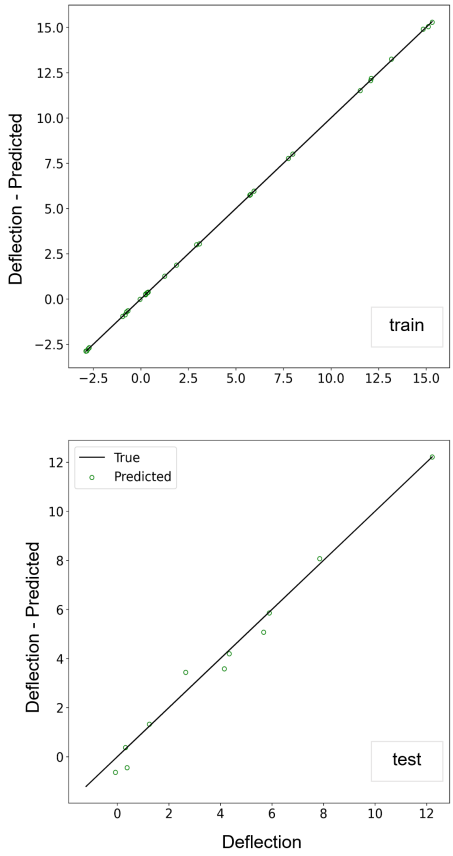


Figure 5.11. Accuracy of the meta-model for deflection during the training and testing phases.

The trained meta-model allowed to fully map the DoE between extreme values of input parameters as per Table 5.1, thus building a 6-dimensional hyperspace of solutions to predict the correlation between incidence and deflection.

A partial graphical representation of the incidence-deflection relationships in the DoE of serrated leading edges is shown in Figure 5.12 and Figure 5.13. In particular, Figure 5.12 reports three different plots, showing how deflection between 0° and 10° can be achieved with A=4%, 8% or 12%. The plots show the average value of Λ required to achieve each solidity and the standard deviation of Λ for each point as error bars. As incidence increases from -4° to 12°, increasing values of deflection are achieved. However, a peculiar trend is found as there are sudden increases at 2° and 9°. We can speculate that this is due to the limits of the DoE and that a smoother transition could be achieved by varying the position of maximum camber (here constant at 40% of the chord) following the results of [115]. Also, by increasing A , the average value of Λ increases as well for high incidence (red marks at $i \geq 9°$ for A=8% and A=12%). In Figure 5.13 three different plots are presented, that show how deflection between 0° and 10° can be achieved with Λ=4%, 8% or 12%. In this case similar figures apply to the discussion, yet standard deviation of points is on average larger than in Figure 5.12.

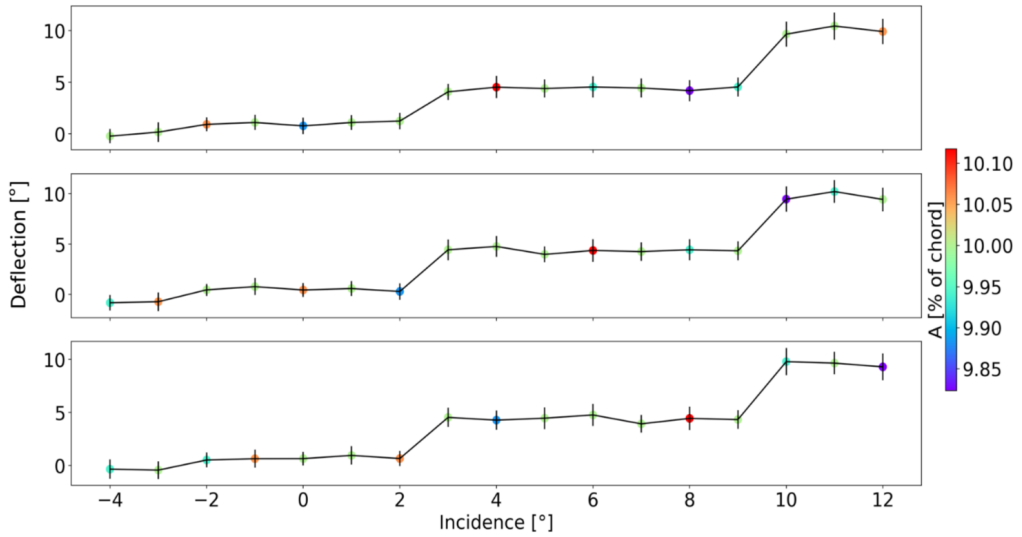


Figure 5.12. Deflection VS Incidence at A=4% (TOP), A=8% (MIDDLE) AND A=12% (BOTTOM). Marker color according to Λ.

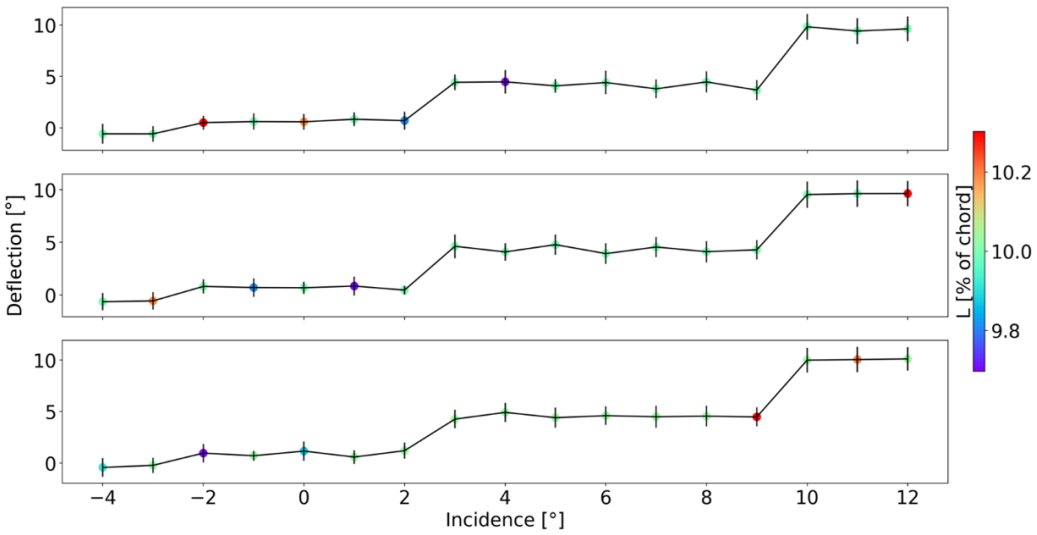


Figure 5.13. Deflection VS Incidence at Λ=4% (TOP), Λ=8% (MIDDLE) AND Λ=12% (BOTTOM). Marker color according to A.

5.1.5 Final remarks

In this work an integrated method to derive a meta-model for deflection prediction of a compressor cascade with modified sinusoidal leading edge is presented.

The method is based upon a series of 76 RANS computations of the flow field in the cascade used to build a dataset according to a Central Composite DoE. The method consists of unsupervised machine learning for feature reductions through PCA and GM clustering to identify the boundary layer and wake inside the computational domain and separating the far-field, the boundary layer on the airfoil and the

turbulent wake. While PCA was instrumental to speed-up the clustering algorithm reducing the required number of features, the critical part of the work is constituted by determining the clustering technique most suited for this task. GM was found as the optimal trade-off between the speed of k-Means and the accuracy of DB-SCAN.

The data preliminary sorted using the clustering algorithm was exploited to build a meta-model able to predict deflection of the cascade given its geometry, that of the leading edge serrations and incidence. This was done by selecting XGBOOST regression as it was found to be the most resilient against outliers in the original DoE. The metamodel was able to predict the deflection of the cascade with an accuracy of 93.67%.

5.2 Unsupervised Learning for high-fidelity compression of large experimental dataset: an application on HPT blade tip contouring

Rotor tip contouring is one of the most difficult and promising methods to enhance the performance of high-speed turbines (HPT). In this study, a new unsupervised machine learning approach based on dimensionality reduction is presented. By advocating methods borne in image recognition, this methodology explores the functional relationships between tip contouring geometry and aero-thermodynamic performance of HPT blades. The procedure is validated against experimental measurements carried out on the rotating turbine rig at von Karman Institute, which operates on a rotor with multiple tip geometries divided over 7 sectors.

5.2.1 Introduction

Industry 4.0 refers to a new industrial paradigm that encourages the use of Big Data (BD) and Artificial Intelligence (AI), as well as the creation of cyber-physical systems (CPS) [120]. Through vertical, horizontal, and end-to-end integration of industrial processes, CPS aims to establish a dynamic environment where smart technologies can exchange data, assess production issues, and boost productivity while reducing economic, time, and energy costs [121].

In the past decade, data from engineering processes have grown in scale, since CPS-related technologies are primarily based on sharing and analyzing data from an entire industrial production process [122]. The exploration of these unorganized data pools has demanded the development of BD tools, that aim to explore and reorder data collected from massive and heterogeneous resources [123]. With respect to traditional datasets, BD involve both structured and unstructured data and require more time and complex resources for in-depth analysis [124]. BD analytics are now strategically found in most engineering applications, including environmental monitoring [125], smart grids for renewable energy networks [126], manufacturing process optimization [127], turbulence research [55], identification and quantification of loss mechanisms during turbomachinery design processes [128], condition-based maintenance and predictive maintenance based on SCADA signals [129], health care applications [130] and face recognition [131].

The turbomachinery community is also reconsidering manufacturing processes in light of the exponential growth in the usage of smart technologies, particularly those that are connected to the creation of more intricate solutions to boost core engine efficiency [33]. To this end, the use of surrogate models to speed up prediction time and lower the overall number of numerical simulations and experimental campaigns needed to perform design optimization is the state-of-the-art in turbomachinery design at the time of writing [132]. However, standard approaches may fail in capturing the complexities of the underlying physics, as the modelling accuracy is limited by the number of features that are considered. As a result, designers frequently rely on empirical approaches and review of analysis graphs to develop design strategies because it is challenging or impracticable to explicitly describe these features in advance. Employing AI models that can take into account every feature

required for design optimization may constitute an answer. In fact, in recent years, the adoption of more sophisticated sensors and the growth in computing power have made both high-resolution numerical simulations and experimental measurements possible, making a vast quantity of data on turbomachinery performance available. As an example, Cernat et al. in their work [133] used an experimental campaign to assess the unsteady aerothermal field of various blade tip geometries, reporting measurements of static pressure and heat transfer on the turbine rotor shroud that were both time-averaged and time-resolved. Biedermann et al. have carried out another extensive experimental campaign [134] to highlight how sinusoidal leading edges of blades modify the acoustic signature of industrial fan rotors. Zhang et al. [135] experimentally investigated the trajectory and dynamics of the tip leakage vortex in axial flow pumps. Lavagnoli et al. [136] describe the full implementation of a high-frequency capacitive sensor on the shroud of a large transonic turbine stage. Consequently, today's challenge is to assess the capabilities and limits of BD approaches in turbomachinery performance analysis, with the goals of unveiling useful information and assist the design process. However, in the exploration of large datasets it is essential to reduce the number of features to a selection that has the maximum informative capability. This process is called dimensionality reduction that prevents dealing with a large amount of data and complicating AI models. Principal Components Analysis (PCA) and Variational Autoencoders (VAEs), a probabilistic form of Autoencoders, are the two main components of BD dimensionality reduction approaches (AEs). In his research [137], Zagler trained a VAE to simulate the compressor blade's supersonic airflow characteristics under various mass flow scenarios. Pongetti et al. [138] proved how AEs can be utilized to provide compressed form insights from the complete flow field in forecasts of quantities of interest, which can be used to improve the accuracy of AI models employed in high load transonic compressor blade optimization cycles. Angelini et al. [56] developed a dimensionality-reduced parametrization of complex tip geometries based on PCA scores for gas turbine rotors, overcoming the limitations of discrete topology approaches.

In similar multi-scale and multi-physic problems, it is difficult to find analytical modeling routes to correlate the geometric parametrization of the blade tip and the aerodynamic and thermal performance. Therefore, this paper proposes a novel and generalized Machine Learning (ML) methodology to exploit large experimental data sets using unsupervised dimensionality reduction tools in order to investigate the causal relationships between various tip geometries and unsteady aero-thermodynamic performance in HPT blading. The approach is validated using an experimental dataset measured at the rotating turbine rig at the von Karman Institute, which mounts a so-called rainbow rotor consisting of 48 blades separated into 7 sectors based on the similarity of their various geometries.

5.2.2 Methodology

This manuscript reports on the use of a novel unsupervised ML workflow that, advocating AI methods developed in the field of image recognition, aims at deriving functional relationships between tip contouring geometry and aero-thermodynamic performance of HPT blades rotor. The rotor features 48 rotor blades that share the

same baseline profile and are distributed among 7 distinct circumferential sectors. Each sector hosts six or seven blades with a specific geometry, including: single squealer, multi-cavity squealer or contoured profile. This testing method allows simultaneous measurements on different tip geometries. The unsupervised ML workflow is illustrated in Figure 5.14. According to such a description, the proposed method:

- exploits an experimental dataset including unsteady heat transfer and pressure measurements carried out on the casing of the HPT turbine test rig at the von Karman Institute when operating of a series of tip contouring geometries;
- entails data augmentation with up-sampling of both geometry and measurements;
- provides unsupervised dimensionality reduction by means of PCA on the tip geometries, and AEs on the performance data;
- trains a learner, based on a neural network (ANN), to predict the latent representation of thermo-fluid-dynamic data (performance) derived from the AE using the Principal Components of the tip geometries as the input of the ANN;
- finally decodes the latent representation and provides the corresponding performance maps. The accuracy of the results is evaluated against the original data using Structural-Similarity Index Measure (SSIM) as performance index [139].

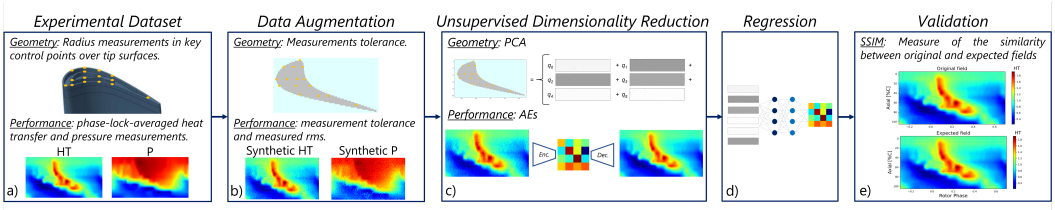


Figure 5.14. Unsupervised Machine Learning Method workflow.

