Sapienza Università di Roma

Doctoral Thesis

Machine learning in Industrial Turbomachinery: development of new framework for design, analysis and optimization

PhD Candidate:
Gino Angelini

Supervisor:
Prof. Alessandro Corsini

A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Industrial and Management Engineering

in

October 2019

SAPIENZA UNIVERSITÀ DI ROMA

Abstract

Department of Mechanical and Aerospace Engineering

Doctor of Philosophy

Machine learning in Industrial Turbomachinery: development of new framework for design, analysis and optimization

by Gino ANGELINI

In the age of Industry4.0, turbomachinery community is facing a significant effort in re-thinking all manufacturing activities. A new industrial paradigm is developing, which encourages the introduction of Big-Data, Internet of Things (IoT) and Artificial Intelligence (AI) into the production and manufacturing environment to follow the market trend of innovation and strong customization of product. This thesis aims to improve the classical approach to turbomachinery design, analysis and optimization, which relays to empiric procedures or computational intensive CFD based optimizations, implementing new framework and strategies based on the use of Machine Learning and Deep Learning, the successful sub-fields of AI, recognized as crucial technologies to stay competitive in this dynamic landscape. The introduction of learning algorithms changes the global structure of classic approach determining modern design support tools. The introduction of regression models facilitates the exploration of design space providing easy-to-run analytical models. Regression models can also be used as cost functions in multi-objective optimization problems avoiding the use of time consuming simulations. The application of unsupervised learning methods to turbomachine performance data extracts new insights to improve the designer understanding of blade-flow interaction and allows a better formulation of specific problem. The design space exploration artificial neural network based allowed the creation of multi-dimensional Balje charts, enriched using correlations between the main interesting geometric parameters and the rotor performance, which guide designer in preliminary blade geometry selection. Surrogate model based optimization allowed reducing the computational cost required by canonical approach producing reliable set of better solutions in comparison with the standard optimization. Application of unsupervised learning techniques allows developing a new parametrization strategy based on statistical shape representation of complex three dimensional surfaces. This work can be ideally divided into two parts. The first reviews the state-of-the-art of artificial intelligence techniques highlighting the role of machine learning in supporting design, analysis and optimization for turbomachinery community. Second part reports the development of a set of strategies to exploit metamodeling algorithms improving the classical design approach.

Publications

- Angelini G., Corsini A., Delibra G., Tieghi L., "A Multidimensional Extension of Balje Chart for Axial Flow Turbomachinery Using Artificial Intelligence-Based Meta-Models." Journal of Engineering for Gas Turbines and Power 141.11, 2019.
- Angelini G., Bonanni T., Corsini A., Delibra G., Tieghi L., Volponi D., Venturini P., "A Methodology for Optimal Aerofoil Selection for Axial Fans Subjected to particle-laden flows“, abstract submitted to ETC 2019.
- Angelini G., Bonanni T., Corsini A., Delibra G., Tieghi L., Volponi D., "Application of IA-driven turbulence modelling to NACA aerofoil with leading edge bumps to control stall dynamics “, abstract submitted to ETC 2019.
- Angelini G., Bonanni T., Corsini A., Delibra G., Tieghi L., Volponi D., "On Surrogate-Based Optimization of Truly Reversible Blade Profiles for Axial Fans." Designs 2.2: 19, 2019.
- Angelini G., Corsini A., Delibra G., Giovannelli M., Lucherini G., Minotti S., Rossin S., Tieghi L. "Meta-modelling of gas-leak in gas turbine enclosures”. ASME GT2019-91198, 2019.
- Angelini G., Corsini A., Delibra G., Giovannelli M., Lucherini G., Minotti S., Rossin S., Tieghi L. "Identification of poorly ventilated zones in gas-turbine enclosures with machine learning." ASME GT2019-91199, 2019.
- Angelini G., Bonanni T., Corsini A., Delibra G., Tieghi L., Volponi D., “A metamodel for deviation in 2D cascade with variable stagger, solidity and reversible profiles”, ASME IGTI TurboExpo 2018, June 11-15, Oslo, Norway, GT2018-76363.
- Angelini G., Bonanni T., Corsini A., Delibra G., Tieghi L., Volponi D., “Effects of fan inflow distortion on heat exchange in air-cooled condenser. Unsteady computations with synthetic blade model”, ASME IGTI TurboExpo 2018, June 11-15, Oslo, Norway, GT2018-76518.
- Angelini G., Bonanni T., Corsini A., Delibra G., Tieghi L., Volponi D., “Noise Reduction Of A Large Axial Flow Fan For Csp Air-Cooled Condensers”, submitted at FAN, International Conference on Fan Noise, Aerodynamics, Applications and Systems 2018, April 18-20, Darmstadt, Geramny.
- Angelini G., Bonanni T., Corsini A., Delibra G., Tieghi L., Volponi D., “Rapid Prototyping And Testing Of A Small Scale Fan For Csp Power Plant Applications”, submitted at FAN, International Conference on Fan Noise, Aerodynamics, Applications and Systems 2018, April 18-20, Darmstadt, Geramny.
- Angelini G., Bonanni T., Corsini A., Delibra G., Tieghi L., Volponi D., “Optimization of an axial fan for air cooled condensers”, ATI 2017.
- Angelini G., “A tool for preliminary design and selection of industrial fans”, student poster for IGTI ASME Turbo Expo 2017

Contents

List of Abbreviations

ANN	Artificial Neural Network
CCD	Central Composite Design
CCI	Central Composite Inscribed
CFD	Computational Fluid Dynamic
CP	Composed Parameter
CSP	Concentrated Solar Power
DOE	Design Of Experiments
DFR	Direct Fitness Replacement
EAs	Evolutionary Algorithms
EDA	Exploratory Data Analysis
Eff	Efficiency
FNDS	Fast Non Dominated Sorting
GAs	Genetic Algorithms
GBR	Gradient Boosting Regressor
H	Hyperarea
HPT	High Pressure Turbine
IFR	Indirect Fitness Replacement
IQR	Interquartile Range
LSM	Least Square Method
LV	Latent Variable
MB	Meta-model Based
MLP	Multilayer Perceptron
MOEA	Multi-Objectives Evolutionary Algorithms
MOOPs	Multi-Objectives Optimization Problems
MS	Max Sum
MSE	Mean Square Error
NEC	No Evolution Control
NSGA	Non-dominated Sorting Genetic Algorithm
ONVGR	Overall Non-dominated Vector Generation Ratio
OS	Overall Pareto Spread
PC	Prediction Capability
PCA	Principal Component Analysis
PLS	Projection to Latent Structure
SMs	Surrogate Models
SP	Spacing
THL	Tip Heat Load
TLV	Tip Leakage Vortex
Tr	Threshold
UPV	Upper Passage Vortex

List of Symbols

A	Tip area portion	-
a	Principal component number	-
AoA	Angle of attack	deg
AoC	Angle of cascade	deg
AR	Blade aspect ratio; $AR = h/l$	-
b	Cell status	-
C	Absolute velocity	$\mathrm{m\,s^{-1}}$
C_1	Inlet velocity	$\mathrm{m\,s^{-1}}$
C_D	Drag coefficient	-
C_L	Lift coefficient	-
C_LS	Specified lift coefficient	-
Dt	Fan diameter	m
Ds	Specific diameter	-
F	Factors	-
FA	Filling factor A	
g	Array containing aerofoil geometry	-
h	Blade height	m
HR	Hub to tip ratio; $HR = r_{hub}/r_{tip}$	-
l	chord length	m
Ns	Specific speed	-
O	Objectives	-
P	Pressure rise	Pa
Pt	Initial population	-
P_{pe}	Pressure rise at peak efficiency	Pa
P_{pp}	Pressure rise at peak pressure	Pa
Q	Volume flow rate	$\mathrm{m^3\,s^{-1}}$
Q_{pe}	Volume flow rate at peak efficiency	$\mathrm{m^3\,s^{-1}}$
Q_{pp}	Volume flow rate at peak pressure	$\mathrm{m^3\,s^{-1}}$
Qt	Offspring population	-
Q_1	Lower quartile	-
Q_3	Upper quartile	-
r	Blade radius	m
Re	Reynolds number; $Re = U(r)l/\mu$	-
Rt	Global population	-
s	LSM number of factors	-
t	Blade spacing	m
U	PCA score matrix	-
U_{tip}	Tip speed of the blade	$\mathrm{m\,s^{-1}}$
V	PCA loading matrix	-
Var	Dataset performance variables	-
Var^{YJ}	Var Yeo-Johnson transformed	-
$\tilde{V}ar$	Var after dimension reduction	-

w	PLS loading	-
W	Relative velocity	$\mathrm{m\,s^{-1}}$
X	Dataset matrix	-
X_{ai}	x-coordinates of B-Spline parametrization	-
Y	PLS output matrix	-
Y_{ai}	y-coordinates of B-Spline parametrization	-
y^+	Wall distance	m
z	Blade number	-
α	Stall margin	-
β	Flow angle	deg
γ	Aerofoil stagger	deg
Γ	Pitch angle	deg
ϵ	Aerodynamic efficiency	-
ϕ	Global flow coefficient	-
ψ	Global work coefficient	-
θ	Fluid deflection	deg
λ	Linear activation function	-
ν	Star point distance	-
σ	Sigmoid activation function/blade solidity	-
$\tilde{\nu}$	Spalart-Allmaras variable	$\mathrm{m\,s^{-1}}$
η	Total efficiency	-
ω	Angular velocity	rad
ζ	LSM regressors	-

Dedicated to my Family

Introduction

We are living the digital age and the changing role of engineers. In 2011 the term Industry4.0 was coined in Europe to identify a new industrial paradigm enabled by the introduction of smart technologies and artificial intelligence into the production and manufacturing environment. Since then with the term Industry4.0 we refer to the branch of fourth industrial revolution that promotes the use of Big-Data, Internet of Things (IoT) and Artificial Intelligence (AI) to create an environment whereby smart technologies can communicate and share data to the end of analyze and understand production issues and to solve them. Regardless the industry sectors, this trend of innovation and strong customization of product is involving many communities. Deeply immersed in this "water", we will analyze how the introduction of AI technologies can determine significant improvements in design and optimization procedure of Industrial Turbomachines. Typically turbomachine design is an expensive, either in terms of computational or experimental effort, procedure that involves several activities inherently multi-disciplinary. A classic design approach relies on elements from aerospace design and previous turbomachinery studies for the blade design and performance analysis, element of Computation Fluid Dynamics (CFD) for a detailed analysis of flow behavior (losses, unsteady behavior, noise etc.) and elements from manufacture and experiments for the final laboratory tests. In a world of increasingly complex and integrated problems, this classical approach, routed in the 1950s and condensate in the know-how and intellectual property of traditional manufacturers, results unable to face the challenges of flexibility and deeply customization of product required in the era of Industry4.0. Advancements in simulation software coupled with high-performance computing provide a venue for analyzing complex multi-physics systems never thought possible a decade ago.

In such scenario, the main objective of this thesis is revisiting the state-of-the-art approach to turbomachinery design, analysis and optimization implementing new framework and strategies inspired by elements based paradigm in Industry4.0.

Since there is not an univocal definition and because the term Industry4.0, until now, has not yet been conclusively defined, is not easy to find an exhaustive definition, however we can identify the main three features behind this word:

- Digitization of vertical and horizontal value chains: The implementation of such networkings determine processes that are flexible and respond rapidly to changes in customer demands.

- Digitization of product and service offering: Innovation in engineering involve design, development and manufacturing processes. The introduction of new method of data collection and analysis enables the creation of new design procedures utilizing a large amount of collected informations (big-data).

- Acceleration through exponential technologies: The introduction of smart technologies, in particular Artificial Intelligence (AI), enables to reduce costs, increase flexibility and customize products.

Despite the extensive number of components that are contributing to radical shift in industrial process flows (i.e., IoT, Cloud computing, Augmented reality, rapid prototyping), we will focus on the role played by data analysis, machine learning and deep learning, as sub-fields of AI. The role data scientists play in engine design, manufacture and maintenance will continue to grow. They can quickly analyze data and use it to make real-time decisions. They can solve problems rapidly avoid relying on old labor-intensive manual processes. In such landscape a key role is played by Machine Learning. Even if Artificial Intelligence has its origins in 1950s, the early AI algorithms were just applications of long chains of hard coded rules crafted to solve tasks easy to describe formally and then to model (i.e., the chess game). This approach, known as symbolic AI, failed when the task to solve was harder to be formalized, as in the case of real world problems. With Machine Learning we overcome the limits of symbolic AI. Machine learning defines a new programming paradigm in which the learning algorithm is trained instead of programmed. Using huge dataset related to a specific task, users input data and expected answers, while the algorithm finds hidden structures inside the examples and outcomes rules for automating the task. To this end machine learning automatically search transformations that move data into more useful representation for a given task. This concept is empathized by Deep Learning, in which the representation is expressed in terms of successive layers of increasingly relevant representations. Nowadays machine learning and deep learning have been recognized as a crucial technology to stay competitive in this industrial changing landscape. The recent success is due to the availability of massive amount of data usable during training phase and the increment of computational performance.

The introduction of machine learning and deep learning in the classic approach to Turbomachinery design and analysis changes the global structure of this process and provides modern tools for each intermediate design step.

Replacing computation intensive functions with approximated models (i.e., regression models) facilitates the exploration of Design Space and the selection of preliminary geometry avoiding the use of old and fragmented empiric rules.

Furthermore, such meta-models will also facilitate the use of advanced and nested optimization methods, such as those based on Evolutionary Algorithms. Surrogate-based optimization has proven to be effective and reliable tool to quickly find local and global optima.

The application of unsupervised learning method, as dimensionality reduction and clustering, to turbomachine performance data extracts new information to improve the designer comprehension of flow behavior and allows a better formulation of specific problem (i.e.,the selection of different constraints and different design variables or elimination of unnecessary information reducing the dimensionality of the problem).