5.2.2.1 Experimental dataset description

Figure 5.15 provides details of the tip designs that characterize the non-confidential sector geometries. Available data include both information on the reduced tip geometries and aero-thermodynamic measurements. The static pressure field above the tip is acquired through an instrumented insert equipped with 66 pressure taps on the casing: 33 fast-response piezo resistive pressure sensors and 33 pneumatic lines sampled by an external pressure scanner. The unsteady casing heat flux is measured by 35 double layer thin film gauges [133].

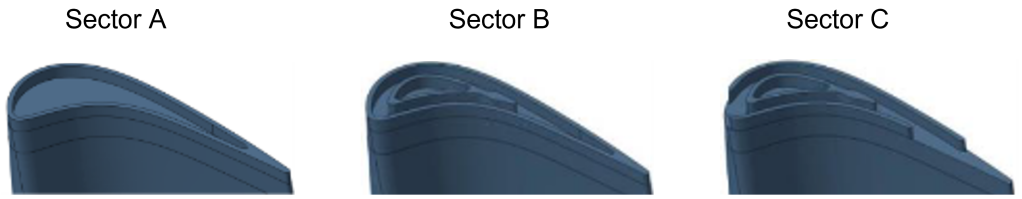


Figure 5.15. Rainbow rotor tip designs.

Data is mapped on two dimensional planes that can be also interpreted as images. Reduced tip geometry, as reported in Figure 5.16, refers to radial measurements at significant control locations on each of the 48 blades. Depending on the specific baseline profile, the control points on the tip surface can range from 10 to 16. The aero-thermodynamic performance is investigated using phase-lock-averaged (PLA) pressure (P) and heat transfer (HT) measurements for each of the 7 sectors (Figure 5.17).

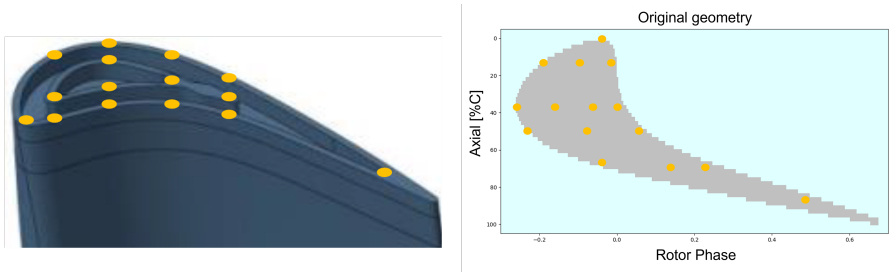


Figure 5.16. Key control points for reduced tip geometry of the B-sector baseline profile.

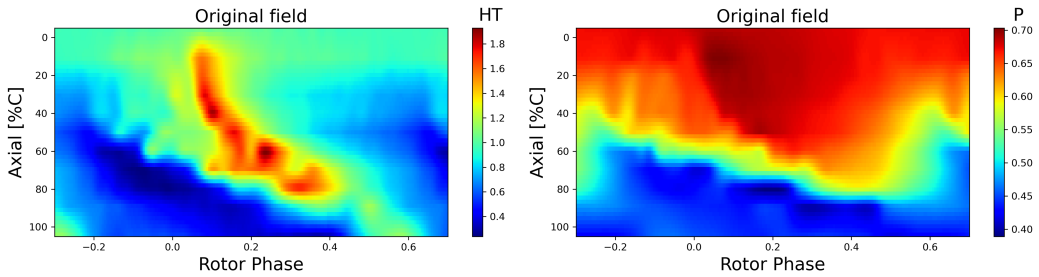


Figure 5.17. PLA measurements of heat transfer and pressure for C sector.

5.2.2.2 Data augmentation

The automatic detection and characterization of the blade tip shape within the geometry's point-cloud is the first step in the unsupervised machine learning process. This is accomplished by utilizing a Python function that locates the tip perimeter and assesses whether a grid point fits within it. The routine is based on the Concave Hull algorithm [140]. The task of the algorithm is to compute the envelope of a set of

points in a plane, which generates convex or nonconvex hulls that represent the area occupied by the given data points. The algorithm is based on a k-neighbor approach: the point with the lowest y-axis value is the current vertex, then among its k-nearest neighbors the one matching the largest right-hand turn from the horizontal line that includes the current vertex is selected as the next vertex, and so on. Accordingly, after identifying the envelope of the tip perimeter, the algorithm automatically assigns the points that are identified in this way either the corresponding radius (59 mm in the test case) or a radius of 0 mm if it lies outside the tip perimeter. Furthermore, if a point is coincident with one of the key points reported in Figure 5.16, it assigns the real radius measurement. In the second phase of the framework, synthetic data for both performance and geometry are generated to enrich the training dataset. Traditional ML approaches enrich training dataset with different operations on images, such as cropping, rotating and resizing. However, such approach is not feasible for fluid dynamic fields in a fixed reference system. Therefore, the dataset was upsampled using a gaussian distribution of both geometry and performance. The gaussian distributions are based on measurement tolerances for tip representation and heat transfer, and measured rms for pressure. As summarized in Table 5.3, the synthetic radius (R) is the sum of the actual radius ($\text{radius}_{\text{actual}}$) and its randomized measurement tolerance (r_1). The same applies to the synthetic heat transfer (HT), which is the sum of the real heat transfer ($\text{ht}_{\text{actual}}$) and its randomized measurament tolerance (ht_1), while the synthetic pressure (P) is the sum of the static pressure (p_{static}) and the randomized rms_{p} (p_1). The generated Synthetic database entails 5952 combinations of geometries and performance: 124 combinations for each original blade. Figure 5.18 provides a comparison between synthetic heat transfer and pressure data with the original data.

Table 5.3. Synthetic data generation criterion.

	Radius	**Heat Transfer**	**Pressure**
Random data generation	$r_1 = N \times 0.0005, N \in (-1, 1)$	$ht_1 = N \times 0.6, N \in (-1, 1)$	$p_1 = rms_{\text{p}} \times N, N \in (-1, 1)$
Synthetic Data	$R = radius_{\text{actual}} + 0r_1$	$HT = ht_{\text{actual}} + ht_1$	$P = p_{\text{static}} + p_1$

5.2.2.3 Unsupervised dimensionality reduction

PRINCIPAL COMPONENTS ANALYSIS

The synthetic tip geometries database is constituted by 5952 different images of possible tip characterizations with 11781 features, i.e., image pixels. The dataset is then processed through a principal component analysis projection (PCA) [141]. This decomposition deriving a dimensionally reduced (latent) representation of each synthetic geometry. This compression is achieved by projecting the data in a new coordinate system in which the basis vectors are the directions with maximum variance. The number of bases is chosen to limit the loss of information that inevitably is associated with the projection on latent space.

In the case of images, a continuous analogical datum can be represented as a matrix of pixels, which can then be combined column by column to create a 1D vector. The total number of pixels determines the dimension of the vector space, whereas the intensity variation of an image is expressed by the values of the pixels.

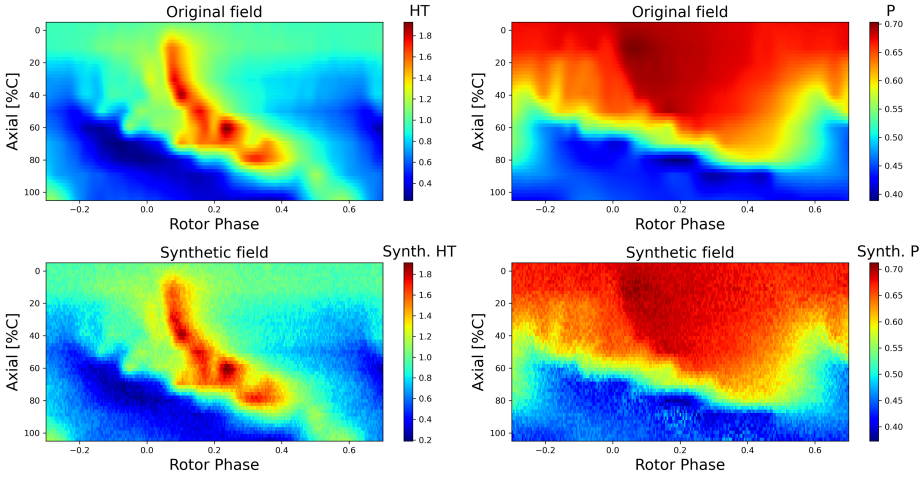


Figure 5.18. Synthetic and original performance fields comparison of 1 out of 124 pairs of combinations for the 1st blade in sector C.

Six principal components are required for the training dataset to reduce the information loss below 5%, as shown by the elbow diagram in Figure 5.19. The corresponding loadings, also known as eigen-images (eigen-blades) are displayed in Figure 5.20. In so doing, each database synthetic tip geometry can be obtained as a linear combination of the original eigen-blades, weighted by their own score.

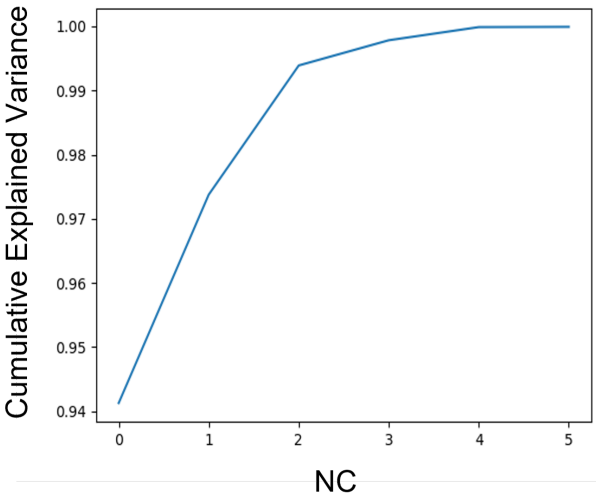


Figure 5.19. PCA elbow chart.

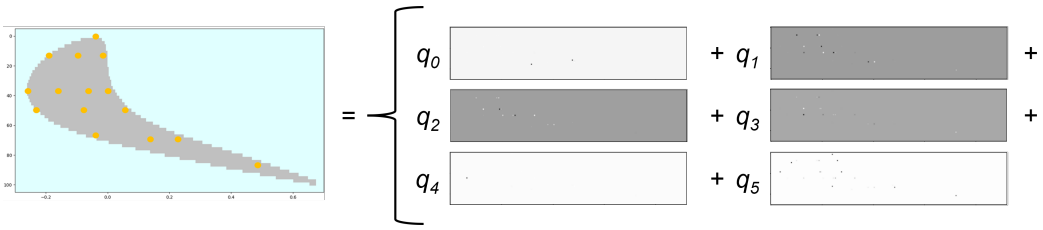


Figure 5.20. PCA graphical illustration.

AUTOENCODERS

Autoencoders (AEs) are a special class of neural network trained to learn a compressed representation of data in an unsupervised manner to copy its input to its output from a reduced encoding latent representation [102]. AEs consist of two parts: an encoder function *h=f(x)* trained to map the input x to its latent representation h, and a Decoder that produces a reconstruction function *(r=g(h))*, that maps the latent representation back to the original form. The compression-reconstruction process generates an error between reconstructed and original data. The minimization of this error is the objective function of AE training.

In the reference dataset, mean squared error (MSE) is chosen as reconstruction error metric. The encoding process is capable of derive a latent representation of HT and pressure with a dimensionality of 16 features. The decoder then maps back from the latent space to the original data. Figure 5.21 shows a pair of dimensionality reduction results by AE: the upper part of the image shows the encoding in latent representation and subsequent decoding of 1 synthetic heat transfer field out of 124 of the first blade of sector C, while the lower part shows the same procedure for 1 synthetic pressure field out 124 of the fourth blade of sector B.

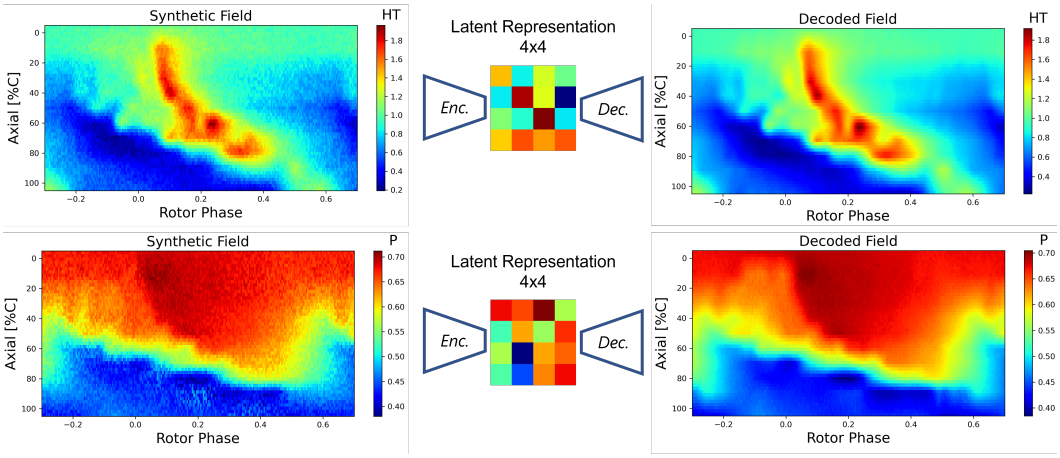


Figure 5.21. Encoding and decoding procedure through AEs.

5.2.2.4 Regression

The compressed data can be more easily exploited for turbomachinery design and optimization processes. For example, in this reference dataset it is possible to correlate the blade tip contouring with the flow fields. This is conducted in a

latent space, treating geometry PCA scores and the encoded features from AEs as input/output features. The relationship is here build exploiting an ANN, however different algorithms may be easily applied. Table 5.4 details the final setup of the MLP model optimized through a grid search algorithm [118], considering early stopping to avoid overfitting.

Table 5.4. Regression.

Number of hidden layers	2
Number of neurons	32
Number of training epochs	1000
Activation function	relu
Initial learning rate	1E-3
Optimizer	ADAM
Cost function	MSE
Batch size	250

Figure 5.22 show a schematic view of uses of the compressed data, with the creation of a functional relationships between geometry and pressure/HT fields.

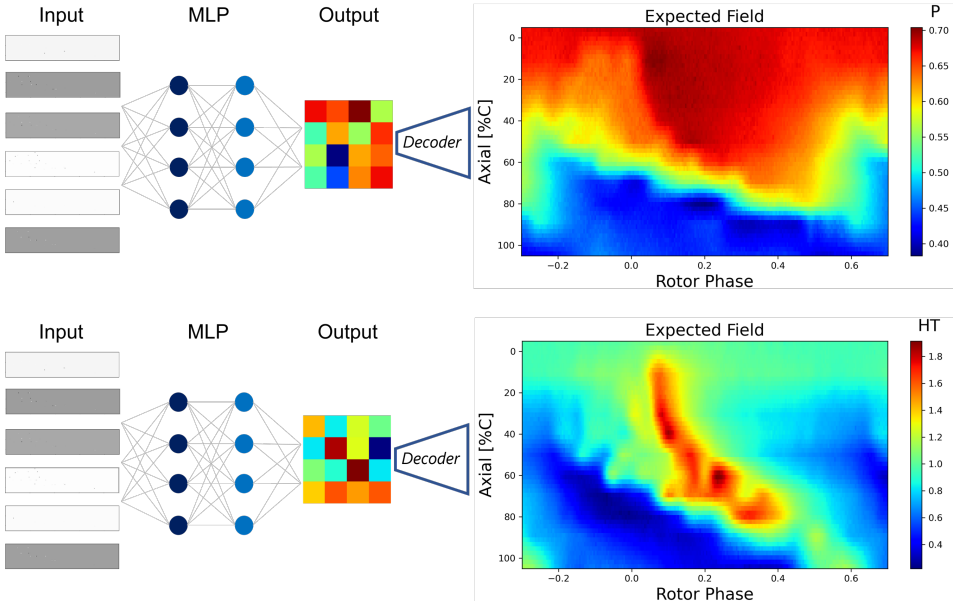


Figure 5.22. Example of network for predicting expected pressure (top) and heat transfer fields (bottom).

This process can be also reversed to discover, given a desired field, the geometry that is more likely to provide this distribution, Figure 5.23.

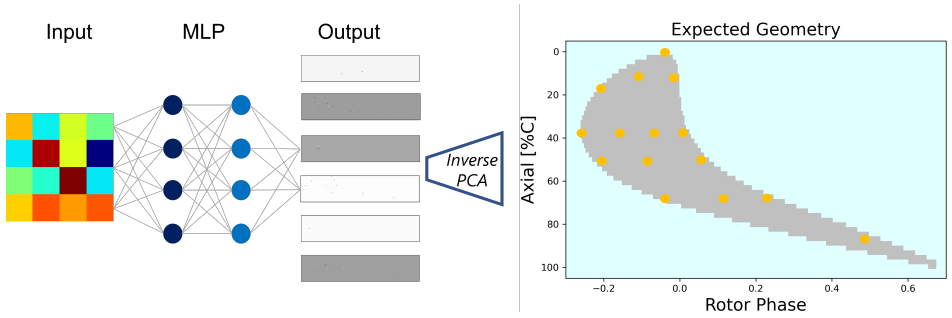


Figure 5.23. Example of network for predicting the expected reduced Rainbow rotor tip geometry.

5.2.3 Validation of the framework

Results obtained from the validation of the framework are reported in Figure 5.24. The agreement between reconstructed and experimentally sampled data has been evaluated by means of the Structural Similarity Index Measure (SSIM). SSIM is found in image processing, and it is used to assess the similarity between two images [139].

A value of SSIM=1 indicates perfect similarity between two images, both in terms of shape and contrast. A quasi-perfect similarity emerges when comparing the original and expected pressure fields, while the similarity is complete for the heat transfer fields (Table 5.5).

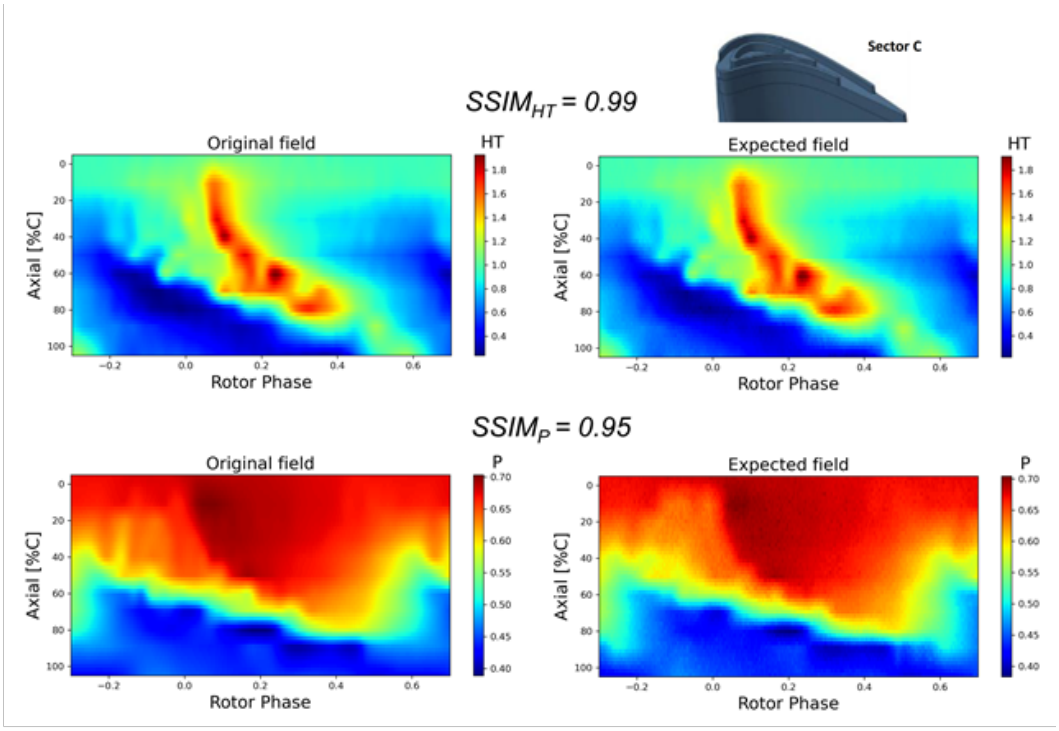


Figure 5.24. Example of SSIM evaluation between the original and expected fields of both heat transfer and pressure for sector C.

Table 5.5. SSIM per sector.

SECTOR	**A**	**B**	**C**
$SSIM_{\mathrm{P}}$	0.95	0.96	0.95
$SSIM_{\mathrm{HT}}$	0.99	0.99	0.99

5.2.4 Final remarks

In this manuscript an using unsupervised dimensionality reduction based on unsupervised machine learning was proposed. The new method is tested against experimental data from a turbine test rig at the von Karman Institute, which has 48 blades separated into 7 sectors with the same baseline rotor geometry but various tip contouring geometries.

The aim of the framework is to discover causal relationships between different tip geometries and unsteady aero-thermodynamic performance, exploiting methods commonly applied to image and sound recognition. The models first up-sample geometries and performance data and then perform unsupervised dimensionality reduction through PCA and AEs to project the original data on latent spaces. Latent space is eventually treated using regression algorithms for further processing in turbomachinery design and optimization.

The resulting maps are in excellent agreement with the original data. A quasi-perfect similarity ($SSIM_{\mathrm{HT}} = 0.99$ and $SSIM_{\mathrm{P}} = 0.95$), evaluated using similarity algorithms, is achieved for both pressure and heat transfer fields.

5.3 A Machine Learning Framework for Condition-Based Maintenance of Gensets in District Heating Networks

The growing interest in natural gas fired genset (a combination of engine and electric generator used to generate electricity and/or heat) is driven by district heating (DH) grid applications, especially in urban areas. Even if they represent customized solutions, when used in DH duty regimes are driven by network thermal energy demands resulting in discontinuous operation which affect their remaining useful life. As such paying attention to effective condition-based maintenance has gained momentum.

In this work, a novel unsupervised anomaly detection framework for gensets in DH networks based on SCADA data is proposed. The framework relies on multivariate ML regression models trained with a Leave-One-Out Cross-Validation method. Model residuals generated during the testing phase are then post-processed with a sliding threshold approach based on a rolling average. This methodology is tested against 9 major failures occurred on the gas genset installed in the Aosta DH plant, in Italy. The results show that the proposed framework succesfully detects anomalies and anticipates SCADA alarms related to unscheduled downtimes.

5.3.1 Introduction

District heating (DH), also known as heat networks or teleheating, provides a platform for heat supply based on the integration of low-carbon technologies, including renewable energy sources, and thermal storage to improve overall efficiency and minimize greenhouse gas emissions. In operation since the end of the XIX century, DH represents an efficient way to provide heat to a large number of users in densely populated urban areas [142, 143, 144, 145]. According to IEA's 2021 report [146], DH systems are important solutions to describe the heating sector in any NZE 2050 scenario [147].

DH systems are composed of thermal plants and a distribution network of insulated pipes that deliver heat to the end users. The thermal plant is based on technologies to generate heat from fossil fuels, renewable energy sourcess, or to valorise waste heat [148]. In 2020 nearly 90% of heat was produced from fossil fuels and one of the most common technologies in DH thermal power plants involves the use of generator sets, with internal combustion engines (ICEs) either in CHP configurations or directly coupled with heat pumps [149]. Wang et al. [150] reported that in 2012 in China, more than 36% of total building energy demand was consumed for residential heating purposes and about 62.9% of district heat was produced by CHP systems. As another example, in Finland, DH accounts for about 50% of the total heating market and in this context the city of Helsinki has around 20% of district heat produced by genset with use of wastewater as low grade heat source [151].

Gensets installed in those contexts, sepcifically when directly coupled with heat pumps, can suffer from intermittent operation caused by the variability and seasonality of the network heat demand. These operation modes often lead the

engine off-design and can be interpreted as the root-cause of genset anomalies and failures. Therefore, the research on automatic Fault Detection (FD) of gensets based on proper Condition-Based Maintenance (CBM) strategies is of paramount importance to monitor operation, reduce downtime and ensure the reliability and productivity of the overall heat supply process [152, 153].

Rooted in condition monitoring systems, CBM aims to establish frameworks for the diagnosis of equipment under supervision indicating incipient failures using sensor networks. CBM defines and monitors health indicators capable of signaling an anomaly in case of deviation from reference values. Based on the evaluation of the current state of the equipment, it is possible to identify faults and malfunctions at early stage, thus allowing the timely planning of maintenance interventions. Despite the fact that scheduled maintenance and CBM are complementary, CBM is by far the most cost-effective approach and the one which enhances the life expectancy of the equipment [154].

A recent review on ICE diagnostics [155] pointed out that a limited number of papers dealt with analytical models specifically designed for the CBM of gensets operating in DH networks. Most of the literature is dedicated to load prediction and analysis of optimal network design, with few contributions focusing on operation and maintenance of networks and distribution pipelines [156].

As reported in [157], Machine learning (ML) algorithms have been established as a viable solution also in the DH arena, because they are easily adaptable to changing conditions, capable of modelling non-linear phenomena, and they can benefit from historical data readily available in modern control systems (e.g. SCADA data). While ML approaches based on classification algorithms, such as the Bayesian classifier or SVM, have been widely used for FD of ICEs [158, 65, 159, 160, 161, 162], regression algorithms seem to represent the most suitable option to perform an effective CBM. This is because the first category of supervised ML tools only allows for FD (based on training of events that have already occurred in the past), while unsupervised models based on regression approaches, classified in [163] as Normal Behaviour Models (NBM), are able to detect anomalies in real time that can signal the onset of fault events in advance. As a general outline, NBM approaches for CBM involve training a reference model representing the healthy status of the system, and evaluating the deviation between the predicted and expected values in real time to detect anomaly occurrence.

To date, most of the applications of NBM in unsupervised fault diagnosis in ICEs fall in the automotive, aeronautical and naval sectors. Some examples include linear regression [164], logistic regression [165], multiclass SVM [166], Extreme Gradient Boosting (XGBoost) [68] and Random Forest [167]. In addition, different types of ANNs have been used such as MLPs [168] or sparse-autoencoders [155].

However, NBM approaches can present a number of critical aspects when applied to multivariate SCADA data. A first aspect concerns the management of the high dimensionality of the data, which affects the response times of NBM models, making them in many cases unsuitable for near real time applications typical of CBM. A second aspect concerns the difficulty in isolating the interval for training the reference model, since the seasonal operational variations of the signals together with the possible presence of anomalies in the dataset make it difficult to identify the standard dynamics of the system using, for example, standard clustering approaches. Finally,

a further critical aspect concerns the handling of residuals for alarm activation. In this case, since residuals are evaluated as the difference between the value of a signal predicted by the regression model trained under reference conditions and the actual value of the same signal recorded by the SCADA sensor, in many cases they present a high level of noise and typical signal variability, which makes it very challenging to trigger alarms using standard control charts.

To overcome these limitations, this work proposes a framework for CBM of natural gas gensets in DH networks, based on a NBM approach applied on multivariate analysis of SCADA data. Specifically, the framework introduces a series of solutions to effectively manage the entire data mining process, starting from the reduction of dimensionality in the pre-processing phase by means of a feature selection approach, passing through the training methods of the reference models with a Leave-One-Out Cross-Validation approach [169], up to the prost-processing of residuals by means of the introduction of a two-stage sliding treshold metric for triggering the alarms. For the ML module two different regression algorithms are compared, namely XGBoost and Multilayer Perceptron. The developed framework is tested against SCADA data sampled on a 7.5 MW NG genset installed in the district heating plant of the city of Aosta, Italy. The testing dataset includes 45 parameters with 5 minute sampling during 16 months of engine operation (September 2019 to December 2020).