This is the arena in which this work, complementing more than 3 years of collaboration with the turbomachinery group DIMA Sapienza University of Rome, lies. The present document is divided into seven main chapters. The first four chapters describe the method used to implement new turbomachinery design strategies machine-learnt based. The last three chapters report three applications of such strategies and represent the core of this thesis work.

First chapter introduces the concept of learning algorithms clarifying the meaning behind the word "Artificial Intelligence", which in the early days has been subjected of intense media hype. In particular, it provides essential definitions of new programming paradigms introduced by the two AI sub-fields: Machine Learning and Deep Learning with a particular focus on their application in turbomachinery. This chapter also clarifies the reasons of the current success of Machine Learning.

Second chapter introduces the key concept of turbomachinery machine-learnt design. It firstly describes the classical approach to turbomachinery design and optimization including the many steps required from the preliminary design to the optimized blade geometry.

Successively, it analyzes the role played by machine learning and deep learning algorithms highlighting the advantages coming from their implementation in the classical approach.

Third chapter describes the machine learning branches starting from the notorious classification between unsupervised learning (i.e., clustering and dimensionality reduction) and supervised learning (i.e., classification and regression). The chapter deeply details all the machine learning and deep learning algorithms exploited in the turbomachinery applications, which are the core of the present work.

Fourth chapter introduces the key role played by data treatment before performing any machine learning analysis. It shows how exploratory data analysis (EDA) should be used to improve the quality and readability of data during the preprocessing phase and how this process deeply affects the result of any machine learning campaign.

After introducing the method, following chapters provide applicative examples of machine learnt applications to several aspects of design and analysis of industrial turbomachines. The fifth chapter provide a machine learnt strategy to explore the axial flow turbomachinery design space to build a set of multi-dimensional Balje charts, which will provide new procedures during the preliminary design phase. The chapter shows how create a dataset of tested rotor performance using a deviation meta-model artificial intelligence based and how use dimensionality reduction methods as principal component analysis and projection to latent structures to find correlation between rotor performance and enriched geometric and kinematic parameters. Such correlations will be used to improve the amount of informations achievable using the Balje chart.

The sixth chapter assesses how exploiting surrogate-based techniques in optimization problems concerning the aerofoils aerodynamic. It shows how meta-model techniques affect results of multi-objective optimization problem based on NSGA-II optimization algorithm, and how these meta-models should be exploited in an optimization test-bed in order to reduce the computational effort required by canonical optimization strategy. The object of this optimization will be the truly reversible axial fans largely employed in tunnel and metro ventilation systems, to supply and extract air from the tunnels.

Last chapter before conclusions reports a data driven based framework to develop a new statistical representation of complex tip surfaces in high-pressure turbine (HPT) stages. In particular, an existing database of optimized HPT blade tips has been exploited. The chapter shows how develop a new continuous parameterization of tip surface by means of principal component analysis and unsupervised learning algorithms (i.e., clustering methods).

All the developed and implemented codes, the blade designer, AxLab, the algorithms for machine learning, deep learning and surrogate model-based optimization are written in Python language (using scikit-learn library and TensorFlow). The numerical simulation needed are implemented in OpenFOAM finite volume solver written in C++., at exception of the simulations performed in the work reported in chapter 7 where a proprietary solver has been used.

Chapter 1

Learning in turbomachinery

"E questa è la vera regola come li speculatori delli effetti naturali hanno a procedere, e ancora che la natura cominci dalla ragione e termini nella sperienzia, a noi bisogna seguitare in contrario, cioè cominciando dalla sperienzia, e con quella investigare la ragione."

And this is the true rule as speculators of natural effects must proceed, and although nature starts from reason and terms in experience, we must continue to do the opposite, that is to commence with experience, and from this to proceed to investigate the reason.

Leonardo Da Vinci

Following the human being intuition, learning simply means to better accomplish a task over a period of time through the acquired experience. In the past few years, the development of algorithms capable of learning and solve real word problems has been the centre of thriving research and artificial intelligence (AI) has been subjected of intense media attention.

In this chapter we are going to provide essential definitions of artificial intelligence, machine learning and deep learning with a particular focus on their application in turbomachinery.

1.1 Fundamentals and classification

Due to its popularity and related media hype, there is some confusion about the world we mean mentioning artificial intelligence. Before speaking of machine learning algorithm and applications, we need to clearly answer the question on what artificial intelligence, machine learning and deep learning are and what is the relation between these different learning approaches. Figure 1.1 shows that deep learning is a sub-field of machine learning being both artificial intelligence approaches.

1.1.1 Artificial Intelligence

Artificial intelligence has its origin in 1950s, when a handful of computer scientists started asking whether computers could automate intellectual tasks normally performed by humans. In the broadest sense, AI field includes machine learning and deep learning, but also involves not learning algorithms.

The early AI algorithms were just a long chain of hard coded rules crafted by programmers, as for the game of chess programs. Chess game can be easily described by a brief list of formal rules. AI applications focused on tasks that are demanding for human beings but relatively simple for computers, being such tasks easy to describe formally and then

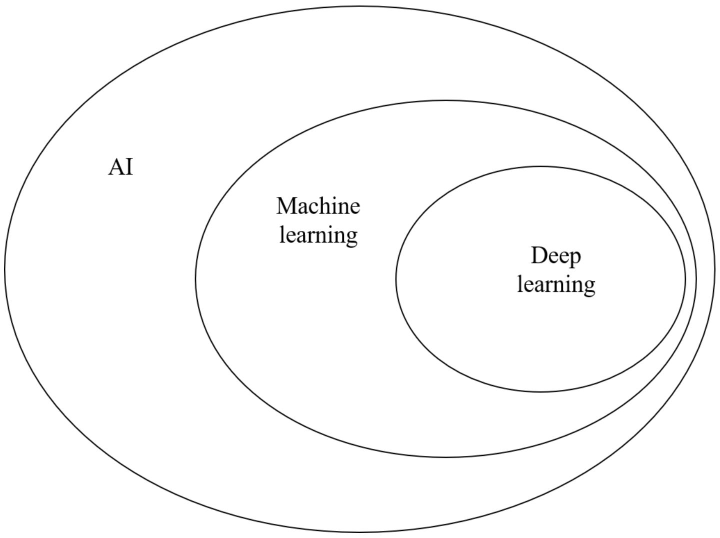


FIGURE 1.1: Artificial intelligence approaches

to model. Lasted with a fair amount of success until 1980s, in the first AI era experts tried replicating the human intelligence encoding a sufficiently broad set of rules. This approach is called symbolic AI. The archetype of symbolic AI requires users define rules and data to be processed according to these rules, while the algorithm provides the answers (Figure 1.2).

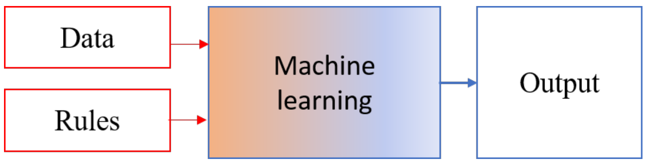


FIGURE 1.2: Classic programming paradigm

As guessed, symbolic AI has an intrinsic key limitation. The line between what a symbolic AI algorithm can or cannot do depends on how easily the problem can be formalized. A lot of real tasks, as image classification or transcribing human speech to text, are harder to describe formally so they cannot be achieved using a symbolic AI approach. The limit of early days AI has been exceeded replacing symbolic AI with machine learning.

1.1.2 Machine Learning

To overcame the difficulties encountered by systems relying on hard-coded knowledge become clear that AI systems need the ability to acquire their own knowledge by extracting patterns from row data. This capability is known as machine learning. Machine learning pushed learning algorithm beyond our capability to order it to perform: algorithm learns how performing a specific task, without hand crafted rules, by only looking at data.

With machine learning a new programming paradigm born in which users input data and expected answers, while the algorithm outcomes rules (see Figure 1.3). Rules can be successively used on new data (not included in the input set) producing new answers. A machine learning algorithms is trained instead of programmed. The training process involves the use of large amounts of data related to a specific task. During this process the algorithm finds hidden structures inside the example dataset that will be used to define rules for automating the task. Machine learning exploits several statistical tools, but it is different from statistics because machine learning deals with huge dataset (millions of

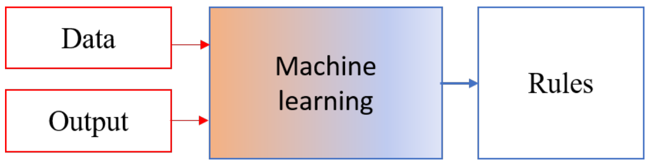


FIGURE 1.3: Machine learning paradigm

samples) unmanageable with classical statistical analysis (as Bayesian analysis). Moreover machine learning presents less mathematical theory than statistics resulting more engineering oriented.

Machine learning started to flourish in 1990s and quickly became popular and successful sub-field of artificial intelligence. Nowadays, an interpretation of the state-of-the-art distinguish between 5 different approaches to machine learning [1]:

- Symbolists – intelligence can be casted in a symbolic form. Machine learning (ML) is based on induction (i.e. decision trees)
- Connectionists – learning is cognitive and the goal is to reverse engineered human mind ML is backpropagation (i.e. ANN)
- Evolutionists – trust in natural selection and the goal is to discover the learning structure ML is genetic programming (i.e. EA or GA)
- Bayesian – knowledge is in the uncertainty. ML is a probabilistic inference
- Analogists – key ingredient is the identification of patter of similarity ML is a support vector machine

Irrespective to the approach, to program a machine learning algorithm we need a way to measure whether the algorithm is doing a good job. This condition is necessary to quantify the distance between the algorithm predicted output and the real output value. Moreover, the performance of machine learning algorithms depends heavily on the representation of the data they are given [2]. For instance, if the learning algorithm is used to predict the health condition of a patient, the algorithm will evaluate the information collected by the doctor into a representation of patient. Each piece of information passed to the algorithm is known as a feature. Many artificial intelligence tasks can be solved by designing the right set of feature to extract for each specific task. Some tasks, which may be difficult to accomplish using one representation, can became easy to solve with another [3]. Suppose the task consists dividing two categories of data, reported in a scatter plot, by drawing a straight line. In the plot reported on the left part of Figure 1.4 the task is impossible. Instead, in the plot reported on the right, in which the polar coordinates replaced the Cartesian ones to represent the same set of data, the task becomes easy to do [3].

The central problem in machine learning (and deep learning) is to learn useful representation of input data. Machine learning algorithms consist of automatically searching transformations, within a pre-defined space of possibility, that move data into more useful representation for a given task. In many real-world applications can be very difficult to obtain a new representation from row data can be as difficult as solving the original problem using a simple ML algorithm.

1.1.3 Deep Learning

Deep learning is a sub-field of machine learning in which the representations of data are expressed in terms of many simpler representations. Deep learning approach empathizes

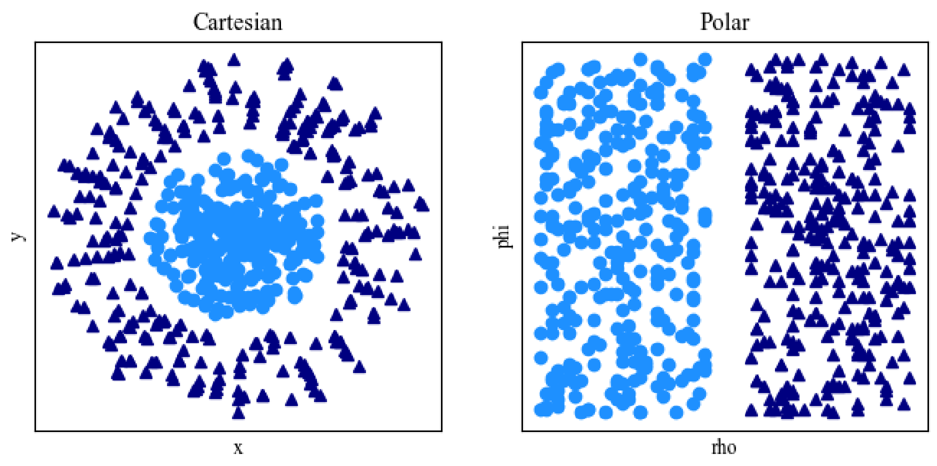


FIGURE 1.4: Influence of different representations

the concept of learning successive layers of increasingly relevant (and increasingly abstract for human user) representations of input data. Flowcharts reported in Figure 1.5 show the different parts in machine learning and deep learning. In this figure, colored boxes indicate components that have the capability to learn from data and its clear that, while machine learning is based on single representation of data, deep learning includes many representations in which the complexity gradually increases. To avoid misunderstandings,

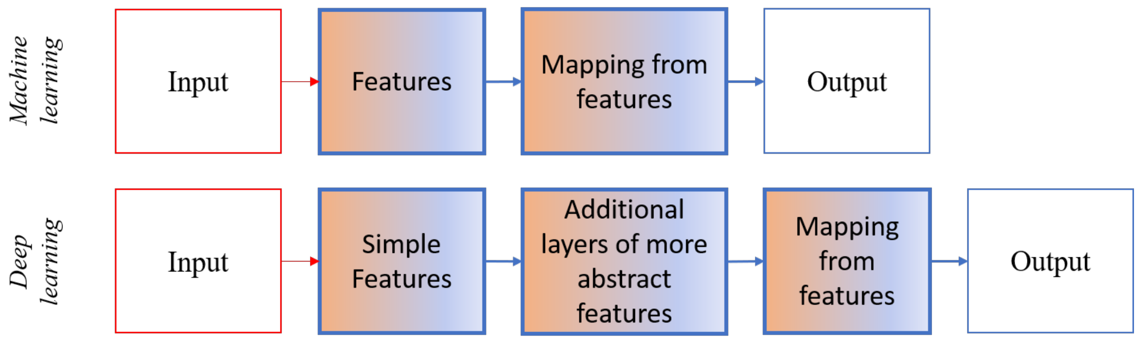


FIGURE 1.5: Flowchart of AI approaches

it's important to note that deep learning approach is only inspired to neurobiology, not imitates the human brain. The word "deep" in deep learning doesn't refers to a deeper comprehension of data, it stands for the idea of learning the hierarchy of concepts directly from raw data. The depth of the model refers to the number of layers contributing to the model response.