5.3.2 Anomaly Detection Framework

The proposed anomaly detection framework based on genset SCADA data is illustrated in Figure 5.25. As a first step the framework preprocesses SCADA event logs and monitored signals in order to optimize the model performance and reduce computational costs. Afterwards, all SCADA signals are first pre-processed in a data cleaning phase and then a feature selection method based on a variable importance approach is applied to select the best predictors for a specific target variable. This optimises the performance of the ML model in terms of computational costs, making it suitable for CBM purposes. In the next step, an XGBoost and an MLP are applied independently for the construction of the reference model. In this case, the training management of ML models is dealt with using a Leave-One-Out Cross-Validation approach. This avoids any risk of overfitting, and guarantees greater robustness and flexibility of the results by simulating an unsupervised application in real time (since the month's data used for testing were never seen during training). However, in order to guarantee the effective learning of the recurring relational dynamics between the different SCADA signals while taking into account the seasonal operational variations typical of the analysed users, it is recommended to have at least one year of data for the training phase. For the tesing phase, the framework provides for the definition of a warning rule for the anomaly detection based on a sliding threshold metric approach applied to the Local Residual Indicators (LRIs) of each parameter. This approach involves an initial filtering of the noise of the local residual indicators and a subsequent definition of a control chart that bases the triggering of the alarms on the intensity and persistence of the filtered LRIs. This allows to report only significant anomalies and thus limit the number of false alarms. Concerning the SCADA event logs, after a preliminary filtering of the minor events, the framework integrates the evaluation of mean-time between alarms (MTBA) indicator and the

quantification of the total downtime. In the final steps, the anomaly detection results are evaluated with reference to the ability to identify specific precursors and early detection of major faults included in the SCADA event logs. The entire framework is implemented using Python 3.9 Scikit-Learn open source library [170]. A step-by-step framework description is given in the following figure.

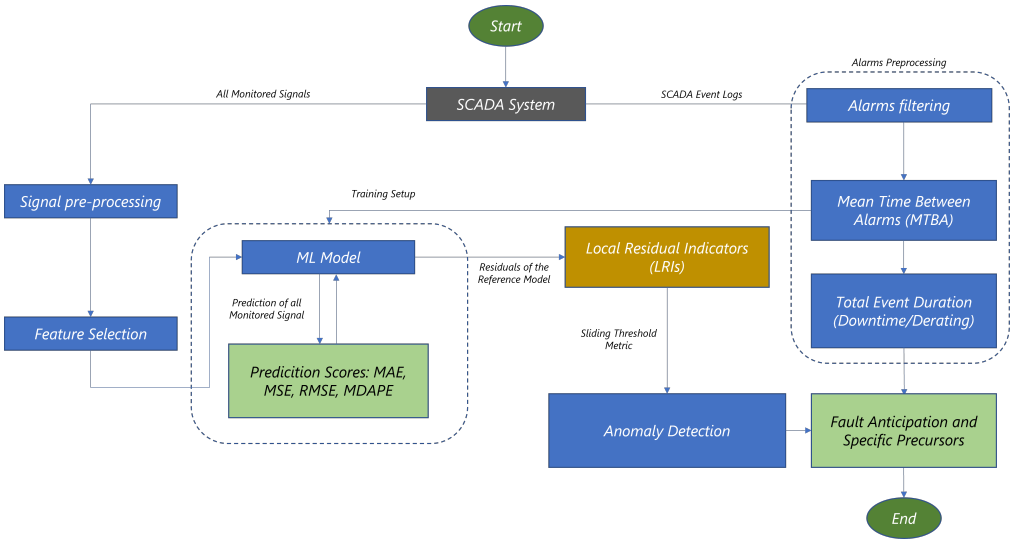


Figure 5.25. ML Framework for CBM, schematics.

5.3.3 SCADA signal and event log preprocessing

The preprocessing of the SCADA event logs filters all minor alarms unrelated to specific faults or anomalies, along with events recorded during engine downtime. The remaining logs are then used to estimate operation metrics such as the MTBA and the total duration of the outage events until correct operations are recorded. Those indicators represent key parameters for the training setup of the ML model.

The information content of each time series is evaluated using the Shannon Entropy (H) metric [171]. Consequently, all parameters with H close to zero have been removed from the training dataset and interpreted as not relevant. In this way, all redundant or derived parameters have been filtered out. Furthermore, the remaining signals have been preprocessed by applying a sigma rule to indentify and remove extreme outliers related to measurement errors.

5.3.4 Feature Selection

The feature selection method has been implemented with a variables importance approach, leveraging the Predictive Power Score (PPS) [76]. The output of the PPS analysis is an asymmetric, data-type independent index that identifies the relationships among the features in a data set. Specifically, through this algorithm it is possible to understand how much one input variable affects the prediction of the values of the target variable. In fact, the PPS is computed by considering a single input feature (x_i) per time that tries to predict the target variable (y_i) via a Decision Tree algorithm with the mean absolute error (MAE) as evaluation metric.

The index is expressed by the formula:

$$PPS = 1 - \frac{MAE^{x_i,y_i}_{model}}{MAE^{y_i}_{naive}} \tag{5.2}$$

where $\mathrm{MAE}^{x_i,y_i}_{model}$ is the mean absolute error of the regression model that predicts y_i starting from a candidate x_i and $\mathrm{MAE}^{y_i}_{naive}$ is relative to a naive model that always predicts the median of y_i. The index ranges from 0 (no predictive power) to 1 (perfect predictive power). On this basis, as suggested by the authors of the algorithm [76], a minimum PPS acceptability limit of 0.2 is consistently set and for each specific target variable (y_i) the vector of best predictors B_i is derived by selectining from the set of all possible input features (x_i) the ones that have a PPS score above the set threshold. For example, as highlighted in Figure 5.26, for the specific target variable (y_i) the vector of best predictors B_i will be given by the subset of input features ranging from x_1 to x_8.

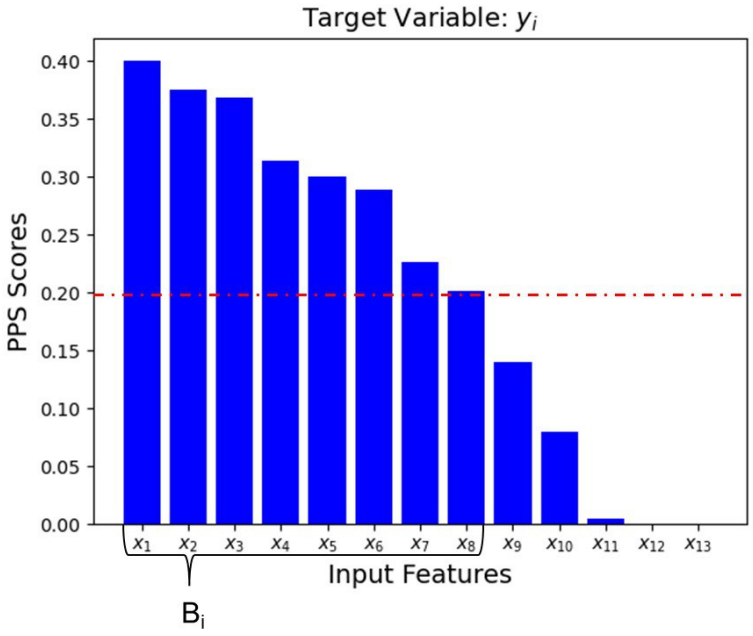


Figure 5.26. Example of the criteria used for the selection of best predictors based on the PPS score.

5.3.5 Machine Learning model

In the present work, two different regression algorithms are tested, namely a XGBoost (XGB) [172] and a Multi-layer Perceptron (MLP) [173], as candidate for the ML module. Both the regression algorithms were optimised using a grid search approach [174] to select the best combination of hyper-parameters. Notably, since XGB belongs to the category of ensemble algorithms and its structure is composed of several decision trees, the results are independent from feature normalisation [104]. On the contrary, being based on statistical analysis, Artificial Neural Network are strongly

influenced by the distribution and quality of the data and are highly dependent on the order of magnitude of their input values. As a consequence, the predictor may neglect or overestimate the influence of some features according to their values [175]. To avoid this, the input signals for the MLP model are normalized using a Standard Scaler. Then, the features predicted by the MLP are scaled back to their original size through an inverse Standard Scaler transformation. This ensures the comparability of results between the two ML models in terms of prediction scores. Both models work by identifying within the training dataset one parameter at a time as the target variable (y_i) and exploiting all the others to predict it.

Figure 5.27. Leave-One-Out Cross-validation schematics.

Training Setup. The training model adopts a Leave-One-Out Cross-Validation method [169], due to the difficulty of isolating reference operating conditions with standard unsupervised approaches due to the highly discontinuous operation (e.g. in the period March to September) combined with the strong seasonality of the signals. In detail, as shown in Figure 5.27, one month is cyclically isolated as the testing dataset D_{test} and the model is trained on the remaining months split between training D_{train} and validation D_{val} datasets. This approach, meant to avoid possible overfitting, is based on the hypothesis that most of the operational data over such a long period of time refers to normal engine operation. To further reduce the possible presence of failure precursors in the reference model, all downtimes and a period prior to them equal to the value of the MTBA index calculated during the preprocessing phase of the SCADA logs have been excluded from D_{train}. By applying this training logic, a specific regression model (ML $Model_i$) for each target variable (y_i) is obtained, harnessing the vector of best predictors (B_i) previously identified.

The accuracy of the two models during the training phase on the reference period has been evaluated with customary scores i.e. MAE (Mean Absolute Error), MSE

(Mean Squared Error), RMSE (Root Mean Squared Error), MDAPE (Mean Absolute Percentage Error).

5.3.6 Residual Indicator definition

The CBM framework entails, then, an anomaly detection rule to enable the early warning of incipient failures included in the test dataset. To this end, a Local Residual Indicator (LRI) for each monitored variable [129] is defined. The LRI of each signal is defined as the absolute value of the difference between the actual values (f) and those predicted by the models trained on the reference period (f_p):

$$LRI = |f - f_p| \tag{5.3}$$

Therefore, with the aim of triggering an early warning before the occurrence of faults, while limiting false alarms associated with instantaneous peaks of LRIs, a sliding threshold metric based on a moving average is developed. In particular, as shown in Figure 5.28, an alarm is triggered for a signal when the following condition is satisfied for P consecutive time steps:

$$LRI_i \geq 0.5 \cdot \frac{1}{W} \Sigma_{j=i-W}^{i} LRI_j \tag{5.4}$$

where LRI_i is the local residual indicator of the signal at time i and W is the length of the sliding window. The value of W is selected according to the periodicity of the observed phenomena and, in this case, is set to 288 (24 hours). Thus, when the average LRI value of a signal undergoes a variation $\geq$50% compared to the last 24 recorded hours and this variation persists for at least P time steps, an alarm is triggered for the specific sensor of that LRI. The persistence threshold P was set to 72 (6 hours) and has been optimized to effectively remove residual noise.

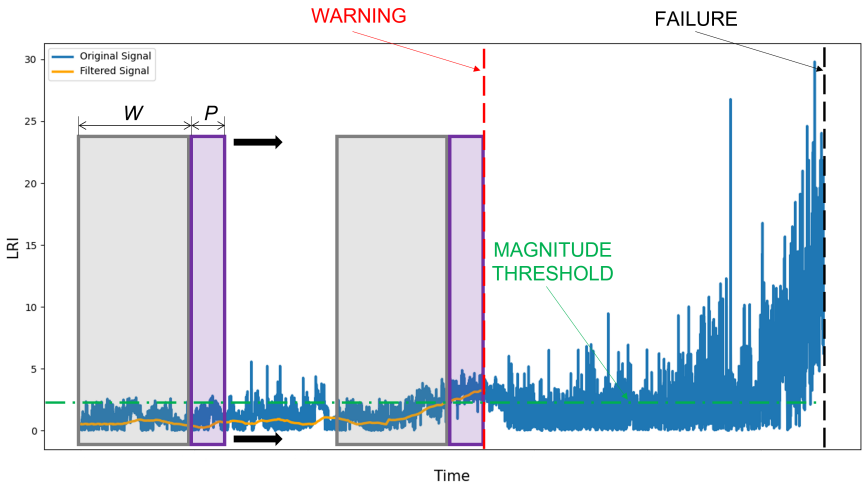


Figure 5.28. Sliding threshold metric.

This approach is particularly suitable for this type of dataset because a standard control chart with a fixed threshold on the size of residuals may be ineffective due to

their extreme variability in different seasons. Moreover, it guarantees high robustness in handling the noise of the residuals of the models.

Based on such a warning rule, the performance of the models in terms of anomaly detection capability on each cross-validation datasets being cyclically isolated is finally evaluated. This assessment aims to quantify the ability of each warning to anticipate the major failure events included in the SCADA log.

5.3.7 Dataset description

Data is collected from a natural gas genset installed in the Aosta District heating plant, which is equipped with a 16-cylinder turbocharged ICE. The engine has a nominal electric power output of 7.5 MWe, and it is directly coupled to a 17.5 MWt heat pump. ICE technical specifications are given in Table 5.6.

Table 5.6. Technical specifications of the engine

N. of Cylinders	16	[–]
Engine Speed	720	[r/min]
Electrical Output	7235	[kW]
Charge Air Cooler HT	1305	[kW]
Charge Air Cooler LT	490	[kW]
Lube Oil Cooler	730	[kW]
Jacket Water Cooler	925	[kW]
Exhaust Mass	39600	[kg/h]
Exhaust Gas Temperature	355	[°C]

A SCADA system monitors different operating parameters which are collected by the main components of the genset togheter with environmental measurements. In detail, the starting dataset includes 45 parameters sampled every 5 minutes for a period of 15 months: from September 2019 to December 2020. After the application of the signal preprocessing described before, the total number is reduced to 33 significant parameters listed in Table 5.7.

Table 5.7. List of SCADA signals

Signal ID	**Description**
P01-P19, P23, P25-P26	Cylinder, exhaust and intake temperatures
P20-P22, P24	Cylinder and fuel subsystem pressures
P27-P31	Generator phase and bearing temperatures
P32	Active Power
P33	Ambient Temperature

Table 5.8. Event logs

Month	Event ID	SCADA Event Log	Type of Event	Start (dd/mm/yy; hh:mm)	End (dd/mm/yy; hh:mm)	Total Duration (hh)
Oct-2020	S_01_10	Shutdown from Main Control	Normal Stop	01/10/2020; 21:30	02/10/2020; 07:30	10,00
	S_02_10	Shutdown from Main Control	Normal Stop	02/10/2020; 14:20	02/10/2020; 16:25	2,08
	S_03_10	Shutdown from Main Control	Normal Stop	03/10/2020; 00:00	02/10/2020; 07:25	7,42
	DS_05_10	Exh Temp Deviation Cylinder	Derating + Unscheduled Downtime	05/10/2020; 03:10	05/10/2020; 13:55	10,75
	S_06_10	Shutdown from Main Control	Normal Stop	06/10/2020; 08:35	06/10/2020; 10:10	1,58
	S_07_10	Emergency Stop Activated	Unscheduled Downtime	07/10/2020; 07:30	12/10/2020; 10:30	123
	DS_15_10	Exh Temp Deviation Cylinder	Derating + Unscheduled Downtime	15/10/2020; 18:55	15/10/2020; 23:45	10,75
	S_20_10	Emergency Stop Activated	Unscheduled Downtime	20/10/2020; 10:45	20/10/2020; 16:00	5,25
	DS_26_10	Exh Temp Deviation Cylinder	Derating + Unscheduled Downtime	26/10/2020; 04:30	26/10/2020; 15:35	11,08
Nov-2020	S_06_11	Shutdown from Main Control	Normal Stop	06/11/2020; 21:30	07/11/2020; 00:10	2,67
	S_08_11	Shutdown from Main Control	Normal Stop	08/11/2020; 03:16	08/11/2020; 05:00	1,73
	S_09_11	Shutdown from Main Control	Normal Stop	09/11/2020; 08:22	09/11/2020; 09:40	1,30
	S_13_11	Emergency Stop Activated	Unscheduled Downtime	13/11/2020; 10:13	13/11/2020; 14:15	4,03
	D_14_11	Exh Temp Deviation Cylinder	Derating	14/11/2020; 00:47	14/11/2020; 06:55	6,13
	D_16_11	Charge Air Temp After Cooler High	Derating	16/11/2020; 13:30	16/11/2020; 14:05	0,58
	S_19_11	Emergency Stop Activated	Unscheduled Downtime	19/11/2020; 02:55	21/11/2020; 03:15	48,33
	S_26_11	Shutdown from Main Control	Normal Stop	26/11/2020; 19:40	26/11/2020; 21:05	1,42
Dic-2020	D_12_12	Charge Air Temp After Cooler High	Derating	12/12/2020; 01:15	12/12/2020; 01:35	0,33
	S_13_12	Shutdown from Main Control	Normal Stop	13/12/2020; 20:09	13/12/2020; 21:05	0,93
	DS_16_12	Generator Stator Temp Windings	Derating + Unscheduled Downtime	16/12/2020; 11:15	16/12/2020; 12:20	1,08
	S_19_12	Shutdown from Main Control	Normal Stop	19/12/2020; 05:13	19/12/2020; 06:40	1,45
	S_21_12	Emergency stop Activated	Unscheduled Downtime	21/12/2020; 06:11	21/12/2020; 18:35	12,40
	D_21_12	Charge Air Temp After Cooler High	Derating	21/12/2020; 00:40	21/12/2020; 04:30	3,83

In addition to the SCADA signals, to assess the anomaly detection capability of the framework, all the alarms recorded by the SCADA system in the period October-December 2020 are considered. During this period several major failures occurred. After filtering out the minor alarms, the remaining events included scheduled (i.e. normal stop) and unscheduled downtime (i.e. emergency stop or outages after engine deratings). Table 5.8 lists all the filtered SCADA events in the reference period.

5.3.8 ML settings and prediction errors

Identical training and cross-validation phases have been carried out for the ML models, namely XGB and MLP algorithms. The dataset is split into training set D_{train} (70%) and a validation set D_{val} (30%). The testing set D_{test} consists of a specific month cyclically isolated from the available data and includes the time periods of failure occurrences.

The XGB model learning task was set to linear regression and Table 5.9 lists the other hyperparameters optimized through a grid search algorithm. Table 5.10, on the other hand, details the final setup of the MLP model, considering early stopping to avoid overfitting.

Table 5.9. Hyperparameters of the XGB Model.

Subsampling of columns	0.20
Learning rate	0.10
Max depth	50
Nr. of trees	150
Nr. of parallel trees	20
Alpha	0
Lambda	1

Table 5.10. Hyperparameters of the MLP Model.

Nr. of Neurons	22
Nr. of hidden layer	1
Nr. of training epochs	150
Activation function	relu
Initial learning rate	1E-5
Optimizer	ADAM
Batch size	1/50th

The ML models' predictions are evaluated in terms of the reconstruction errors of all SCADA signals (during the training phase the predicted values are compared with the actual ones). As can be seen from Table 5.11, XGB outperforms MLP in terms of customary scores (e.g. MAE, MSE, RMSE and MDAPE).

Table 5.11. Reconstruction errors for the proposed ML models.

	XGBoost	**MLP**
MAE	0.04	0.11
MSE	0.10	0.14
RMSE	0.21	0.31
MDAPE	0.01	0.13

5.3.9 Anomaly detection results

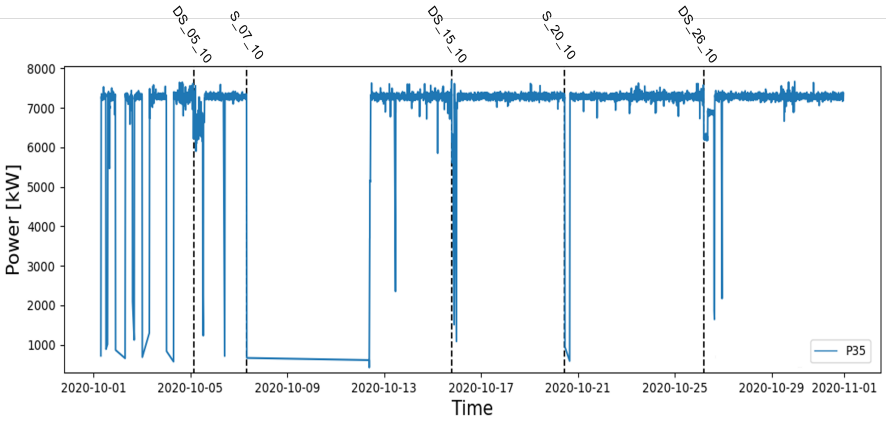
For the evaluation of the anomaly detection capabilities, the results of the testing phase during the period October-December 2020 are considered.

Specifically, ML model results are discussed by plotting the LRIs against the relative warnings activated on the individual parameters after the application of the sliding threshold metrics. Furthermore, as a reference to identify engine derating and shutdown, the graph of the active power together with the details of the main alarms recorded by the SCADA system in the same time interval are shown.

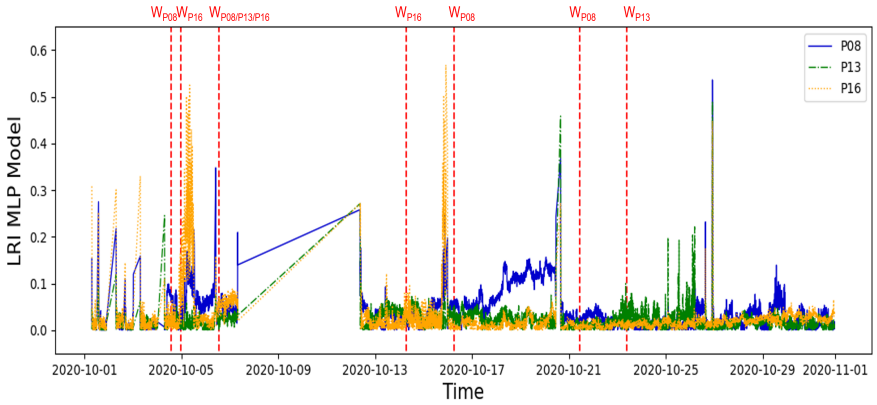
Figure 5.29, first, illustrates the results in October 2020. Figure 5.29a shows the active power, with the detail of SCADA event logs recorded in that period. Event IDs refer to Table 5.8. Figures 5.29b and 5.29c show the LRI trends together with the warnings triggered by the framework (highlighted in dashed red lines).

By analysing October 2020 SCADA logs, a total of five significant events can be isolated. Those events include three anomalies that resulted in a preliminary power output derating followed by the engine shutdown, along with two emergency stops linked to unscheduled downtimes. Regarding the first category of events it is worth noting that all the shutdowns are anticipated by cylinder temperature anomalies and that the application of the proposed framework allows the early detection of such a precursors. In detail, concerning the events detected on 05/10/2020 (event ID: DS_05_10) and 15/10/2020 (event ID: DS_15_10) respectively, a significant deviation of the LRI associated with cylinder temperature parameter (P16) can be seen both in Figure 5.29b and 5.29c, which leads to early warnings with respect to the actual SCADA log (details of the advance times relative to the two ML models

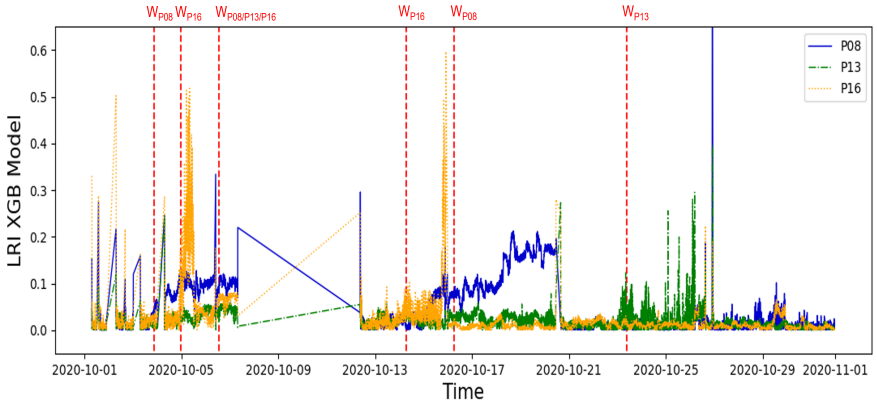
can be seen in Table 5.12). Furthermore, while the warning on the S_15_10 event is triggered by the two models at the same time, for the DS_05_10 event the MLP model detects the anomaly about ten hours earlier than the XGB model. The third derating event followed by an engine shutdown was recorded on 26/10/2020 (event ID: DS_26_10) and concerned a high temperature alarm on cylinder 5B (P13) detected on 26/10/2020. Also in this case in Figure 5.29a and 5.29b it is possible to notice a specific precursor signalled by a significant variation of the LRI of the parameter P13 that leads to a common warning both in the MLP and XGB models on 23/10/2020, about three days in advance compared to the SCADA log.



(a) Active Power

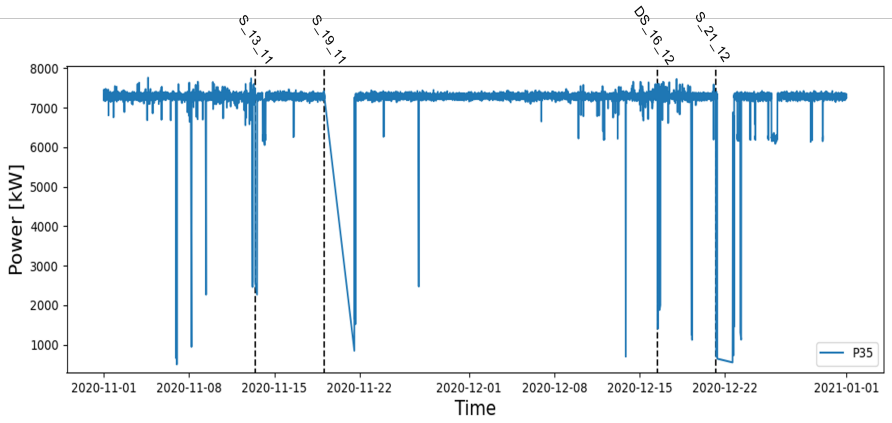


(b) MLP Model Results

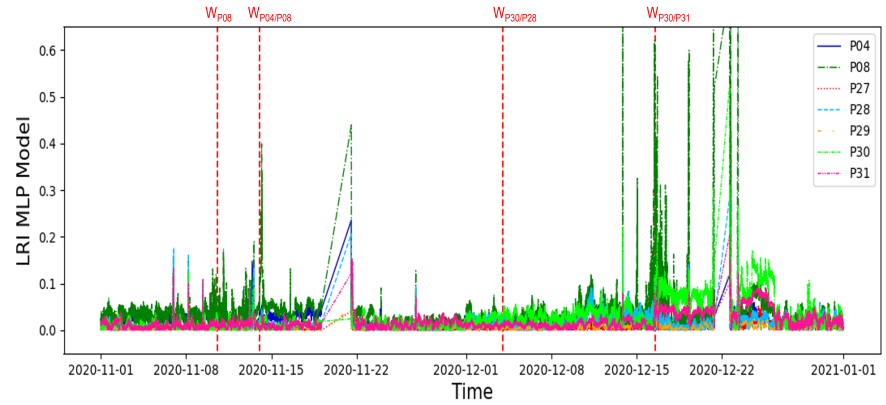


(c) XGB Model Results

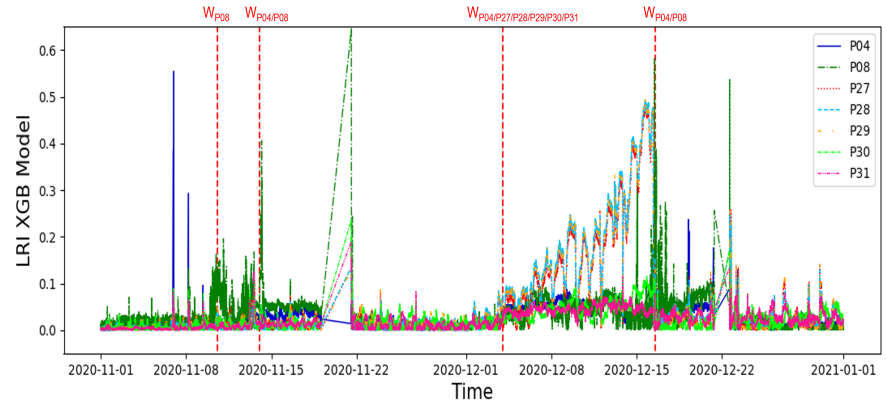
Figure 5.29. Results of the ML Framework for CBM with reference to October 2020. 5.29a shows the trend of the active power, with the details of the SCADA events recorded in that period (black dashed line); 5.29b and 5.29c show the trend of the Local Residual Indicators (LRIs) relating to the parameters that generated specific warnings (red dashed line) after the application of the sliding threshold metric for the MLP and XGB models, respectively.