In deep learning, the many representation levels are obtained via models called neural networks, in particular: the feed-forward deep network or multilayer perceptron (MLP). The algorithm is a mathematical framework that transforms the input features into increasingly different representations of the original data. These representations are part of a multi-stage process in which the information goes through successive filters increasing its relation with the final result (as in a process of information-distillation). Each layer can be seen as a simple data transformation and the details of each transformation are learned by exposure to example dataset and stored into numbers called the layer's weights. The weights parametrize the transformation implemented in each network layer and during the learning phase the algorithm needs finding the weight values for all layers in the network.

A deep network contains millions of these parameters and changing the value of a single weigh affects the behaviour of all the network.

As already seen for machine learning, to evaluate the network performance we need to measure how far is the prediction from our expectation. To this end, each deep learning algorithm includes a loss function (also called objective function) that measures the distance between the network prediction and the true value of the example. The loss function associates a score to each prediction and this score is used in a feedback approach to adjust the weight values decreasing the current loss score. Each algorithm has an optimizer that adjusts these parameters following the so called Back-propagation approach. The framework of a generic deep learning algorithm is reported in Figure 1.6. Before training

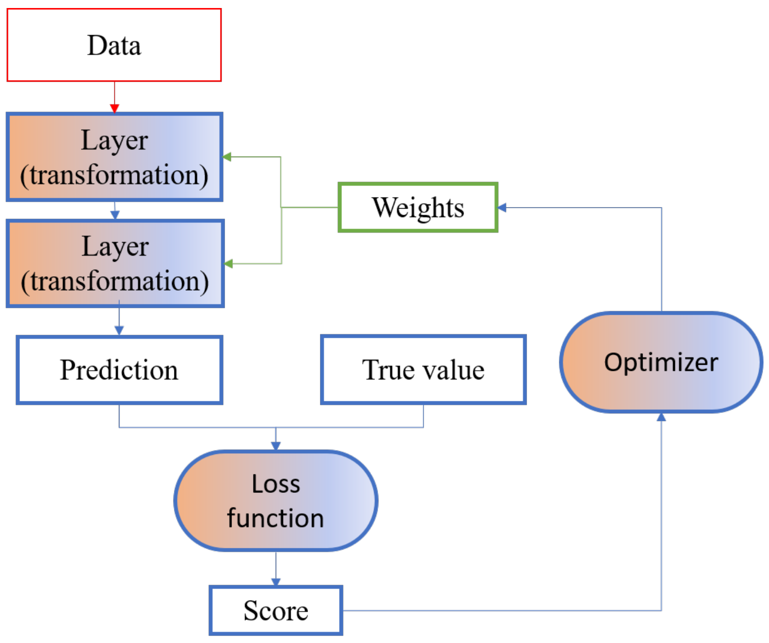


FIGURE 1.6: Backpropagation algorithm

phase, the weight values are randomly assigned and therefore the loss score related to each observation is high. During the training loop, which is repeated a proper number of times depending on the specific task, the weights are adjusted following the direction that minimize the loss function.

1.2 Why Today?

Only recently deep learning has been recognized as a crucial technology though AI has its origin in 1950s. The recent success of this AI approach is driven by two technical forces: the availability of appropriate data usable during the training phase and the increment of hardware power that makes increasingly accessible the computational performance needed to try new ideas. Today, a massive amounts of data from many scientific disciplines are available and gaining insight from them has became a new scientific inquiry as well as a commercial opportunity. Even in turbomachinery applications, over the last decade, a great amount of data has been collected due to new high-performance computing and advances in experimental measurement capabilities [4].

1.2.1 Dataset

In a paper published in 2001 [5], Microsoft researchers showed that very different Machine Learning algorithms, including fairly simple ones, performed almost identically well on a complex problem of natural language disambiguation once they were given enough data (see Figure 1.7). The amount of technical skills required to get good performance from a

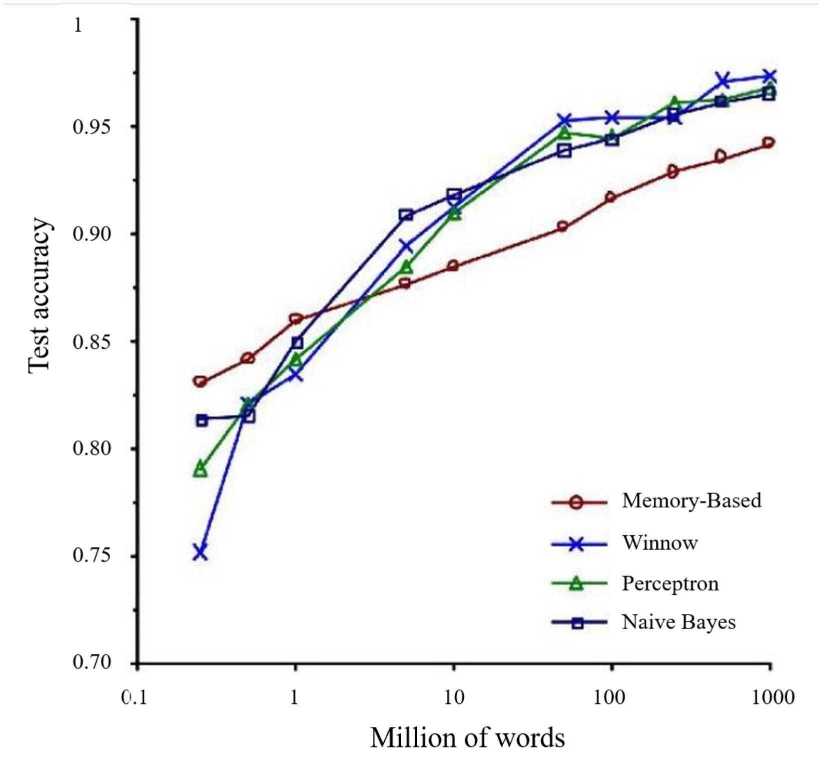


FIGURE 1.7: Relevance of data size. Readapted from [5]

machine learning algorithm reduces increasing the amount of training samples. Today, we are living the age of Big Data. The increasingly digitization of society drastically improves the quantity and accessibility of data from many industrial activities. The size of datasets that can be used for machine learning applications has notably increased during the last few years and the computers are more and more networked together becoming easy to centralize and storage these data. To better understand the big data functional value we can look at the DIKW Pyramid (Figure 1.8) [6]. It is a hierarchy representation reporting various ways of extracting insights and value from all sorts of data. DIKW illustrates well the features of knowledge, i.e. understanding the world filtering reality. Knowledge filter is in data reduction or modeling.

At the pyramid base we have the data. They are just a group of signals or symbols without organization. They are useless in this form. We start getting information from data when we are able to organize and classify this unstructured noise answering to simple questions about the nature of data (i.e., what,when). Next and probably most significant step in the pyramid is Knowledge. Knowledge requires learning. Whereas Information allows understanding relationships, Knowledge can detect patterns, allows the generation of predictive models producing real insights. The top of pyramid is Wisdom. The final step allows correctly predicting future not only relying on patterns but deeply comprehending the reason behind each pattern.

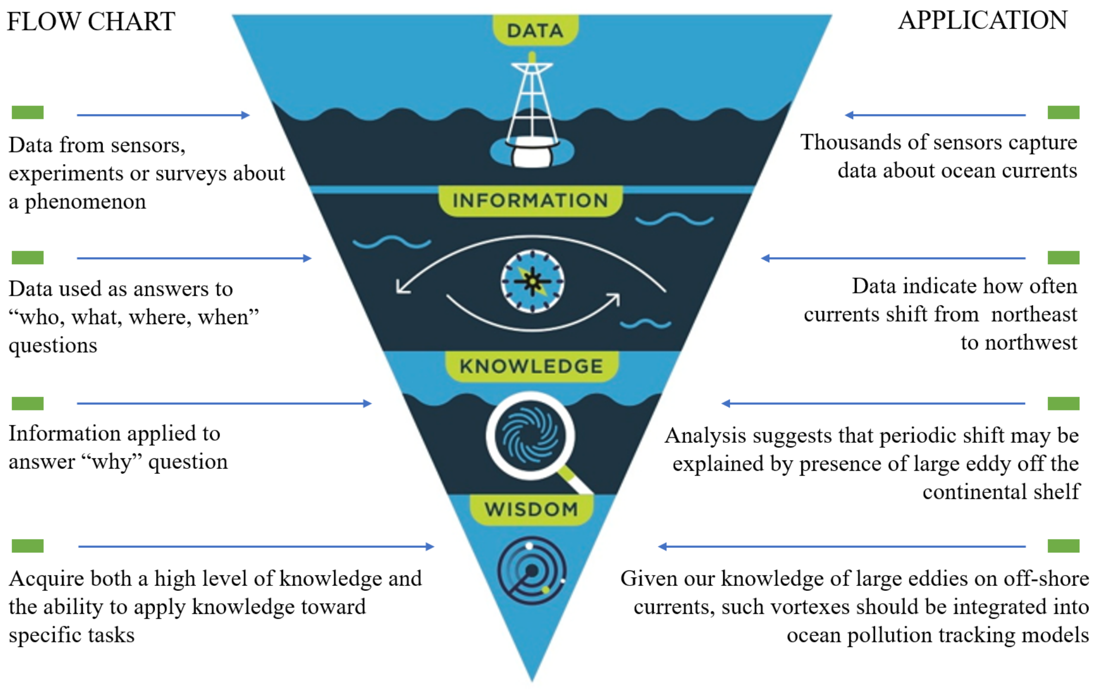

FLOW CHART
APPLICATION
DATA
INFORMATION
KNOWLEDGE
WISDOM
Data from sensors, experiments or surveys about a phenomenon
Thousands of sensors capture data about ocean currents
Data used as answers to "who, what, where, when" questions
Data indicate how often currents shift from northeast to northwest
Information applied to answer "why" question
Analysis suggests that periodic shift may be explained by presence of large eddy off the continental shelf
Acquire both a high level of knowledge and the ability to apply knowledge toward specific tasks
Given our knowledge of large eddies on off-shore currents, such vortexes should be integrated into ocean pollution tracking models

Figure 1.8: Big Data's functional value: the DIKW pyramid

1.2.2 Approximation model accuracy

Another important factor responsible for the actual success of machine learning is the availability of computational resources that allows run larger and more accurate models. Introducing the hidden units determined the size duplication of artificial neural network every 2.4 years. Early neural networks had only few neurons. It is not surprising they cannot get good accuracy in solving sophisticated artificial intelligence problems. Today, the availability of faster and powerful CPUs, the advent of GPUs, better software infrastructures and faster network connectivity allow the creation of larger model, quite demanding from a computational point of view, but able to solve complex real word tasks with great accuracy levels. Moreover, the chance of exploring more and more complex models having the possibility of training these models on huge dataset determined several important improvements of gradient propagation algorithm as:

- better activation functions for hidden layers
- new weight- initialization schemes
- more performing optimization algorithm as Adam optimizer or Limited Memory BFGS

Due to the aforementioned reasons, the scale and accuracy of deep learning algorithms has dramatically increased. Deep learning has been successfully applied to broader set of applications providing accurate performance in both prediction or classification purposes so as the complexity of the tasks they can solve[7].

1.2.3 A review of machine learning applications

Referring to [8], several applications of machine learning on turbomachinery and fluid mechanics are reported in the following. Fluid dynamics applications differ from other machine learning applications as speech recognition or image classification because it is necessary understand and quantify the physical mechanism before analyze them.

In 1990s many ML strategies were developed in flow-related problems as trajectory analysis, particle tracking [9], [10] or in the field of multi-phase flows [11].

Successively, great attention has been directed on the central problem concerning flow modeling: the computational effort needed to solve small scale Navier-Stokes equation or experiments for a specific application. Since simulations and experiment are often too demanding for real-time application [12], a great effort has been applied to obtain reduced-order model keeping a sufficiently high accuracy at a fraction of the cost [13]. Pattern recognition and data mining are core strengths of this task. Machine learning provided new ways for dimensionality reduction, which aims extracting key features and patterns used to describe the domain with a reduced set of coordinates [14] or identifying transformations that simplified the capture of essential flow physics [15].

Machine learning has been also successfully applied for clustering and classification tasks. The k-Means clustering algorithms has been used by [16] aiming to discretize high-dimensional phase space for the fluid mixing layer. Classification is also used in fluid dynamics to investigate fluid behavior. The supervised learning approach is used to sort data into a proper set of categories. Recently, [17] investigated the classification of wake topology from vorticity measurements. In [18], the k nearest neighbors algorithm has been used to isolate exotic wakes. A similar work has been realized by [19].

Concerning optimization and control problems, machine learning algorithms have been successfully applied to develop surrogate models that can substitute the original cost function (i.e., experiments or numerical simulations) reducing the computational effort needed to determine optimized parameters. Stochastic optimization has been widely used in engineering design. A short list of application includes aerodynamic shape optimization [20], shape and motion optimization [21] and improved power extraction in cross-flow turbines [22].

Neural network have been also applied to flow control. [23] was a pioneer of application of neural network in turbulence flow control. Using a local wall-normal blowing based on skin-friction sensors, a single layer neural network has been optimize to the end of decreasing the skin-friction drag of a turbulent boundary.