(a) Active Power



(b) MLP Model Results



(c) XGB Model Results

Figure 5.30. Results of the ML Framework for CBM with reference to November and Dicember 2020. 5.30a shows the trend of the active power, with the details of the SCADA events recorded in that period (black dashed line); 5.30b and 5.30c show the trend of the Local Residual Indicators (LRIs) relating to the parameters that generated specific warnings (red dashed line) after the application of the sliding threshold metric for the MLP and XGB models, respectively.

Even more interesting are the advances found before the emergency stops on 07/10/2020 (event ID: S_07_10) and 20/10/2020 (event ID: S_20_10) since they are not associated with a specific SCADA anomaly alarm on a component of the gas genset. With reference to these unscheduled downtimes, the results of the ML models show an anomaly on the LRI of cylynder temperature (P08) which causes a warning three days in advance of the first event (84 hours for XGBoost and 62 hours for MLP). Subsequently, the indicator of parameter P08 returns to normal values after the maintenance intervention, as visible in the active power plot in Figure 5.29a), and then deviates significantly starting again from 16/10/2020 (see Figures 5.29b and 5.29c) up to the emergency stop event recorded on 20/10/2020.

In a similar fashion, Figure 5.30, compares the results of the CBM method during the period of November and December 2020. Looking at the events, Table 5.8, four significant unscheduled downtimes have been reported by the SCADA system in these two months. Those events include three emergency stop alarms recorded respectively on 11/13/2020 (event ID: S_13_11), 11/19/2020 (event ID: S_19_11), and 12/21/2020 (event ID: S_21_12), and a shutdown transient due to an anomaly found on the generator temperature on 12/16/2020 (event ID: DS_16_12). From a global analysis of the LRI trends, shown in Figure 5.30b and 5.30c, different anomalies appeared during period under scrutiny, affecting engine cylinders and the generator. In detail, the anomalies already found on cylinder exhaust temperature, correlated with two long outages in October 2020, recur from 13/11/2020, when a warning on the involved parameter was triggered by both MLP (Figure 5.30b) and XGBoost (Figure 5.30c) models. This significant deviation of the P08 parameter indicator persists for about three days until the emergency stop recorded on 13/11/2020. Immediately after this 4-hour engine outages, a new significant anomaly was detected by both models on P08 involving also other cylinder temperature and anticipating the failure detected by SCADA on 19/11/2020 (event S_19_11).

Of particular interest are the results obtained with reference to the other two significant events recorded by the SCADA in December 2020, namely DS_16_12 and S_21_12. This is because, while in the evaluation of the results conducted so far, the warnings detected by XGBoost and MLP have always involved the same parameters (with some divergence only on the anticipation of SCADA events), in these cases the two models highlight different precursors. In particular, the XGBoost LRIs (Figure 5.30c) highlight on 04/12/2020 a marked variation in the three temperatures of the generator related variables, phases and bearings (P27-P31). This results in a warning that anticipates the SCADA log DS_16_12 by about twelve days. Comparing these results with those of the MLP model (Figure 5.30b), the same significant deviation is not noticed on the generator stator winding, but only an alarm is triggered on the two generator bearings. As for the unscheduled machine downtime of 12/21/2020, it is detected about 5 days in advance by both models, with different precursors: exhaust cylinder temperatures (P01-P19) for the XGB model and generator bearing temperatures (P27-P31) for the MLP model.

Eventually, Table 5.12 summarizes the results discussed so far. In particular, the ability of each of the two ML models to identify specific precursors for major faults included in the SCADA log is assessed and then the time of advance warning of the model relative to the occurrence of the reference SCADA alarm is quantified.

Table 5.12. Unscheduled downtime events recorded by the SCADA system in the period October-December 2020.

Event ID	XGB Results			MLP Results		
	Detection (dd/mm/yy; hh/mm)	Anticipation (hh)	Precursors ID	Detection (dd/mm/yy; hh/mm)	Anticipation (hh)	Precursors ID
DS_05_10	04/10/2020; 23:30	4	P16	04/10/2020; 13:45	14	P16
S_07_10	03/10/2020; 20:25	84	P08	04/10/2020; 17:20	62	P08
	06/10/2020; 13:15	18	P13, P16	06/10/2020; 14:05	17	P13, P16
DS_15_10	14/10/2020; 06:40	37	P16	14/10/2020; 08:10	34	P16
S_20_10	16/10/2020; 06:00	101	P08	16/10/2020; 07:05	100	P08
DS_26_10	23/10/2020; 09:30	67	P13	23/10/2020; 10:45	65	P13
S_13_11	10/11/2020; 12:55	69	P08	10/11/2020; 14:15	68	P08
S_19_11	14/11/2020; 00:25	123	P04, P08	14/11/2020; 01:05	122	P04, P08
DS_16_12	04/12/2020; 00:10	299	P28, P29, P30	04/12/2020; 01:25	298	P29, P31
	04/12/2020; 00:20	299	P04, P31, P32	-	-	-
S_21_12	16/12/2020; 12:05	114	P04, P08	16/12/2020; 13:00	113	P31, P32

5.3.10 Final Remarks

This work presents an unsupervised anomaly detection framework for the CBM of gas genset in DH networks based on SCADA data. The core of the method is a ML model, which considers the whole SCADA data stream as input and predicts one signal at the time. For this purpose, two different models are tested, namely MLP and XGBoost regressor. These models was trained to learn the normal behaviour of the system based on a Leave-One-Out Cross-Validation approach and, based on the model reconstruction errors, a Local Residual Indicator (LRI) was defined for each monitored variable. Therefore, with the aim of triggering an early warning before the occurrence of faults, while limiting false alarms associated with instantaneous peaks in LRIs, a sliding threshold metric based on a moving average is developed . In this way, a warning is triggered for the signals having the highest reconstruction error, allowing to isolate the parameters mostly involved in the anomaly for troubleshooting purposes.

The proposed method was validated on 5-minute SCADA data collected from a 7.5 MWe natural gas genset installed in the District Heating Plant of the Aosta city, in Italy. The model was trained on normal behaviour data isolated using an unsupervised method and was tested on anomalous periods selected using the SCADA event log. Results show that the proposed multivariate nowcasting approach allows to unveil hidden precursor dynamics that anticipate all the main fault events occurred in the investigated period. It is interesting to note that these anomalies was not detected by the single-variable operational control approaches typical of SCADA control systems. In addition, even if both the ML models tested anticipate the same faults with similar advance times, the better performance of XGB compared to MLP is evident in terms of training customary scores for the nowcasting of single parameters (Table 5.11). This is reflected in a better accuracy in detecting specific fault precursors, as in the case of the LRIs shown in Figure 5.30, where, with reference to a fault occurred on 16/12/2020 which involved the generator, XGB reports about twelve days in advance an anomaly on the temperature stator windings, which was not caught by MLP. Since the proposed multivariate framework is unsupervised and completely data-driven, it fits well with the purposes of the CBM and also can be potentially applicable to any gas genset equipped with a SCADA system.

Chapter 6

Conclusions

Although the concept of the Digital Twin is decades old, its real impact is only becoming apparent in recent years. The digital market is and will continue to grow as more and more industries adopt this technology owing to its potential in reducing operational costs and time, increasing the productivity of the existing system, improving maintenance, easing accessibility, creating a safer work environment, and other purposes yet to be realized. Indeed, identifying and understanding the potential of the Digital Twin in any industry and integrating it appropriately provides an opportunity to develop tools for Industry 4.0. that offer numerous benefits such as simulation and prediction capabilities. Digital Twin technology, when combined with other technologies and/or other Digital Twins, will open the door to new applications and potentials, and even though there are many challenges in developing this technology, the benefits are far greater. The benefits include reduced costs, increased results, remote access, and rationalization of services and resources, all of which can be achieved by operating the system digitally instead of physically, without any additional material or investment costs and in less time; while the challenges can be attributed to the novelty of the technology: lack of consensus on its definition and value, lack of standards and regulations, lack of competent engineers and technicians, and lack of supporting software.

As a result, Digital Twin is still an emerging technology and the infrastructure for its implementation needs to be improved to increase its effectiveness. Therefore, to create a successful Digital Twin, further researches on technologies such as high-performance computing technology, machine learning technology, real-time virtual-real interactive technology, intelligent perception and connection technology, among others, are needed. However, in recent years, thanks to the introduction of Artificial Intelligence and Machine Learning into manufacturing processes that leverage Big Data collected from them, Digital Twinning is evolving at a rapid rate, and many new opportunities are emerging.

This thesis proposes new Artificial Intelligence strategies and Machine Learning frameworks for the implementation of Digital Twin and Cyber-Physical systems that realize the transition from classical design, performance optimization and maintenance of energy systems to advanced procedures inspired by Industry 4.0. Great efforts have been made to intertwine Machine Learning techniques with standard design and performance analysis procedures for energy systems and turbomachinery.

Learning algorithms have been thoroughly analyzed and described, highlighting their advantages and limitations and evaluating their role in transforming the entire energy industry to achieve increased efficiency with reduced costs.

The first chapter provides an overview of the background and motivation for this work, while the key concepts of Cyber-Physical Systems, Digital Twin, and Digital Thread for energy systems are discussed in the second chapter. The growing interest in those technologies by designers of turbomachinery and energy systems is explained by the many benefits that would be gained by adopting them: forecasting, design, testing and monitoring capabilities, in a real-time digital environment, would significantly reduce costs and time consumption at multiple stages of product life.

The third chapter explores the crucial role of Artificial Intelligence tools in developing a successful Digital Twin model. In fact, the core of a Digital Twin is based on data, and the information contained therein facilitates all production and monitoring activities, enabling a holistic understanding of the entire process. Therefore, a successful Digital Twin relies on continuously evolving digitization technologies such as Machine Learning.

All the machine learning algorithms that have been implemented in this work and the importance of data analysis as a preprocessing tool to improve data quality before any further analysis are detailed and discussed in the fourth chapter.

The last chapter is the focus of this thesis, in which new design and maintenance concepts for turbomachinery and energy systems are presented that provide a digital transition from classical and most often empirical approaches through machine learning strategies:

- the first application deals with turbomachinery design via CFD and presents an integrated method to derive a meta-model for predicting the deflection of a modified sinusoidal leading-edge cascade; the method is based on unsupervised learning techniques and explores a dataset that includes 76 RANS computations of the flow field in a cascade according to the Central Composite Design of Experiment;

- the second application concerns turbomachinery design through experimental campaigns and illustrates a high-fidelity dimensionality reduction procedure of a large experimental dataset from HPT with unsupervised learning. The goal of the method is to discover the causal relationship between different tip geometries and performance by first up-sampling the geometries and performance data and then unsupervised dimensionality reduction through PCA and AE to project the original data onto latent spaces, which are treated with regression algorithms for further processing in turbomachinery design and optimization;

- the third application is based on energy system operation monitoring and involves condition-based maintenance of gensets with a machine learning framework. The framework is trained with a Leave-One-Out Cross-Validation approach to learn normal behavior of the system and a Local Residual Indicator is defined and processed for triggering early warnings that anticipate failures.

To conclude, AI algorithms play an important role in the development of DTs for decision making at any level of the product life cycle. However, selecting a particular

model from hundreds of ML models with a customized configuration is challenging, since each artificial intelligence approach has different levels of accuracy and efficiency with different applications on different datasets. Therefore, as highlighted in this thesis, depending on the reason and application of a DT, selecting the best ML algorithm and feature set is challenging but at the same time essential.

Bibliography

[1] Morteza Ghobakhloo. "Industry 4.0, digitization, and opportunities for sustainability". In: *Journal of cleaner production* 252 (2020), p. 119869.

[2] Elena Goosen et al. "Toward industry 4.0 in energy sector". In: *IOP Conference Series: Materials Science and Engineering*. Vol. 865. 1. IOP Publishing. 2020, p. 012020.

[3] Abdus Samad and Kwang-Yong Kim. "Surrogate based optimization techniques for aerodynamic design of turbomachinery". In: *International Journal of Fluid Machinery and Systems* 2.2 (2009), pp. 179–188.

[4] David B Durocher and Gerry R Feldmeier. "Predictive versus preventive maintenance". In: *IEEE Industry Applications Magazine* 10.5 (2004), pp. 12–21.

[5] Kyu Tae Park et al. "Cyber physical energy system for saving energy of the dyeing process with industrial internet of things and manufacturing big data". In: *International Journal of Precision Engineering and Manufacturing-Green Technology* 7.1 (2020), pp. 219–238.

[6] Victor Singh and Karen E Willcox. "Engineering design with digital thread". In: *AIAA Journal* 56.11 (2018), pp. 4515–4528.

[7] M Mazhar Rathore et al. "The role of ai, machine learning, and big data in digital twinning: A systematic literature review, challenges, and opportunities". In: *IEEE Access* 9 (2021), pp. 32030–32052.

[8] Oliver Kramer. "Scikit-learn". In: *Machine learning for evolution strategies*. Springer, 2016, pp. 45–53.

[9] Peter Goldsborough. "A tour of tensorflow". In: *arXiv preprint arXiv:1610.01178* (2016).

[10] Goong Chen et al. "OpenFOAM for computational fluid dynamics". In: *Notices of the AMS* 61.4 (2014), pp. 354–363.