Despite the great amount of data from simulations, experiments or monitoring sensors the turbomachinery community is still distant from the vastness of training data available in other machine learning applications. In addition, many successful machine learning applications rely on not expensive evaluation functions and exhaustive categorization of the problem. Unfortunately, this is not the case of turbomachinery, where experiments are generally difficult to repeat or automate and numerical simulations may require great computational effort for extended period of time.

Chapter 2

Turbomachinery machine-learnt design

"We must develop a comprehensive and globally shared view of how technology is affecting our lives and reshaping our economic, social, cultural, and human environments. There has never been a time of greater promise, or greater peril."

Klaus Schwab

This chapter firstly presents the classical approach to turbomachinery design: the process includes all the operations, from the preliminary design, through single or multi-objective optimization, to the final blade shape and, finally, the experimental validation. Successively, the contribute of machine learning in each of these phases will be analyzed. Finally, the approach one should use to develop and correctly exploit a generic machine learning algorithm is presented.

2.1 Classical turbomachinery design approach

Turbomachinery community is witnessing a great push in rethinking all manufacture activities. This is mainly caused by what scholars are identifying using the term Industry4.0, referring to the trend of optimization or customization of products and data exchange in manufacturing technologies. In this contest, the use of smart technologies is becoming predominant, in particular the application of meta-modeling techniques to reduce the computational cost of CFD simulations.

On the other hand, the change in regulation that established minimum efficiency grade for Energy related Products determined the need to redesign many products currently available on the market.

2.1.1 Flow chart of classical design approach

Industrial turbomachinery designers generally rely on empirical methods developed during the years for aerospace design process. These methods reflect the cumulative experience of engineers that have designed turbomachines for specific application so this process can be seen as a form of local optimum. Figure 2.1 reports a classical design approach, which can be outlined by the following four steps: (i) definition of Design Space, (ii) the Stage Design, (iii) the numerical verification, and finally, (iv) the experimental test.

The *Design Space* is defined according to the desired duty point, in terms of volume flow rate and specific work (Q,gH), together with geometric and kinematic constraints imposed by the application. In so doing, the flow coefficient (Φ_d),defined as the ratio of

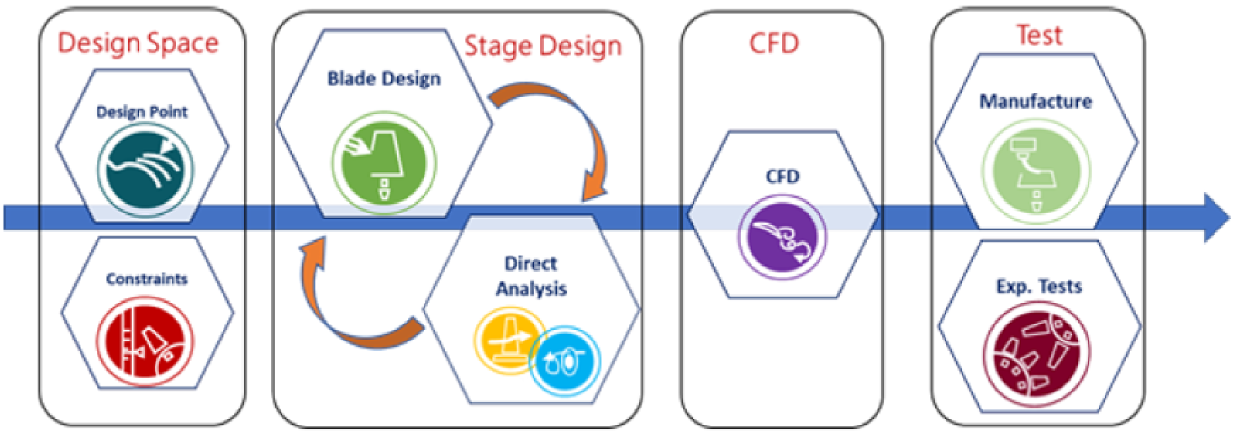


FIGURE 2.1: Flow chart of classical design approach

volume flow rate to the blade speed, and work coefficient (Ψ_d), defined as the ratio of specific work to the square of blade speed, are defined [24]. Successively, to identify what type of machine will be the best choice for the given duty, the specific speed (N_s) and specific diameter (D_s) are used [24]. The nondimensional parameters, derived from flow and work coefficient, are described in Table 2.1.

Design objectives	**Formula**
Shroud diameter	Dt
Angular velocity	ω
Global flow coeff.	$\Phi_d = Q/(\omega D_t^3)$
Global work coeff.	$\Psi_d = gH/(\omega D_t)^2$
Specific speed	$N_s = \Phi_d^{1/2}/\Psi_d^{3/4}$
Specific diameter	$D_s = \Psi_d^{1/4}/\Psi_d^{1/2}$

TABLE 2.1: Global design parameters

Taken together, these global parameters facilitate the definition of turbomachine design space and identify the design point on a Balje chart (see Figure 2.2 and Figure 2.3).

The flow evolving through the blade to blade passage may vary considerably from hub to shroud, thus expecting global dimensionless group are able to deal with complex three dimensional flow may seem optimistic. However, following the meridional plane theory, the flow can be modeled in several meridional stream lines and the aforementioned flow coefficient and work coefficient can be rewritten as: $\Phi_d(r) = c_m(r)/(U(r)$ and $\Psi_d(r) = gH/(U(r)^2$. Being $c_m(r)$ the local meridional velocity and $U(r) = \omega r$ the local blade speed.

The *Stage Design*, second step in the classic design process, involves the design of the blade and the direct analysis. In this phase, the designer iterates until the resulting geometry converges on the selected duty point. The blade design is an inverse problem design in which, starting from the target fluid deflection, the geometry of blade surface, needed to produce the desired performance, is determined. The performance evaluation is instead a direct analysis of the geometry generated by the first step. In particular, the fluid flow evolving through the designed rotor is predicted assessing the performance of the blade. The stage design loop is iterated until new geometry reaches the desired operative performance. The entire process can be seen as an optimization process that explores a small portion of the design space identified after the selection of target duty point.

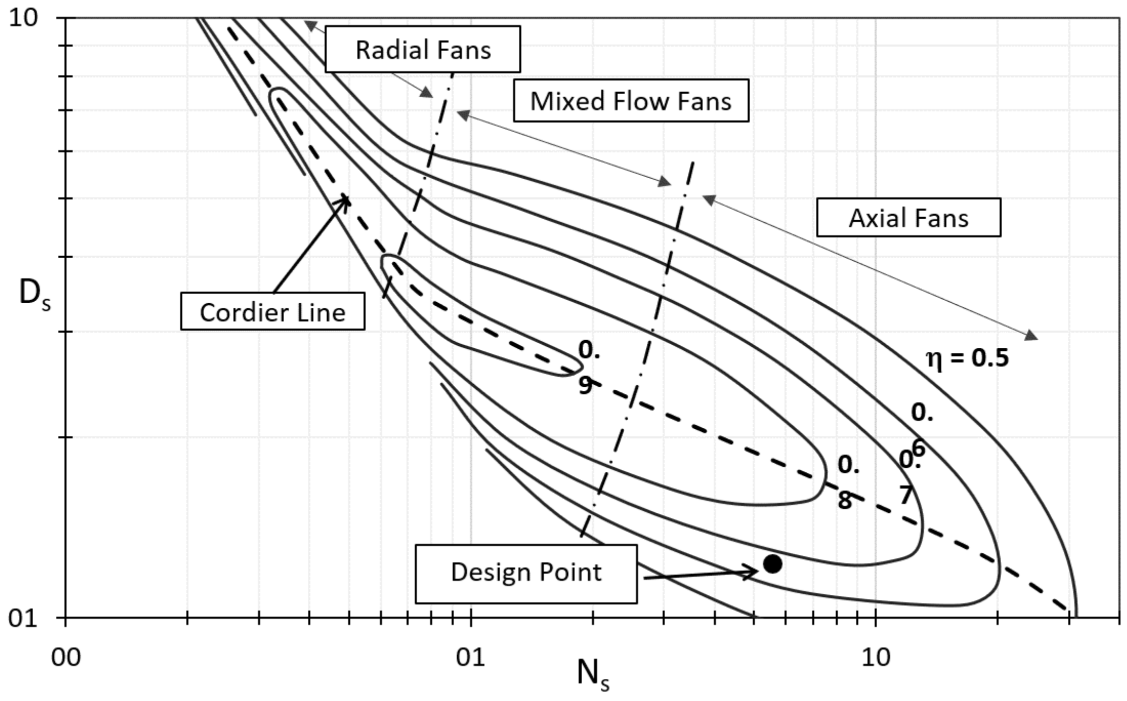


Figure 2.2: Preliminary design location on a Balje chart for compressors

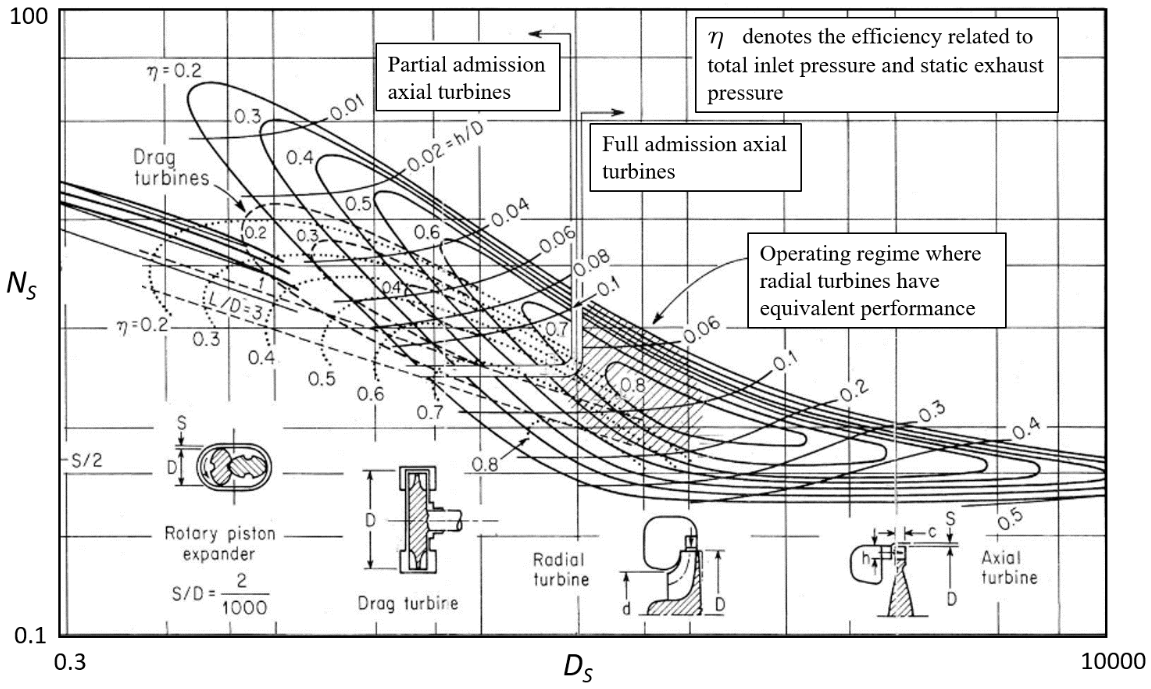


Figure 2.3: Balje chart for turbines

2.1.2 Blade geometry and design parameters

The design of axial flow compressors and turbines has been generally based on measurements of flow evolving through two-dimensional blade cascades. The cascade blade can be seen as a cambered line upon which the profile thickness distribution is superimposed. Historically, great part of the algorithms used to solve both the direct (performance analysis) and inverse (blade design) problems rely on 2D cascade theory. The cascade blade is characterized by the following geometric parameters:

- stagger angle, γ ; angle between chord and rotation axis direction
- solidity, σ; chord-space ratio
- blade inlet angle, β_1'; angle of the tangent to the camber line at the leading edge
- blade outlet angle, β_2'; angle of the tangent to the camber line at the trailing edge
- camber angle, $\theta = \beta_2' - \beta_1'$

The change in angle of the flow is called deflection, $\epsilon = \beta_1 - \beta_2$, and in general this parameter will differ from the camber angle due to flow incidence at the leading edge and deviation at the trailing edge. Incidence is the difference between the inlet flow angle and the blade inlet angle, $i = \beta_1 - \beta_1'$. The deviation is the difference between the exit flow angle and the blade exit angle.

The Angle of Attack (AoA) is defined as follows:

$$AoA = \beta_m - \gamma \tag{2.1}$$

where the mean flow vector angle (β_m) may be expressed in terms of the inlet and outlet flow angles, $tan(\beta_m) = (tan(\beta_1) + tan(\beta_2))/2$.