[11] Takeru Kuroiwa, Yusuke Aoyama, and Noriyuki Kushiro. "Testing environment for CPS by cooperating model checking with execution testing". In: *Procedia Computer Science* 96 (2016), pp. 1341–1350.

[12] A. V. Jha et al. "Smart Grid Cyber-Physical Systems: Communication Technologies, Standards and Challenges". In: *Wirel. Netw.* 27.4 (2021), 2595–2613. ISSN: 1022-0038. DOI: 10.1007/s11276-021-02579-1. URL: https://doi.org/10.1007/s11276-021-02579-1.

[13] Fan Zhang et al. “A novel CPS system for evaluating a neural-machine interface for artificial legs”. In: *2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*. IEEE. 2011, pp. 67–76.

[14] Stephan Weyer et al. “Future modeling and simulation of CPS-based factories: an example from the automotive industry”. In: *Ifac-Papersonline* 49.31 (2016), pp. 97–102.

[15] Gang Xiong et al. “Cyber-physical-social system in intelligent transportation”. In: *IEEE/CAA Journal of Automatica Sinica* 2.3 (2015), pp. 320–333.

[16] Jay Taneja, Randy Katz, and David Culler. “Defining cps challenges in a sustainable electricity grid”. In: *2012 IEEE/ACM Third International Conference on Cyber-Physical Systems*. IEEE. 2012, pp. 119–128.

[17] Paulo Leitão, Armando Walter Colombo, and Stamatis Karnouskos. “Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges”. In: *Computers in industry* 81 (2016), pp. 11–25.

[18] Ming-Chuan Chiu, Chien-De Tsai, and Tung-Lung Li. “An integrative machine learning method to improve fault detection and productivity performance in a cyber-physical system”. In: *Journal of Computing and Information Science in Engineering* 20.2 (2020), p. 021009.

[19] Ragunathan Rajkumar et al. “Cyber-physical systems: the next computing revolution”. In: *Design automation conference*. IEEE. 2010, pp. 731–736.

[20] Walid Taha. “Lecture Notes on Cyber-Physical Systems”. In: (2013).

[21] Aaron Parrott and Lane Warshaw. “Industry 4.0 and the digital twin”. In: *Deloitte Insights* (2017).

[22] Maulshree Singh et al. “Digital twin: Origin to future”. In: *Applied System Innovation* 4.2 (2021), p. 36.

[23] M Grieves. “Origins of the Digital Twin Concept. 2016”. In: *Publisher Full Text* ().

[24] Michael W Grieves. “Product lifecycle management: the new paradigm for enterprises”. In: *International Journal of Product Development* 2.1-2 (2005), pp. 71–84.

[25] Michael Grieves. “Back to the future: product lifecycle management and the virtualization of product information”. In: *Product Realization*. Springer, 2009, pp. 1–13.

[26] Mike Shafto et al. “Modeling, simulation, information technology & processing roadmap”. In: *National Aeronautics and Space Administration* 32.2012 (2012), pp. 1–38.

[27] Michael Grieves and John Vickers. “Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems”. In: *Transdisciplinary perspectives on complex systems*. Springer, 2017, pp. 85–113.

[28] Henrique Almeida et al. *Progress in Digital and Physical Manufacturing*. Springer, 2020.

[29] Fei Tao et al. “Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0: Correlation and comparison”. In: *Engineering* 5.4 (2019), pp. 653–661.

[30] David Jones et al. “Characterising the Digital Twin: A systematic literature review”. In: *CIRP Journal of Manufacturing Science and Technology* 29 (2020), pp. 36–52.

[31] C Miskinis. “What does a digital thread mean and how it differs from digital twin”. In: *Retrieved from Challenge Advisory: https://www. challenge. org/insights/digital-twin-anddigital-thread* (2018).

[32] Edward Kraft. “HPCMP CREATE™-AV and the air force digital thread”. In: *53rd AIAA Aerospace Sciences Meeting*. 2015, p. 0042.

[33] Rong Xie et al. “Digital twin technologies for turbomachinery in a life cycle perspective: a review”. In: *Sustainability* 13.5 (2021), p. 2495.

[34] Alessandro Baldassarre et al. “Towards a digital twin realization of the blade system design study wind turbine blade”. In: *Wind and Structures* 28.5 (2019), pp. 271–284.

[35] Iris Graessler and Alexander Poehler. “Intelligent control of an assembly station by integration of a digital twin for employees into the decentralized control system”. In: *Procedia Manufacturing* 24 (2018), pp. 185–189.

[36] Banavara R Seshadri and Thiagarajan Krishnamurthy. “Structural health management of damaged aircraft structures using digital twin concept”. In: *25th aiaa/ahs adaptive structures conference*. 2017, p. 1675.

[37] Xiao Yuan and Chimay J Anumba. “Cyber-physical systems for temporary structures monitoring”. In: *Cyber-physical systems in the built environment* (2020), pp. 107–138.

[38] Yu Zheng, Sen Yang, and Huanchong Cheng. “An application framework of digital twin and its case study”. In: *Journal of Ambient Intelligence and Humanized Computing* 10.3 (2019), pp. 1141–1153.

[39] Kai Ding et al. “Smart steel bridge construction enabled by BIM and Internet of Things in industry 4.0: A framework”. In: *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*. IEEE. 2018, pp. 1–5.

[40] Yu Zhou et al. “Digital-twin-driven geometric optimization of centrifugal impeller with free-form blades for five-axis flank milling”. In: *Journal of Manufacturing Systems* 58 (2021), pp. 22–35.

[41] Adebena Oluwasegun and Jae-Cheon Jung. “The application of machine learning for the prognostics and health management of control element drive system”. In: *Nuclear Engineering and Technology* 52.10 (2020), pp. 2262–2273.

[42] Kazi Masudul Alam and Abdulmotaleb El Saddik. “C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems”. In: *IEEE access* 5 (2017), pp. 2050–2062.

[43] Meng Zhang, Fei Tao, and AYC Nee. "Digital twin enhanced dynamic job-shop scheduling". In: *Journal of Manufacturing Systems* 58 (2021), pp. 146–156.

[44] Cameron Tropea, Alexander Yarin, and John Foss. *Springer Handbook of Experimental Fluid Mechanics*. Jan. 2007. ISBN: 9783540251415. DOI: 10.1007/978-3-540-30299-5.

[45] Douglas L Dwoyer, M Yousuff Hussaini, and Robert G Voigt. *Theoretical approaches to turbulence*. Vol. 58. Springer Science & Business Media, 2012.

[46] Huang Fang. "Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem". In: *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE. 2015, pp. 820–824.

[47] Philippe Spalart and Steven Allmaras. "A one-equation turbulence model for aerodynamic flows". In: *30th aerospace sciences meeting and exhibit*. 1992, p. 439.

[48] Brian Edward Launder and Bahrat I Sharma. "Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc". In: *Letters in heat and mass transfer* 1.2 (1974), pp. 131–137.

[49] Florian R Menter. *Improved two-equation k-omega turbulence models for aerodynamic flows*. Tech. rep. 1992.

[50] Georgi Kalitzin et al. "Near-wall behavior of RANS turbulence models and implications for wall functions". In: *Journal of Computational Physics* 204.1 (2005), pp. 265–291.

[51] John D Denton. "Some limitations of turbomachinery CFD". In: *Turbo Expo: Power for Land, Sea, and Air*. Vol. 44021. 2010, pp. 735–745.

[52] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. "Turbulence modeling in the age of data". In: *Annual Review of Fluid Mechanics* 51 (2019), pp. 357–377.

[53] Paul Kunzemann, Georg Jacobs, and Ralf Schelenz. "Application of CPS within wind energy—current implementation and future potential". In: *Industrial Internet of Things*. Springer, 2017, pp. 647–670.

[54] Ke-Sheng Wang, Vishal S Sharma, and Zhen-You Zhang. "SCADA data based condition monitoring of wind turbines". In: *Advances in Manufacturing* 2.1 (2014), pp. 61–69.

[55] Lorenzo Tieghi et al. "A Machine-Learnt Wall Function for Rotating Diffusers". In: *Journal of Turbomachinery* 143.8 (2021).

[56] Gino Angelini, Alessandro Corsini, and Sergio Lavagnoli. "Machine-learnt topology of complex tip geometries in gas turbine rotors". In: *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy* 235.3 (2021), pp. 383–392.

[57] Yuri Frey Marioni et al. "A machine learning approach to improve turbulence modelling from DNS data using neural networks". In: *International Journal of Turbomachinery, Propulsion and Power* 6.2 (2021), p. 17.

[58] Wihan Booyse, Daniel N Wilke, and Stephan Heyns. "Deep digital twins for detection, diagnostics and prognostics". In: *Mechanical Systems and Signal Processing* 140 (2020), p. 106612.

[59] Eleonora Arena et al. "Anomaly Detection in Photovoltaic Production Factories via Monte Carlo Pre-Processed Principal Component Analysis". In: *Energies* 14.13 (2021). ISSN: 1996-1073. DOI: 10.3390/en14133951. URL: https://www.mdpi.com/1996-1073/14/13/3951.

[60] Eric Stefan Miele, Fabrizio Bonacina, and Alessandro Corsini. "Deep anomaly detection in horizontal axis wind turbines using Graph Convolutional Autoencoders for Multivariate Time series". In: *Energy and AI* 8 (2022), p. 100145. ISSN: 2666-5468. DOI: https://doi.org/10.1016/j.egyai.2022.100145. URL: https://www.sciencedirect.com/science/article/pii/S2666546822000076.

[61] Zhifeng Liu et al. "Data super-network fault prediction model and maintenance strategy for mechanical product based on digital twin". In: *Ieee Access* 7 (2019), pp. 177284–177296.

[62] Gino Angelini et al. "A Multidimensional Extension of Balje Chart for Axial Flow Turbomachinery Using Artificial Intelligence-Based Meta-Models". In: *Journal of Engineering for Gas Turbines and Power* 141.11 (2019), p. 111012.

[63] Mohd Dasuki Yusoff et al. "A hybrid k-means-GMM machine learning technique for turbomachinery condition monitoring". In: *MATEC Web of Conferences*. Vol. 255. EDP Sciences. 2019, p. 06008.

[64] Fei Tao et al. "Digital twin driven prognostics and health management for complex equipment". In: *Cirp Annals* 67.1 (2018), pp. 169–172.

[65] Justin Flett and Gary M Bone. "Fault detection and diagnosis of diesel engine valve trains". In: *Mechanical Systems and Signal Processing* 72 (2016), pp. 316–327.

[66] Francesco Aldo Tucci, Giovanni Delibra, and Alessandro Corsini. "Development of a data-driven model for turbulent heat transfer in turbomachinery". In: *E3S Web of Conferences*. Vol. 197. EDP Sciences. 2020, p. 11006.

[67] Eurika Kaiser et al. "Cluster-based reduced-order modelling of a mixing layer". In: *Journal of Fluid Mechanics* 754 (2014), pp. 365–414.

[68] Deepankar Singh et al. "Aircraft Engine Reliability Analysis using Machine Learning Algorithms". In: *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)*. IEEE. 2020, pp. 443–448.

[69] Gino Angelini et al. "On surrogate-based optimization of truly reversible blade profiles for axial fans". In: *Designs* 2.2 (2018), p. 19.

[70] Jin-Long Wu, Heng Xiao, and Eric Paterson. "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework". In: *Physical Review Fluids* 3.7 (2018), p. 074602.

[71] Jason W Osborne. *Best practices in quantitative methods*. Sage, 2008.

[72] Jason Osborne. “Improving your data transformations: Applying the Box-Cox transformation”. In: *Practical Assessment, Research, and Evaluation* 15.1 (2010), p. 12.

[73] George EP Box and David R Cox. “An analysis of transformations”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 26.2 (1964), pp. 211–243.

[74] In-Kwon Yeo and Richard A Johnson. “A new family of power transformations to improve normality or symmetry”. In: *Biometrika* 87.4 (2000), pp. 954–959.

[75] Jacob Benesty et al. “Pearson correlation coefficient”. In: *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[76] Florian Wetschoreck, Tobias Krabel, and Surya Krishnamurthy. *8080labs/ppscore: zenodo release*. 2020.

[77] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[78] J Gauss. “Combinationis observationum erroribus minimis obnoxiae”. In: *Gottingen: University of Gottingen* (1825).

[79] Raymond H Myers, Douglas C Montgomery, and Christine M Anderson-Cook. *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons, 2016.

[80] Gilles Louppe. “Understanding random forests: From theory to practice”. In: *arXiv preprint arXiv:1407.7502* (2014).

[81] Tom Dietterich. “Overfitting and undercomputing in machine learning”. In: *ACM computing surveys (CSUR)* 27.3 (1995), pp. 326–327.

[82] Robin Genuer. “Variance reduction in purely random forests”. In: *Journal of Nonparametric Statistics* 24.3 (2012), pp. 543–562.

[83] Gérard Biau and Erwan Scornet. “A random forest guided tour”. In: *Test* 25.2 (2016), pp. 197–227.

[84] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[85] Joseph O Ogutu, Hans-Peter Piepho, and Torben Schulz-Streeck. “A comparison of random forests, boosting and support vector machines for genomic selection”. In: *BMC proceedings*. Vol. 5. 3. BioMed Central. 2011, pp. 1–5.

[86] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[87] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[88] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).

[89] John A Hartigan and Manchek A Wong. “Algorithm AS 136: A k-means clustering algorithm”. In: *Journal of the royal statistical society. series c (applied statistics)* 28.1 (1979), pp. 100–108.

[90] Bahman Bahmani et al. “Scalable k-means++”. In: *arXiv preprint arXiv:1203.6402* (2012).

[91] Sergei Vassilvitskii and David Arthur. “k-means++: The advantages of careful seeding”. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 2006, pp. 1027–1035.

[92] Douglas A Reynolds. “Gaussian mixture models.” In: *Encyclopedia of biometrics* 741.659-663 (2009).

[93] Arthur P Dempster, Nan M Laird, and Donald B Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[94] Gerda Claeskens, Nils Lid Hjort, et al. “Model selection and model averaging”. In: *Cambridge Books* (2008).

[95] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[96] Andrzej Maćkiewicz and Waldemar Ratajczak. “Principal components analysis (PCA)”. In: *Computers & Geosciences* 19.3 (1993), pp. 303–342.