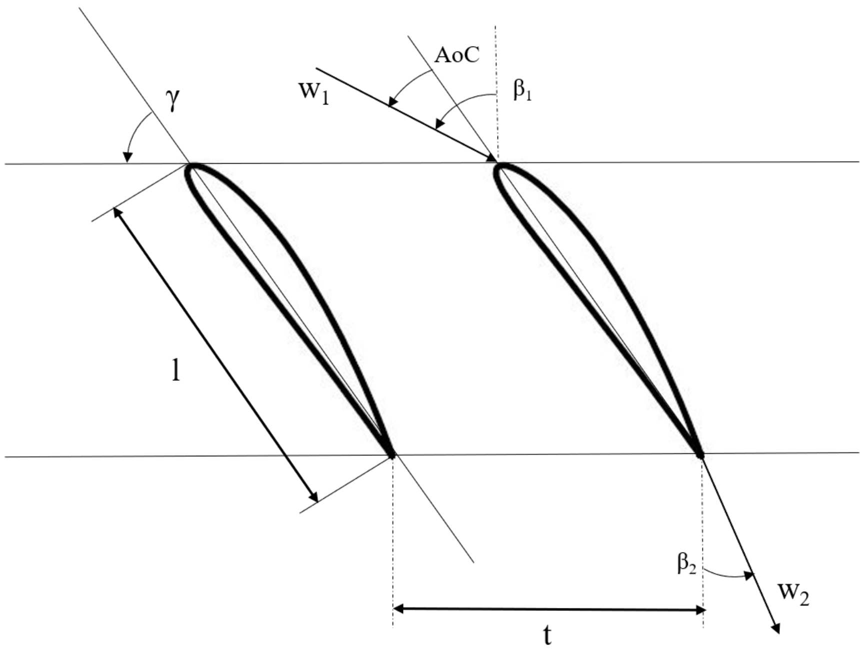


FIGURE 2.4: Compressor cascade geometry

2.1.3 Axisymmetric flow solver

After defining the geometry of the blade, the performance of the new rotor needs to be verified using an analysis tool. At first instance, we may use a virtual test rig that allows reducing the computational cost preserving a performance prediction comparable with CFD. Here, we report a brief description of AxLab, a Python software developed in collaboration with the turbomachinery group in Sapienza University of Rome for axial flow rotor performance prediction. The software is based on the quasi-3D axisymmetric principle: the complexity of the 3D flow is partially reproduces by the juxtaposition of flow conditions on meridional and circumferential plane.

AxLab requires the computation of a simplified radial equilibrium equation at only one axial station, while the flow behavior inside the blade vane is predicted by means of different aerodynamic models.

AxLab calculation process can be summarized in four main steps as shown in Figure 2.5: (i) first, the geometric specification of the blade is set and the algorithm computes the inlet velocity diagrams, then (ii) β_2 angle is computed for each section of the blade performing calculation on circumferential plane and these angle values will be used for (iii) calculation on meridional plane. Steps (ii) and (iii) rely on the coupling between cascade theory and radial equilibrium equation. Flow exit angle is evaluated from polar curves of section profile. The lift and drag coefficient of each section are numerically computed through XFoil [25] an open source software for isolated airfoil design and analysis. The estimation of β_2 requires running an iterative loop using the deviation model implemented in the code, namely *LanD* and *Gamma* model. To determine the cascade deviation, both methods perform a bidimensional flow simulation of a single airfoil of the cascade to calculate aerodynamic parameters, in particular C_l and C_d coefficients in function of blade angle of attack. To maintain small computational cost, 2D simulations are performed using vortex panel method (XFoil). Points (ii) and (iii) are repeated until convergence on outlet angle β_2 is reached according to a user defined tolerance. At the end of the iterative loop (iv) the output variables are computed and fan performance is finally provided. Steps (i) and (iv) are well described in [26], while the iterative procedure and LanD and Gamma deviation models are reported in [27]. Note that the core of this analysis code is the evaluation of the outlet angle of a given cascade geometry. Estimation of β_2, which rely on the computation of *Cl* and *Cd* coefficients, results in:

$$\beta_2 = \beta_2(\sigma, C_l, C_d, \beta_m, \beta_1) \tag{2.2}$$

It is worth noting that additional deviation models can be easily implemented in the code. Its flexibility makes AxLab a very useful tool for numerical design optimization. After the stage design by means of AxLab, the expected performance need to be validated through three-dimensional numeric simulations or experimental tests.

2.2 Role of machine learning

Artificial intelligence determined a great push in rethinking the classical approach to design and optimization above-described. In particular, machine learning played a key role in each of the phases depicted in the flow chart of Figure 2.1 boosting new design and optimization strategies and improving designer understanding of blade geometry impact on fluid flow behavior.

Machine learning techniques became a great support for decision making processes in turbomachinery R&D:

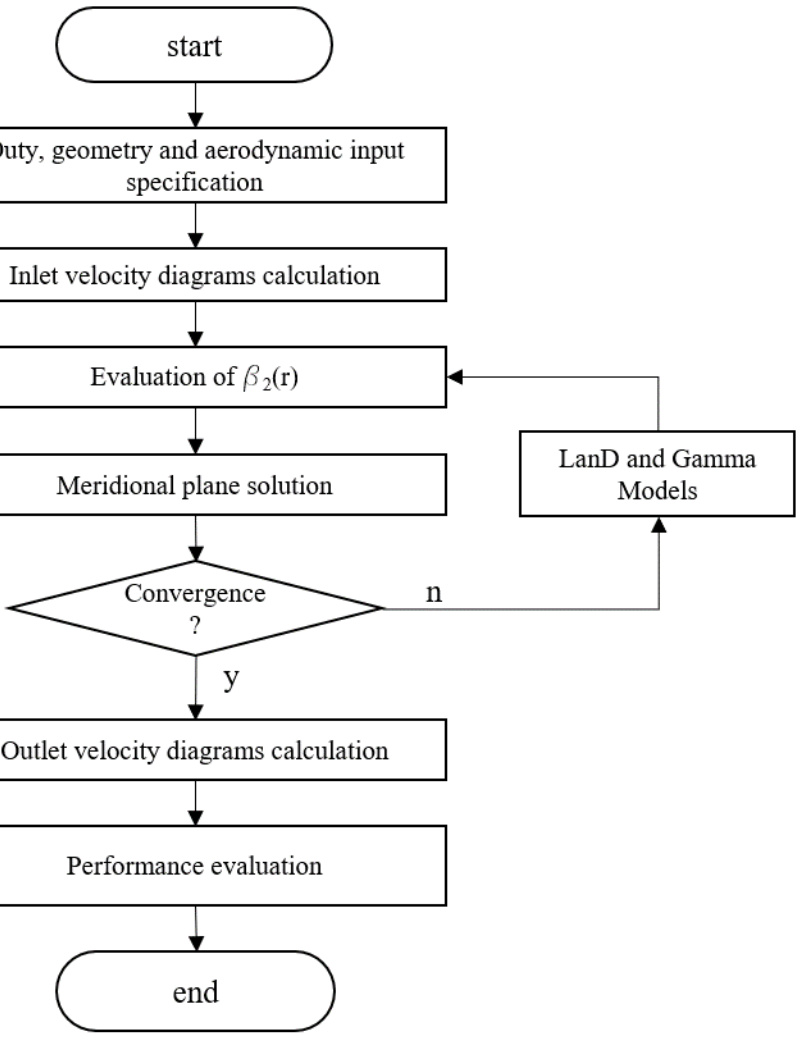

start
Duty, geometry and aerodynamic input specification
Inlet velocity diagrams calculation
Evaluation of β_2(r)
Meridional plane solution
LanD and Gamma Models
Convergence ?
n
y
Outlet velocity diagrams calculation
Performance evaluation
end

FIGURE 2.5: AxLab analysis sequence chart

- *exploratory data analysis* - improves the exploration of fluid machinery design space identifying new correlations between geometric parameters and rotor performance that may be used during the preliminary design process avoiding the use of old empiric rules. Moreover, during a part or the entire design space exploration regression models may be used

- *regression modeling* for the approximation of time consuming procedure (i.e., CFD analysis) - may provide an easy-to-run analytical model that ideally behaves as the original model at a reasonable cost. Furthermore, regression modeling can also be used as *cost functions* in several multi objective optimization problems (MOOP) reducing the computational cost required by standard optimization approaches; and

- *unsupervised learning methods* - may enhance the designer understanding of specific problem enabling a better problem formulation including different constraints, design variables, new parametrizations or eliminating unnecessary information.

The introduction of machine learning radically improved the quantity of support tools for a turbomachine designer allowing testing new ideas using simplified models. To this end, Figure 2.6 illustrates at a glance the aspects of the design chain that beneficed the most from machine learning algorithms. In particular, machine learning strategies have been applied to all the activities from design space exploration, through the optimization of the blade geometry and to the final performance analysis by means of three-dimensional CFD .

These machine learning applications and studies represent the core of research experience documented in this thesis and will be described in the following chapter 5, 6 and 7.

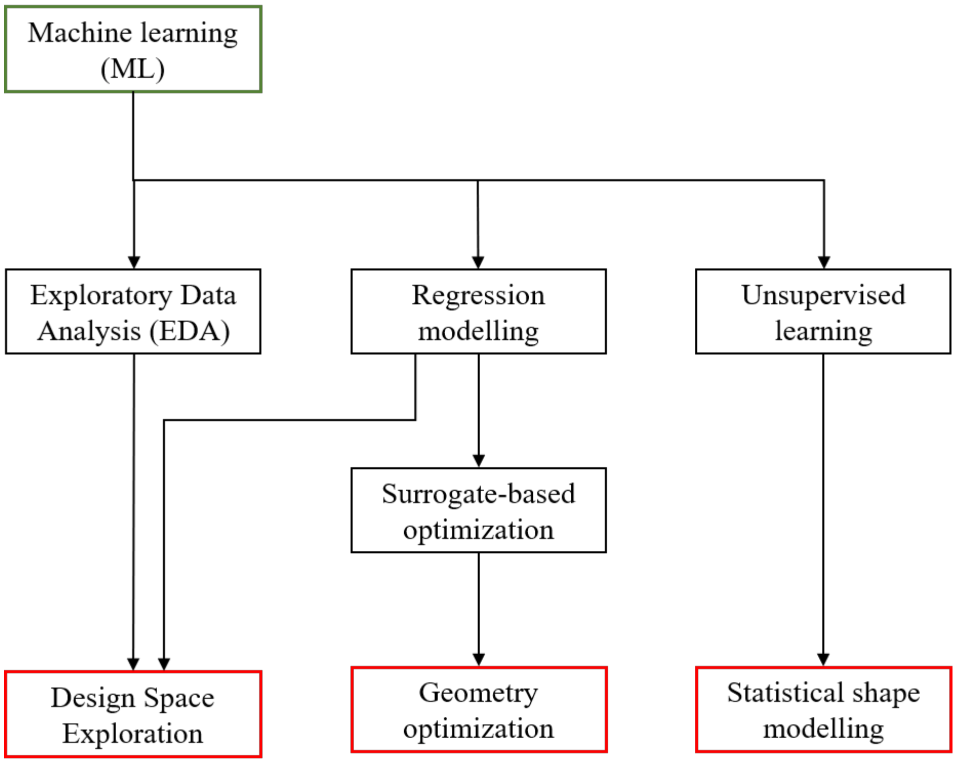


FIGURE 2.6: Machine learning applications inside design phase

2.2.1 Design Space

The first step in any classical design flow chart (see Figure 2.1) concerns the definition of design space. This preliminary phase is critical to define the type (i.,e., axial, radial or mixed) and size (i.,e., tip diameter) of the fluid machinery under development. The fundamentals of the methods currently used for the definition of industrial turbomachines *Design Space* have their origin from the researches done since the 1950s. Cordier [24] used the best efficiency operation data of several high quality designs of hydraulic pumps and fans linking the dimensionless size and rotational speed to the maximum achievable efficiency of well-designed machines. In particular, Cordier diagram is an empirical diagram based on measurements which delivered a relation between flow-rate, pressure, rotating speed, and diameter. However, the diagram does not provide any information on the blade shape (i.e., blade angles, chord distribution etc.).

Smith [28] in 1965 developed a more universal approach to axial turbines performance analysis based upon the use of local dimensionless variables. Elaborating the experimental data of 70 Rolls-Royce aircraft gas turbines, he developed a simple method of correlation which relates the efficiency to flow coefficient and stage loading coefficient. All turbine stages were tested with constant axial velocity, while efficiency values were corrected to eliminate tip leakage loss.

The process of design standardization has been completed by Balje [29] using the mean line models he developed to simulate the design operation of many turbomachines. Several design charts have been developed reporting the isoentropic efficiency as function of specific speed (Ns) and diameter (Ds). These are useful for scoping the turbomachinery type (i.e., axial, radial or mixed) for a give application and to determine the expected efficiency. An interesting feature of these charts is that Balje corrected the curves for the different viscous effects which can occur in machines of different size and speed, by comparing the data on efficiency at the same effective Reynolds number.

The issue associated with the use of these charts regards the dimensionless of the considered parameters. No relations between these parameters and the geometry of the blade in terms of chord distribution, twist disposition and camber angle or the blade hub to tip ratio and solidity are provided. The above cited correlation are not sufficient to univocally define the geometric parameters of a turbomachine design especially when blade loading different from the free-vortex distribution are considered. Currently standard approaches relate to traditional methods, which are based on experimental data on 2D stationary cascade and use empirical relations to close the blade design problem. Several empirical rules are available in literature. These rules have been developed during the years for compressor and turbine blade design. The McKenzie empirical relation relates the stagger γ with the mean flow angle β_m, determining a theoretical stagger angle for maximum efficiency [30]. Carter [31] investigating the effect of profile shape on the performance of blade cascade proposed a relation between lift coefficient and solidity. In particular,Wallis determined an optimal lift coefficient [32] that can be introduced in the Carter's relationship to obtain the solidity value. Another empiric method to derive the value of solidity is introducing the Howell equation [33] into McKenzie empirical rule. This method is only applicable to circular camber line blades. To set the solidity we can also follow the Lieblein method [34], which related the solidity to the span-wise distribution of the diffusion factor. He showed that the loss in a blade row increases rapidly as the flow starts to separate, which occurs when the diffusion factor exceeds about 0.6 [34]. A low value of diffusion factor determines lower pressure gradients across blade passage needed to turn the flow. Using the analytical definition of the diffusion factor, Lieblein method allows for the calculation of the optimal cascade solidity as a function of the relative flow angles and the stagger. A deeper explanation on the use of such empiric methods to determine the blade geometry starting from the definition of the desired performance is reported in [27].

Besides the methods available in open literature, practice of manufacturers is to replicate good designs developed in the past and re-adapt them to similar tasks in order to meet customer's requests, with minimum development time and cost.

All cited empirical rules and procedures, which condensate the know-how of decades or the manufacturer intellectual proprieties, result fragmented and unrelated. Designers still have the need of smart and more robust procedures to select the rotor geometric parameters in order to stay competitive in the actual market landscape.

It is in this framework that an artificial intelligence strategy has been developed aiming to enrich the amount of tools a designer may use in the preliminary design phase. In particular, several machine learning algorithms have been exploited to explore the axial flow turbomachinery design-space and identify correlations between rotor performance and enriched geometric and kinematic rotor parameters. The aim is to derive a set of multidimensional Balje charts in which the value of some dimensionless design parameters defining the blade shape is added to the best efficiency datum for a given pair of specific speed and specific diameter.

The exploration of design space required the enhancement of Axlab by the introduction of a new deflection model artificial intelligence based. It allows replacing the isolated airfoil assumption made by LanD and Gamma model, used in the original version of AxLab for fluid deflection prediction, with a surrogate model capable of predict the effect due to a blade to blade interaction. Moreover, to reduce the dimensionality of performance data of tested axial flow turbomachine population, unsupervised learning methods have been performed on the hyper surface of solutions highlighting a set of linearly uncorrelated geometric features responsible for the major amount of the dataset variability. A subsequent analysis enables to connect these geometric variables with the specific speed and diameter aiming to enrich the Balje chart with new relations between the well known dimensionless parameters and the more representative blade geometry variables (i.e., blade solidity, twist disposition, blade count, hub-to-tip ratio).

All the procedure needed to develop the machine learning strategy and the results obtained is reported in detail in chapter 5 of present thesis.

2.2.2 Design and Optimization

Regardless of specific application field, optimization is the selection of a best element, with regard to one or more criteria, from a set of available alternatives. To the end of selection, we need measuring the performance of all available samples. When looking at optimization strategies, several multi-objective optimization problems (MOOP) have been presented in the open literature and the choice depends upon various aspects, in particular the level of knowledge of the objective function and the fitness landscape [35]. Among these methods, Evolutionary Algorithms (EAs) often perform well approximating solutions to all types of problems. They ideally do not make any assumption about the underlying fitness landscape and they are able to find a good spread of solutions in the obtained set of solutions. Furthermore, genetic methods are independent from level of knowledge of the objective function and the fitness landscape, resulting less sensitive to numerical noise. Over the past decade, a number of Multi-Objective Evolutionary Algorithms (MOEAs) have been proposed [36], [37], primarily because of their ability to find multiple Pareto-optimal solutions in one single simulation run. In the number of different EAs the most popular belong to the family of Genetic Algorithms (GAs) and among these, we focused on NSGA-II [38]. When dealing with real-world optimization problems, the number of calls of the objective function to find a good solution can be high. Furthermore, in optimization problems based on computer experiments, where the simulation acts as the objective function, the computational cost of each run can severely restrict the number of available calls to the fitness function. Shape optimization is one of the most important applications of CFD. However, a single CFD simulation may require days to be accomplished.

Therefore, to reduce these computational efforts, the introduction of surrogate models (SMs) have been proposed. Traditionally, the use of SMs in MOEAs have been classified according to the type of surrogate model at hand [39], [40], [41]. However, these works have shelved MOEA's point of view concerning the integration of SM into the evolutionary process. Jin [42] and Manríquez [43] proposed a classification based on the way EAs or MOEAs incorporate the SMs (focusing on the optimization loop). This taxonomy is called working style classification and allows finding similar optimization problems, putting greater emphasis on the methodology followed and not on the SM used. According to such a classification, the approaches are divided into Direct Fitness Replacement (DFR) methods and Indirect Fitness Replacement (IFR) methods [43]. The DFR method is also called simple-level or No Evolution Control (NEC) and it is considered the most straightforward SM approach. The simple-level framework is illustrated in Figure 2.7 where MOEAs calculate the solutions using the SMs without any assessment against the original fitness function. NEC have proven to be adequate in problems with low dimensionality in both decision and objective space, and they established in turbomachinery applications [44]–[47]. However, in more challenging problems [43], [45] this approach can converge to false or sub-optima, which in a MOOP is a Pareto front not corresponding to the Pareto front estimated by the original function.

The IFR method, also known as bi-level framework (or in-fill criterion), advocates the use of fitness function during the EA process, adding new sample points to the current data set, while one or more components of the MOEA (typically the variation operators) are assessed in the SM. The IFR method is clearly a two-stage approach, as illustrated in Figure 2.8. A number of solutions are produced, evaluated and compared using the SM, among them, n best solutions are then delivered to the parent approach and evaluated against the original fitness function to enrich, iteration by iteration, the sampling data set from which meta-models are derived. Therefore, it is expected to keep the optimization towards the true Pareto front and at the same time to reduce the risk of convergence to false optima [48]. Most of the existing works in this category use the MOEA in a direct coarse grain search, while the SM intervenes in a local search, providing candidate solutions, which are then assessed in the real function. Therefore, the IFR approach uses the SM for exploitation purpose, while the MOEA is employed for design space exploration.

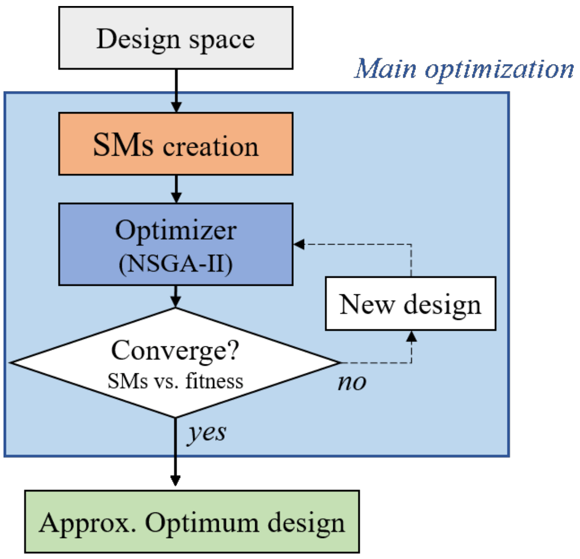


FIGURE 2.7: DFR or simple-level optimization framework

Benefits of surrogate-based optimization are reported from [49]:(i) optimization can be performed using different codes that the model will replace without additional complications;(ii) the process is simple to be parallelized dramatically decreasing the computational time;(iii) metamodels act as filter for numerical or experimental noise;(iv) metamodel is not a theoretical/physical model but allows user exploring a huge amount of design space.

In chapter 6 of this thesis, we will see how exploiting surrogate-based techniques in complex optimization problems pertinent to the aerodynamic of aerofoils.

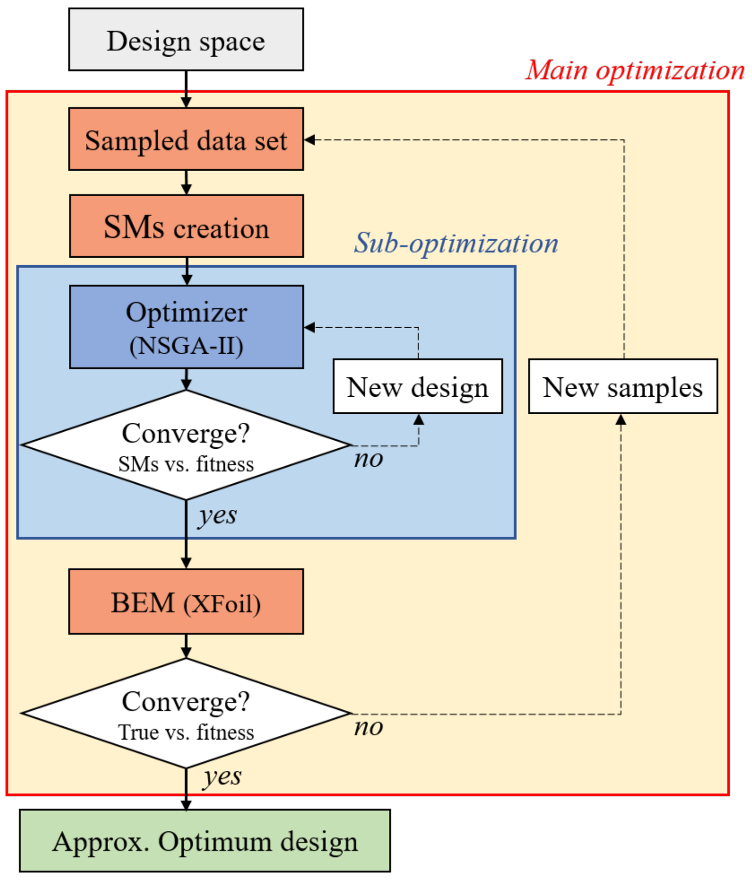


FIGURE 2.8: IFR or bi-level optimization framework

2.2.3 Statistical Shape Modelling

Shape optimization is one of the most important applications of computational fluid dynamics (CFD). For example, focusing on machine durability and performance, the overtip leakage flow remains a crucial challenge to any turbine designer, determining time between replacement of turbine components. The acceleration of hot incoming fluid into the overtip is followed by separation and re-attachment regions, making tip leakage flow responsible for more than 50% of casing heat transfer [50]. Moreover, the flow evolving through the tip gap is not properly turned, thus reduces aerodynamic loading and contributes to secondary flow losses generation in downstream stages [51]. During the last decades a multitude of numerical investigations have been carried out in a view to optimize turbine tip geometry using CFD-aided evolutionary approaches, trading-off between stage efficiency and heat transfer [52], [53]. Recent contributions have proposed and assessed the use of surrogate models in multi-objective optimization to reduce the massive use of simulation processes [37], [54]. Although tip geometry optimization is established as a common practice, the available strategies lack of generality. Those strategies typically refer to problem-specific solutions, with use of simplified objective functions, and the results of the optimization are largely affected by the initial choice of the shape representation. Noticeably, a proper geometrical representation of complex tip geometries is difficult because it should results from a detailed knowledge of the flow physics in the tip region, rarely available "a priori". Any optimization strategy, based on topology parametrization techniques [55], requires a high level of granularity of design space, difficult to handle and possibly resulting in infeasible shapes and often the optimal design resulted too difficult to find. To this end, data science can contribute to overcome the aforementioned limits relying on a combination of data mining and machine learning techniques to analyse and unveil casual geometry-performance correlations in a genuinely multivariate test-bed. The machine learning strategy, reported in the following chapter 7 of this thesis, addresses the use of statistical shape analysis techniques, well established in medical imaging community [56], [57], as a tool to design proper shape parametrization. In particular, unsupervised machine learning algorithms, as dimensionality reduction and clustering (see chapter 3 for a detailed description of machine learning algorithms), are used to explore a dataset of numerical simulations of HPT blade tips to the end of extracting new information improving the user understanding of flow behavior and develop a new statistical parametrization of tip surface. The test-bed is made of a set of optimized tip designs from [58]. The work also aims clarifying the correlation between tip surface zones, turbine stage performance and flow structures evolving behind rotor trailing edge.

Chapter 3

Machine learning algorithms

The following chapter aims to describe the machine learning and deep learning algorithms deeply exploited in the turbomachinery applications described in chapter 5, 6 and 7.

3.1 Introduction

Machine learning algorithms can be divided into two different fields: *supervised learning* and *unsupervised learning*. Supervised learning is probably the most common approach to machine learning. It consists of learning to map input data to known targets having a set of examples to train the model. Classification and regression problems belong to this category.

On the other hand, unsupervised learning determines interesting data transformation, without requiring the use of any target, to the end of better visualize data, to highlight the hidden structures inside data or to find correlations between the features used to describe each sample. Unsupervised learning algorithms are the mile stone behind data analysis and sometimes they represent a required step before performing supervised learning problem. Dimensionality reduction and clustering are the main branch of unsupervised learning. Figure 3.1 shows schematically the mentioned machine learning branches.

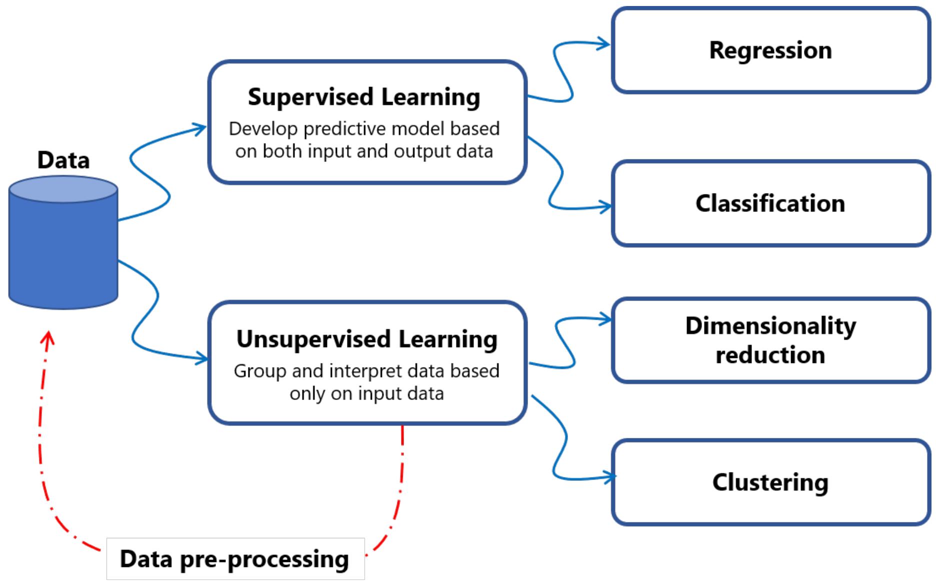


FIGURE 3.1: Machine learning branches

3.2 Classification

Classification problem is a branch of supervised learning that aims to predict the categorical label (i.e., discrete value) of new observations based on the learning acquired by trained model. Suppose you have a domain D defined as (see Equation 3.1)

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\} \tag{3.1}$$

where $\mathbf{x} \in R^n$ and $y = (1, 2, .., k)$ with $k \geq 2$ number of categories. Consider the training dataset is a subset $S \in D$. This dataset of labelled examples will be used to generate a machine learning model in the form $f : x \rightarrow y$ successively used to make predictions over unseen data $(x_i, y_i) \notin S$ in the form

$$y_i = f(x_i) \tag{3.2}$$

where y_i is output variable (or label), x_i is the independent feature in input and f the prediction model (or classifier). In the prediction phase, data described by the feature set are mapped by the function, learned during the training phase, into the proper category, which they belong to.