[97] Hervé Abdi and Lynne J Williams. “Principal component analysis”. In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[98] Wen Wu, DL Massart, and S De Jong. “The kernel PCA algorithms for wide data. Part I: theory and algorithms”. In: *Chemometrics and Intelligent Laboratory Systems* 36.2 (1997), pp. 165–172.

[99] Hervé Abdi. “Partial least squares regression and projection on latent structure regression (PLS Regression)”. In: *Wiley interdisciplinary reviews: computational statistics* 2.1 (2010), pp. 97–106.

[100] W Svante, M Sjöström, and Lennart Erikson. “Partial least squares projections to latent structures (PLS) in chemistry”. In: *Encycl. Comput. Chem.* 3 (2002).

[101] Roman Rosipal and Nicole Krämer. “Overview and recent advances in partial least squares”. In: *International Statistical and Optimization Perspectives Workshop" Subspace, Latent Structure and Feature Selection"*. Springer. 2005, pp. 34–51.

[102] Pierre Baldi. “Autoencoders, unsupervised learning, and deep architectures”. In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings. 2012, pp. 37–49.

[103] Henry J Kelley. “Gradient theory of optimal flight paths”. In: *Ars Journal* 30.10 (1960), pp. 947–954.

[104] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[105] Alessandro Corsini et al. “Aeroacoustic assessment of leading edge bumps in industrial fans”. In: *Proceedings of the International Conference on Fan Noise, Technology and Numerical Methods*. 2015.

[106] Lucio Cardillo et al. “A numerical investigation into the aerodynamic effect of pressure pulses on a tunnel ventilation fan”. In: *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy* 228.3 (2014), pp. 285–299.

[107] Alessandro Corsini, Giovanni Delibra, and Anthony G Sheard. “The application of sinusoidal blade-leading edges in a fan-design methodology to improve stall resistance”. In: *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy* 228.3 (2014), pp. 255–271.

[108] Tze Pei Chong and Phillip F Joseph. “An experimental study of airfoil instability tonal noise with trailing edge serrations”. In: *Journal of Sound and Vibration* 332.24 (2013), pp. 6335–6358.

[109] Florian Krömer and Stefan Becker. “Experimental investigation of the sound reduction by leading edge serrations on a flat-plate axial fan”. In: *2018 AIAA/CEAS Aeroacoustics Conference*. 2018, p. 2955.

[110] Tze Pei Chong et al. “On the effect of leading edge serrations on aerofoil noise production”. In: *2018 AIAA/CEAS Aeroacoustics Conference*. 2018, p. 3289.

[111] Till M Biedermann et al. “Effect of inflow conditions on the noise reduction through leading edge serrations”. In: *AIAA Journal* 57.9 (2019), pp. 4104–4109.

[112] Till M Biedermann et al. “Statistical–empirical modeling of airfoil noise subjected to leading-edge serrations”. In: *AIAA Journal* 55.9 (2017), pp. 3128–3142.

[113] Florian Krömer, Andreas Renz, and Stefan Becker. “Experimental investigation of the sound reduction by leading-edge serrations in axial fans”. In: *AIAA Journal* 56.5 (2018), pp. 2086–2090.

[114] Florian Krömer, Felix Czwielong, and Stefan Becker. “Experimental investigation of the sound emission of skewed axial fans with leading-edge serrations”. In: *AIAA Journal* 57.12 (2019), pp. 5182–5196.

[115] Gino Angelini et al. “Identification of Losses in Turbomachinery With Machine Learning”. In: *Turbo Expo: Power for Land, Sea, and Air*. Vol. 84058. American Society of Mechanical Engineers. 2020, V001T10A008.

[116] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[117] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[118] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization.” In: *Journal of machine learning research* 13.2 (2012).

[119] Florent Monjalet and Thomas Leibovici. “Predicting file lifetimes with machine learning”. In: *International Conference on High Performance Computing*. Springer. 2019, pp. 288–299.

[120] İbrahim Zeki Akyurt, Yusuf Kuvvetli, and Muhammet Deveci. "Enterprise Resource Planning in the Age of Industry 4.0: A General Overview". In: *Logistics 4.0* (2020), pp. 178–185.

[121] Alok Raj et al. "Barriers to the adoption of industry 4.0 technologies in the manufacturing sector: An inter-country comparative perspective". In: *International Journal of Production Economics* 224 (2020), p. 107546.

[122] Harpreet Singh. "Big data, industry 4.0 and cyber-physical systems integration: A smart industry context". In: *Materials Today: Proceedings* 46 (2021), pp. 157–162.

[123] Xindong Wu et al. "Data mining with big data". In: *IEEE transactions on knowledge and data engineering* 26.1 (2013), pp. 97–107.

[124] Philip Russom et al. "Big data analytics". In: *TDWI best practices report, fourth quarter* 19.4 (2011), pp. 1–34.

[125] Andy S Alic et al. "BIGSEA: A Big Data analytics platform for public transportation information". In: *Future generation computer systems* 96 (2019), pp. 243–269.

[126] Ramón Fernando Colmenares-Quintero et al. "Big Data analytics in Smart Grids for renewable energy networks: Systematic review of information and communication technology tools". In: *Cogent Engineering* 8.1 (2021), p. 1935410.

[127] Arfan Majeed et al. "A big data-driven framework for sustainable and smart additive manufacturing". In: *Robotics and Computer-Integrated Manufacturing* 67 (2021), p. 102026.

[128] Alessandro Corsini et al. "Cascade With Sinusoidal Leading Edges: Identification And Quantification of Deflection With Unsupervised Machine Learning". In: *Turbo Expo: Power for Land, Sea, and Air*. Vol. 84898. American Society of Mechanical Engineers. 2021, V001T10A006.

[129] Eric Stefan Miele, Fabrizio Bonacina, and Alessandro Corsini. "Deep anomaly detection in horizontal axis wind turbines using graph convolutional autoencoders for multivariate time series". In: *Energy and AI* 8 (2022), p. 100145.

[130] Mumtaz Karatas, Ilknur Karacan, and Hakan Tozan. "An integrated multicriteria decision making methodology for health technology assessment". In: *European Journal of Industrial Engineering* 12.4 (2018), pp. 504–534.

[131] Yanqing Yang and Xing Song. "Research on face intelligent perception technology integrating deep learning under different illumination intensities". In: *Journal of Computational and Cognitive Engineering* (2022), pp. 32–36.

[132] C Boursier Niutta et al. "Surrogate modeling in design optimization of structures with discontinuous responses". In: *Structural and Multidisciplinary Optimization* 57.5 (2018), pp. 1857–1869.

[133] Bogdan C Cernat et al. "Experimental and numerical investigation of optimized blade tip shapes—Part I: Turbine rainbow rotor testing and numerical methods". In: *Journal of Turbomachinery* 141.1 (2019), p. 011006.

[134] Till M Biedermann, Frank Kameier, and Christian O Paschereit. “Successive aeroacoustic transfer of leading edge serrations from single airfoil to low-pressure fan application”. In: *Journal of Engineering for Gas Turbines and Power* 141.10 (2019).

[135] Simin Shen et al. “Research on Cavitation Flow Dynamics and Entropy Generation Analysis in an Axial Flow Pump”. In: *Journal of Sensors* 2022 (2022).

[136] Sergio Lavagnoli et al. “High-fidelity rotor gap measurements in a short-duration turbine rig”. In: *Mechanical Systems and Signal Processing* 27 (2012), pp. 590–603.

[137] Jonathan Zalger. *Application of variational autoencoders for aircraft turbomachinery design*. Tech. rep. Technical report, 2017.

[138] Julie Pongetti et al. “Using Autoencoders and Output Consolidation to Improve Machine Learning Models for Turbomachinery Applications”. In: *Turbo Expo: Power for Land, Sea, and Air*. Vol. 84935. American Society of Mechanical Engineers. 2021, V02DT36A018.

[139] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[140] Adriano Moreira and Maribel Yasmina Santos. “Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points”. In: (2007).

[141] Haitao Zhao, Pong Chi Yuen, and James T Kwok. “A novel incremental principal component analysis and its application for face recognition”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 36.4 (2006), pp. 873–886.

[142] Vittorio Verda and Francesco Colella. “Primary energy savings through thermal storage in district heating networks”. In: *Energy* 36.7 (2011), pp. 4278–4286.

[143] Dominik Franjo Dominković et al. “Technical, economic and environmental optimization of district heating expansion in an urban agglomeration”. In: *Energy* 197 (2020), p. 117243.

[144] Hanmin Cai et al. “Demand side management in urban district heating networks”. In: *Applied energy* 230 (2018), pp. 506–518.

[145] GG Alexandrov and AS Ginzburg. “Anthropogenic impact of Moscow district heating system on urban environment”. In: *Energy Procedia* 149 (2018), pp. 161–169.

[146] Richard Newell et al. “Global energy outlook 2021: pathways from Paris”. In: *Resources for the Future Report* (2021), pp. 11–21.

[147] Stéphanie Bouckaert et al. “Net Zero by 2050: A Roadmap for the Global Energy Sector”. In: (2021).

[148] Marderos Ara Sayegh et al. “Heat pump placement, connection and operational modes in European district heating”. In: *Energy and Buildings* 166 (2018), pp. 122–144.

[149] Fabian Levihn. “CHP and heat pumps to balance renewable power production: Lessons from the district heating network in Stockholm”. In: *Energy* 137 (2017), pp. 670–678.

[150] Haichao Wang et al. “Modelling and optimization of CHP based district heating system with renewable energy production and energy storage”. In: *Applied Energy* 159 (2015), pp. 401–421.

[151] IEA. *District Heating*. Report. Paris: IEA, 2021.

[152] PL Mendonça et al. “Detection and modelling of incipient failures in internal combustion engine driven generators using electrical signature analysis”. In: *Electric Power Systems Research* 149 (2017), pp. 30–45.

[153] Qinsheng Yun, Chuanqing Zhang, and Tianyuan Ma. “Fault diagnosis of diesel generator set based on deep believe network”. In: *Proceedings of the 2nd International Conference on Artificial Intelligence and Pattern Recognition*. 2019, pp. 186–190.

[154] Oihane C. Basurko and Zigor Uriondo. “Condition-Based Maintenance for medium speed diesel engines used in vessels in operation”. In: *Applied Thermal Engineering* 80 (2015), pp. 404–412.

[155] Masoud Aliramezani, Charles Robert Koch, and Mahdi Shahbakhti. “Modeling, diagnostics, optimization, and control of internal combustion engines via modern machine learning techniques: A review and future directions”. In: *Progress in Energy and Combustion Science* 88 (2022), p. 100967.

[156] Charis Ntakolia et al. “Machine learning applied on the district heating and cooling sector: a review”. In: *Energy Systems* (2021), pp. 1–30.

[157] Gideon Mbiydzenyuy et al. “Opportunities for machine learning in district heating”. In: *Applied Sciences* 11.13 (2021), p. 6112.

[158] Jerzy Baranowski et al. “Bayesian fault detection and isolation using Field Kalman Filter”. In: *EURASIP Journal on Advances in Signal Processing* 2017.1 (2017), pp. 1–11.

[159] Daniel Jung. “Data-driven open-set fault classification of residual data using Bayesian filtering”. In: *IEEE Transactions on Control Systems Technology* 28.5 (2020), pp. 2045–2052.

[160] Piotr Czech and Jerzy Mikulski. “Application of Bayes classifier and entropy of vibration signals to diagnose damage of head gasket in internal combustion engine of a car”. In: *International Conference on Transport Systems Telematics*. Springer. 2014, pp. 225–232.

[161] Faye Zhang et al. “Internal combustion engine fault identification based on FBG vibration sensor and support vector machines algorithm”. In: *Mathematical Problems in Engineering* 2019 (2019).

[162] SN Dandare and SV Dudul. “Support vector machine based multiple fault detection in an automobile engine using sound signal”. In: *Journal of Electronic and Electrical Engineering, ISSN* (2012), pp. 0976–8106.

[163] Jannis Tautz-Weinert and Simon J Watson. “Using SCADA data for wind turbine condition monitoring–a review”. In: *IET Renewable Power Generation* 11.4 (2017), pp. 382–394.

[164] Yinglai Liu et al. “Fault diagnosis and prediction method for valve clearance of diesel engine based on linear regression”. In: *2020 Annual Reliability and Maintainability Symposium (RAMS)*. IEEE. 2020, pp. 1–6.

[165] David J Bryg, George Mink, and Link C Jaw. “Combining lead functions and logistic regression for predicting failures on an aircraft engine”. In: *Turbo Expo: Power for Land, Sea, and Air*. Vol. 43123. 2008, pp. 19–26.

[166] Chengtao Cai, Xiangyu Weng, and Chuanbin Zhang. “A novel approach for marine diesel engine fault diagnosis”. In: *Cluster computing* 20.2 (2017), pp. 1691–1702.

[167] Weizhong Yan. “Application of random forest to aircraft engine fault diagnosis”. In: *The Proceedings of the Multiconference on" Computational Engineering in Systems Applications"*. Vol. 1. IEEE. 2006, pp. 468–475.

[168] Daniel Maraini et al. “Development of a Data-driven Model for Marine Gas Turbine (MGT) Engine Health Monitoring”. In: *Annual Conference of the Prognostics and Health Management Society*. 2018.

[169] Mohammad Braei and Sebastian Wagner. “Anomaly detection in univariate time-series: A survey on the state-of-the-art”. In: *arXiv preprint arXiv:2004.00433* (2020).

[170] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[171] Andrey Kolmogorov. “On the Shannon theory of information transmission in the case of continuous signals”. In: *IRE Transactions on Information Theory* 2.4 (1956), pp. 102–108.

[172] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *CoRR* abs/1603.02754 (2016). arXiv: 1603.02754. URL: http://arxiv.org/abs/1603.02754.

[173] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[174] James Bergstra and Yoshua Bengio. “Random Search for Hyper-Parameter Optimization.” In: *J. Mach. Learn. Res.* 13 (2012), pp. 281–305. URL: http://dblp.uni-trier.de/db/journals/jmlr/jmlr13.html#BergstraB12.

[175] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.