If U is the set of unseen data i.e., $U = D - S$, in order to evaluate the quality of a classification task, we need to measure the distance between the real value of observation y_i and the predicted value $f(\mathbf{x}_i)$ using:

$$E = \frac{1}{|U|} \sum_{(x_i, y_i) \subseteq U} f(\mathbf{x}_i) \neq y_i \tag{3.3}$$

It is worth noting that the size of S is a measure of model experience, while the variable x is referred to input features and the variable y are output variables (or labels) of classification problem.

Classifying data is a common task in machine learning and there are many machine learning algorithms that can accomplish this task. Classification methods range from simpler linear method, in which the classification is based on the value of a linear combination of features, to artificial neural networks.

3.3 Regression

Second branch of supervised learning is regression. We have the same domain seen in Equation 3.1 with the difference that now $y \in R$ and the algorithm task is to develop a model able to generate a real value prediction. Regression algorithms aim determining the relationship between a set of independent features and one or more dependent output variables. They try estimating the function f that maps $\mathbf{x} \rightarrow y$. Since all statistical models need incorporating a random error term, the goal is approximating the real function that generated the observations: $y = f(x) + \epsilon$. The estimate function is $\hat{f}$ and the prediction is $\hat{y} = \hat{f}(x)$. Figure 3.2 and Figure 3.3 help visualizing the differences between the different classification and regression objectives.

The main difference is that the output variables are continuous in regression while they are categorical in classification. In a regression problem the system attempts to predict the continuous value of an input based on past observations. In Figure 3.2, the line represents the true value of samples while the cloud points are the values predicted by the trained model. The vertical distance of each cloud point from the corresponding line point is a measure of prediction error for the i-th observation, $y_i - f(x_i)$. In classification, predictions are made by classifying samples into different categories. In the application reported in

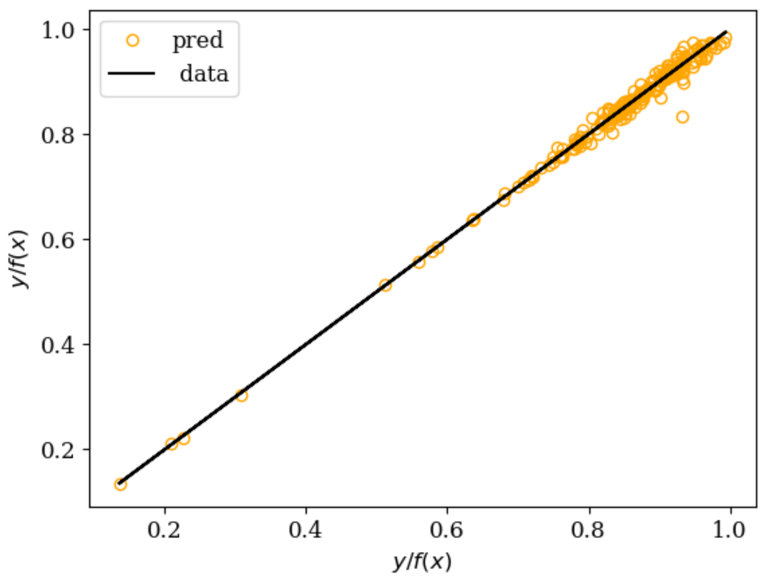


FIGURE 3.2: Regression problem

Figure 3.3, the classification task is dividing the observation set into two categories, then it is an example of binary classification in which the response of the model is $y_i = \pm 1$ identifying one or the other category respectively.

To estimate the best possible model, the user needs an index to measure the quality of regression analysis and in a regression problem such index needs to be minimized. Several performance indicators are available in literature, however they are generally variation of the well known root mean squared error over unseen data

$$E\Big[(y-\hat{y})^2\Big] = E\Big[f(x)+\epsilon-\hat{f}(x)\Big]^2 \tag{3.4}$$

The equation can be written as:

$$E\Big[(y-\hat{y})^2\Big] = (f(x)-\hat{f}(x))^2 + Var[\epsilon] \tag{3.5}$$

where the squared difference in the left hand side of Equation 3.5 is the reducible error that regression needs minimizing while the other term is the irreducible error, that is, the noise inside the original model. The reducible error can be further decomposed into a bias and variance term [59]:

$$E\Big[(y-\hat{y})^2\Big] = Bias\Big[\hat{f}(x)\Big]^2 - Var\Big[\hat{f}(x)\Big] + Var[\epsilon] \tag{3.6}$$

The Bias of the regression model is the expression of the error done approximating a real word problem by a simpler model $\hat{f}$. The variance of regression model quantifies the amount by which the response of the model would change if it had a different dataset from which it was modelled. The bias-variance trade off is the foundation concept of any machine learning algorithm.

Most of the work reported in this thesis concerns the design, optimization and application of several machine learning algorithms applied to the solution of regression problems. The definition and theory of these algorithms is reported in the following. Clearly, most of these algorithms can be used to solve both classification and regression problems.

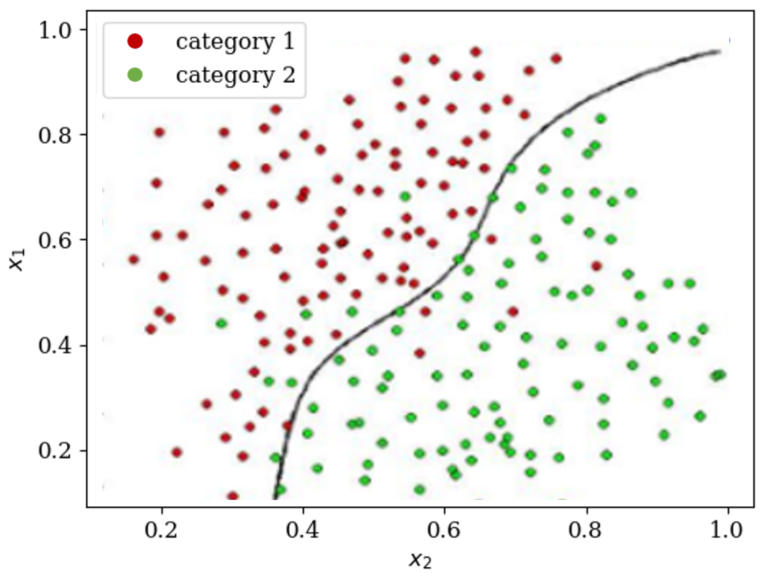


FIGURE 3.3: Classification problem

In this work, all the reported algorithms have been coded in Python language, making extensive use of scikit-learn library and Tensorflow, the open source machine learning library developed by Google Brain team [60].

3.3.1 Least Square Method

The method of Least squares (LSM) is a standard procedure in regression analysis to solve overdetermined systems. This method, which requires just some linear algebra, can be considered as a procedure to determine what is the line best fitting a given dataset. The method was developed by Gauss in 1795 and published several years later [61]. It consists of adjusting the coefficients of a model function (the function is also known as response surface) so that it best fits the data. This procedure can be easily generalized. Instead of finding the best fit line, we could find the best fit given by any finite linear combinations of specified functions.

The method is based on the theory that the real function that determines the observations can be represented by a polynomial approximation created by using a least squares fit approach [62].

The regression model is a function $f(\mathbf{x}, \beta)$, where

$$\beta = [\beta_1, ..., \beta_p]^T \tag{3.7}$$

$$\mathbf{x} = [x_1, ..., x_k]^T \tag{3.8}$$

with β vector of the p coefficients (or regressors) to be tuned, x vector of k input parameters. f is a vector function of p elements that consist of powers and crossproducts of power of x up to a certain degree $d \geq 1$. Generally, the most commonly employed LSM models include the linear or first degree model ($d = 1$),

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \epsilon \tag{3.9}$$

and the second-degree model ($d = 2$),

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i<j}\sum \beta_{ij} x_i x_j + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \epsilon \tag{3.10}$$

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

This method can be use to approximate the relationship between y and x that can be used to solve a regression problem and to determine the optimum settings of x that result in the maximum (or minimum) response over a certain region of interest.

The least squares regression method may become difficult to apply if large amount of data is involved, thus is prone to errors.

3.3.2 Ensemble Methods

Decision tree is one of the most popular supervised learning algorithm used for both classification and regression problem based on binary recursive partitioning [63]. The predictive value of such model is determined based on series of questions and conditions, in particular each node of the tree represents a feature (or regressor), each split represents a decision and each leaf is an outcome (see Figure 3.4). To quantitatively evaluate how good a split is, the algorithm needs evaluating an impurity metric $i(t)$ where t is a potential node that needs to be evaluated.

In a classification problem, in each node we need computing the probability of incorrectly classify our data. This probability is called Gini impurity (G) and is defined as:

$$i(t) = G_t = \sum_{i=1}^{C} p(i)(1 - p(i)) \tag{3.11}$$

where *p(i)* is the probability of incorrectly classify i-th sample and C is the number of classes. In a regression problem, the impurity index used is the mean square error (*mse*) defined as:

$$i(t) = mse_t = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 \tag{3.12}$$

with y_i real value, $f(x_i)$ predicted value and n number of samples.

The process of splitting the dataset into increasingly small subsets continues until no further gain can be made or a pre-set rule is met, e.g. the maximum depth of the tree is reached.

The key limitation deriving from the use of such simple model regards the tendency of predictive model to overfitting [64]. Decision trees are so flexible that small variations of training data determine high variation of learned parameters. Such a model is said to have high *variance*. On the other hand, a more robust model,less sensitive to noise, has high *bias*, meaning that it makes assumptions following pre-conceived ideas of data. In both cases, the resulting model is not able to generalize well to new data.

In order to improve the stability and accuracy, ensemble methods have been developed. An ensemble method use multiple learning algorithms, in this case trees, to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. We will see two ensemble algorithms based on different techniques: Random Forest and Gradient Boosting.

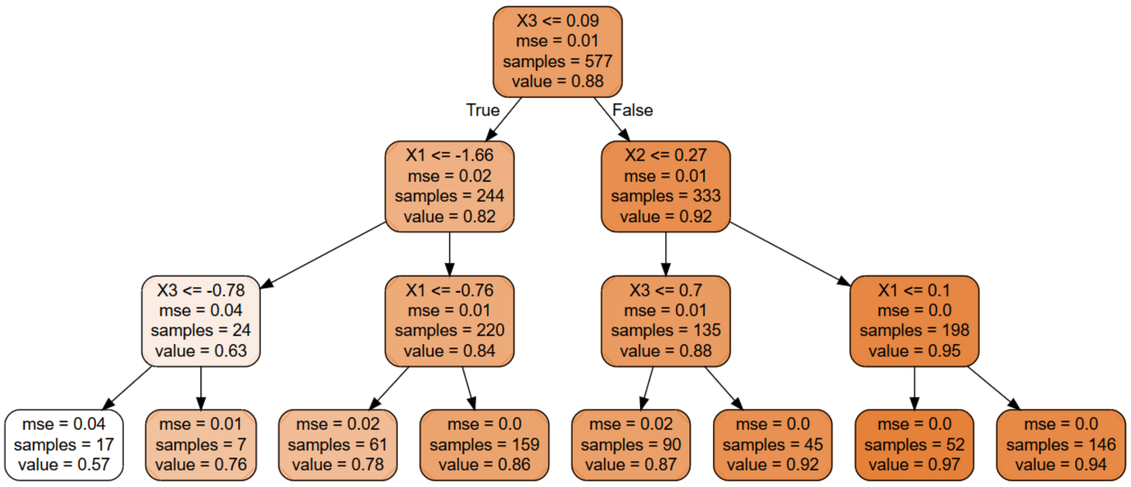


FIGURE 3.4: Example of regression tree with maximum depth=3

3.3.2.1 Random Forest

Random Forest (RF) is an ensemble machine learning method based on a statistical technique called bagging, short for *bootstrap aggregating*. Bagging algorithm builds multiple weak learners, decision trees in the case of random forest, and merges their prediction to get a more accurate response. Referring to Equation 3.1, the bagging algorithm creates M subsets of data. Each subset has $m \leq n$ samples randomly selected from the original training dataset, with replacement, and will be used to develop a tree ϕ_m where $m = 1, ..., M$. The bagging prediction $\hat{f}_b$ at a new sample x_i is then computed by averaging all M predictions:

$$\hat{f}_b(x_i) = \frac{1}{M}\sum_{m=1}^{M} \hat{f}_{\phi_m}(x_i) \tag{3.13}$$

In addition to bagging, RF algorithm introduces an extra random source during the splitting of a tree. It randomizes the partitioning procedure by only considering a random subset of features (independent variable) Φ_m. The RF prediction will be:

$$\hat{f}_{RF}(x_i, \Phi) = \frac{1}{M}\sum_{m=1}^{M} \hat{f}_{\phi_m}(x_i, \Phi_m) \tag{3.14}$$

The method flow chart is shown in Figure 3.5. It is worth noting that the RF merging strategy is not just averaging the prediction of simpler model. It is based on two concepts that are the theoretical base of the method:

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

In a random forest each tree learns from a subset of samples randomly selected. Training each tree on different samples decreases the global variance of the forest without increasing the bias because the deviations in the original training dataset do not impact the final response obtained from the aggregation [65]. Suppose each tree trained on its sub-dataset massively overfits data. All models have errors but the errors are random and not correlated. Averaging the responses will decrease the global error done by the forest.

RF is controlled by only three hyper-parameters. The first parameter is the minimum size of a node that is used to define terminal nodes (leaves) of each tree in the forest. Second parameter determines the total number M of trees grown (generally $M \in (500 - 1000)$, the

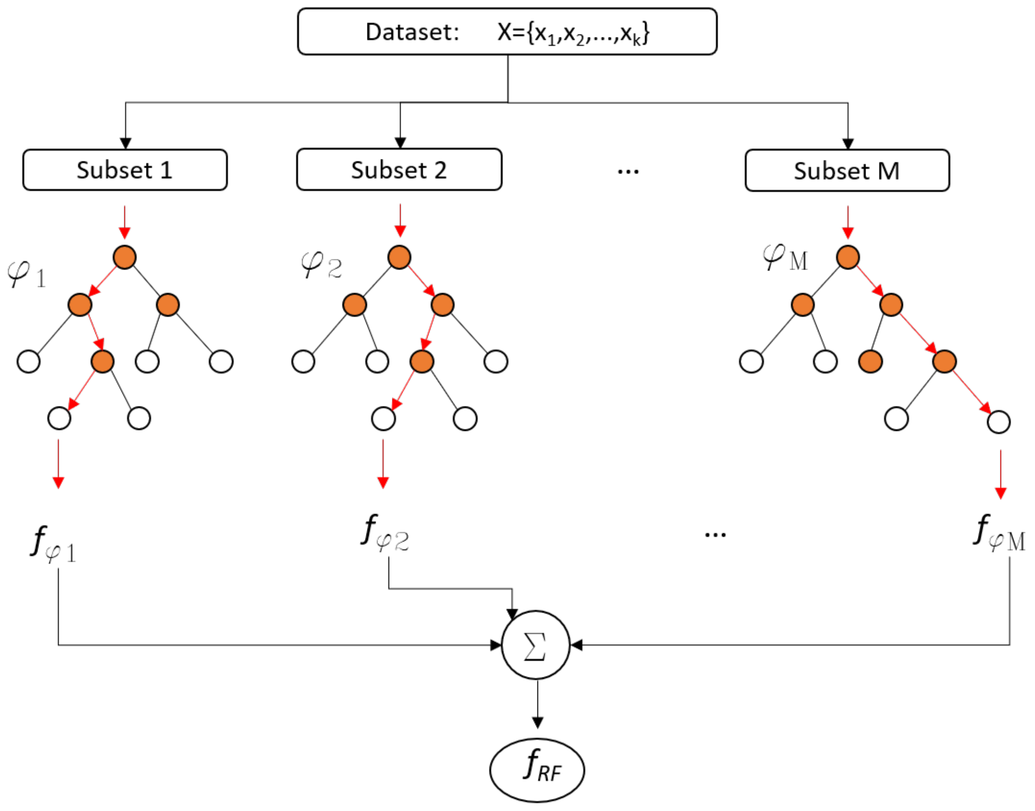


FIGURE 3.5: Random forest flow chart

accuracy asymptomatically decrees using more trees [66]). The third parameter defines the number of features randomly considered per split.

The main advantages of RF are: i), such model is easily implementable being governed by only three hyper-parameters; ii), it can reach high performance prediction for both classification and regression tasks. The model significantly reduces the risk of overfitting when compared to single decision tree and the user can easily measure the importance of each feature on the prediction. Moreover, RF doesn't require any statistical assumption (i.e., the data are normally distributed, or the relation between regressors and output is linear).

The disadvantages of such method are: it can become computationally expensive when applied to real word problem that require a high number of simpler learners; the model is not able to extrapolate data that are outside the training range; the methods based on bagging strategy are difficult to interpret from human being.

3.3.2.2 Gradient Boosting

The second ensemble method considered is Gradient Boosting (GB) [59], [67]. Being an ensemble method, GB predicts by combining the responses of weak learners however it differs from RF in the way the trees are built. According to boosting algorithm, trees are built sequentially and each new tree is developed to the end of correcting the prediction errors made by a previously trained learner.

Boosting focuses on samples that result more difficult to predict. Each new tree is trained using all the samples in the original dataset (no partitioning or feature selection is applied) but these samples are weighted taking into account the success of previous learner. Misclassified sample increases its weights ans subsequent trees will focus on them during training phase.

Suppose the training dataset is the domain D reported in Equation 3.1. The model requires a loss function L to quantify the prediction error accomplished by each learner in order to improve the performance of the subsequent learner. This function needs to be differentiable. The loss function commonly used is the aforementioned mean square error:

$$L = \frac{1}{2}(y - \hat{f}(x))^2 = mse \tag{3.15}$$

The reason why people choose this loss function for GB is that the user differentiates it with respect to predicted value $\hat{f}(x)$ the result is the easy to handle negative residual $-y_i + \hat{f}(x_i)$. Once the input are given, the first step is the initialization of model with a constant value $f_0(x)$:

$$f_0(x) = argmin \sum_{i=1}^{n} L(y_i, \hat{y}_i) = argmin \sum_{i=1}^{n} (y - \hat{f}(x_i))^2 \tag{3.16}$$

where i=1,...,n is number of samples in D. The constant value is the one that minimizes the sum of squared residuals. This value is just the average of output variables y_i. In other words, the first leaf created predicts the average values for all samples x_i. Successively, all

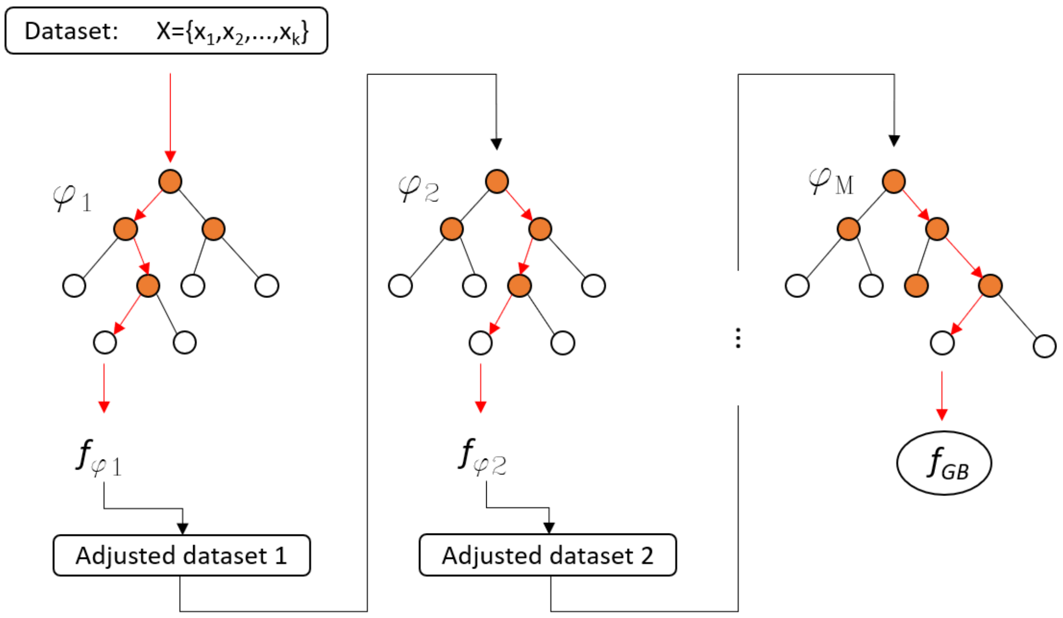


FIGURE 3.6: Gradient Boosting flow chart

trees will be created in loop and for each new tree the algorithm will compute:

$$r_{i,m} = -\left[\frac{\partial L(y_i, \hat{f}(x_i))}{\partial \hat{f}(x_i)}\right] \tag{3.17}$$

where m=1,...,M is the number of trees considered in GB model and $\hat{f}(x) = \hat{f}_{m-1}(x)$. Equation 3.17 can be rewritten as:

$$r_{i,m} = (y_i - \hat{f}(x_i)) \tag{3.18}$$

r is just a residual for each sample. GB fits a regression tree to the r_{im} values and create terminal region R_{jm} for $j = 1, ..., J_m$. The terminal regions R_{jm} are the leaves and j is the index for each leaf in the tree. The algorithm determines the output values for each leaf tacking into account the prediction of the previous tree $\hat{f}_{m-1}(x_i)$:

$$\hat{f}_{jm}(x_i) = argmin \sum_{x_i \in R_{jm}} L(y_i, \hat{f}_{m-1}(x_i) + \hat{f}) \tag{3.19}$$

and replacing with the actual loss function :

$$\hat{f}_{jm}(x_i) = argmin \sum_{x_i \in R_{jm}} \frac{1}{2}(y_i - (\hat{f}_{m-1}(x_i) + \hat{f}))^2 \tag{3.20}$$

GB finds the value for $\hat{f}$ that minimizes Equation 3.20. These values are just the average of residuals that ended up in the R_{jm} leaf. Finally, the new prediction for each sample is:

$$\hat{f}_{GB}(x) = \hat{f}_{m-1}(x) + \nu \sum_{j=1}^{J_m} \hat{f}_{jm} I \tag{3.21}$$

The GB response is based on the value of previous prediction add up the output value $\hat{f}_{jm}$ for all the leaves R_{jm} that a sample can be found in. ν is the learning rate of the model ($\nu \in (0, 1)$). A small learning rate reduces the effect each tree has on final prediction improving accuracy in the long run. GB graphical representation is reported in Figure 3.6

When comparing with RF, GB training process takes higher computational cost and hyper-parameters tuning requires a major effort in order to avoid over-fitting. However, benchmarks results have shown GB models outperform Random Forest in terms of prediction capability [68], [69].

3.3.3 Deep Learning

The quintessential deep learning model are *Deep feedforward networks*, also called *multi-layer perceptrons (MLP)*. A multilayer perceptron defines a mapping $y = f(x, \Theta)$, where Θ is the vector of neural network hyper parameters, and learns the value of Θ that results in the best function approximation [70].

Such model is called *feedforward* because the information flows from *x*, through the computations needed to define *f*, and finally to the output *y*. There are no feedback connections by which the outputs are feed back into itself (when a network has these connections is called *recurrent neural network* but they are still based on *MLP*).

In feedforward neural network the word "network" is used because the model is generally represented by composing many different functions and a directed acyclic graph describes how these functions are composed. In the case of multilayer perceptron there are at least three functions resulting $f(x) = f^3(f^2(f^1(x)))$ where the functions are called first,second and third layer. The length of this chain is a measure of the model *depth* from which the name "deep learning" arises. The first layer of a network is called input layer while the final layer is the output layer. The dimension of these layers are fixed by dimension of input vector *x* and output vector *y*. During the training of a neural network, each sample directly specifies what the output layer must do in fact it must produce a value that is close to *y*. Instead, the behaviour of other layers is not related to training data. The algorithm must decide how to use each individual layer to best implement the approximation function but the user doesn't know the output value for each of these layers, which, for this reason, are called hidden layers. MLP needs at least three layers: an input and an output layer with at least one or more hidden layers.

The layers are constituted by many elementary units that act in parallel. Each unit, called *neuron*, receives input, changes its status according to that input and produce output depending on the input and activation function. A single neuron is composed by three functional operators. First, an input vector $\mathbf{x} \in R^k$ is multiplied by a weight vector $\mathbf{w} \in R^k$ and the resulting vector is summed to the scalar bias b to form the scalar net input z. The final step is accomplished by the transfer or activation function $f : R \rightarrow R$ that produces the neuron output a,

$$a = f(z) = f(\sum_{i=1}^{k} x_i \cdot w_i + b) \tag{3.22}$$

The weight and the input z are the adjustable parameters of the neuron, producing different response to the same input. The transfer function must be set in advance and determines the type of neuron. The number of units constituting the layers defines the *weight* of the model.

When dealing with regression, the simplest models one can consider are the linear models as logistic regression and linear regression. To extend linear models in order to represent also non-linear function of x, one can applies the linear model not to x but to a transformed input $\theta(x)$ where θ is a non-linear transformation providing a set of features describing x, then providing a new representation of x. Before advent of deep learning the strategy to choose θ was to manually engineer θ requiring decades of human effort for each task. The strategy of deep learning is to learn θ. Following this approach, the model become $y = f(x; \Theta, w) = \theta(x; \Theta)^T w$ where Θ parameters are used to learn θ from a broad class of functions, while w maps from $\theta(x)$ to desired input. In other word, the user only needs finding the general function family rather than finding the right function fitting the data.

3.3.3.1 Anatomy of Multi-layer Perceptron

A feed forward neural network is constituted by the following objects:

- The *layers*, which are constructed using the neurons as building blocks;
- The *Activation function*, which determines the output of each neuron inside the network;
- The *Loss function*, which defines the feedback signal used for learning;
- The *Optimizer*, which determines how learning proceed.

In multilayer networks, since each neuron is parametrized by a weight vector and a scalar bias (see Equation 3.22), each layer will be parametrized by a weight matrix $\mathbf{W}_{R\times K}$, with R dimension of input vector $\mathbf{x}$ (i.e., the number of features describing the problem) and S specific layer neuron number, and a vector bias $\mathbf{b}$ of length S. For multilayer networks, the output of one layer becomes input for the following layer and this process, which is defined back propagation algorithm, is described by the following equations:

$$a^{m+1} = f^{m+1}(W^{m+1}a^m + b^{m+1}) \tag{3.23}$$

where $m = 0, 1, ..., M$ being M the number of layers in the network. The neurons inside the first layer (or input layer) receive input features:

$$a^0 = x \tag{3.24}$$

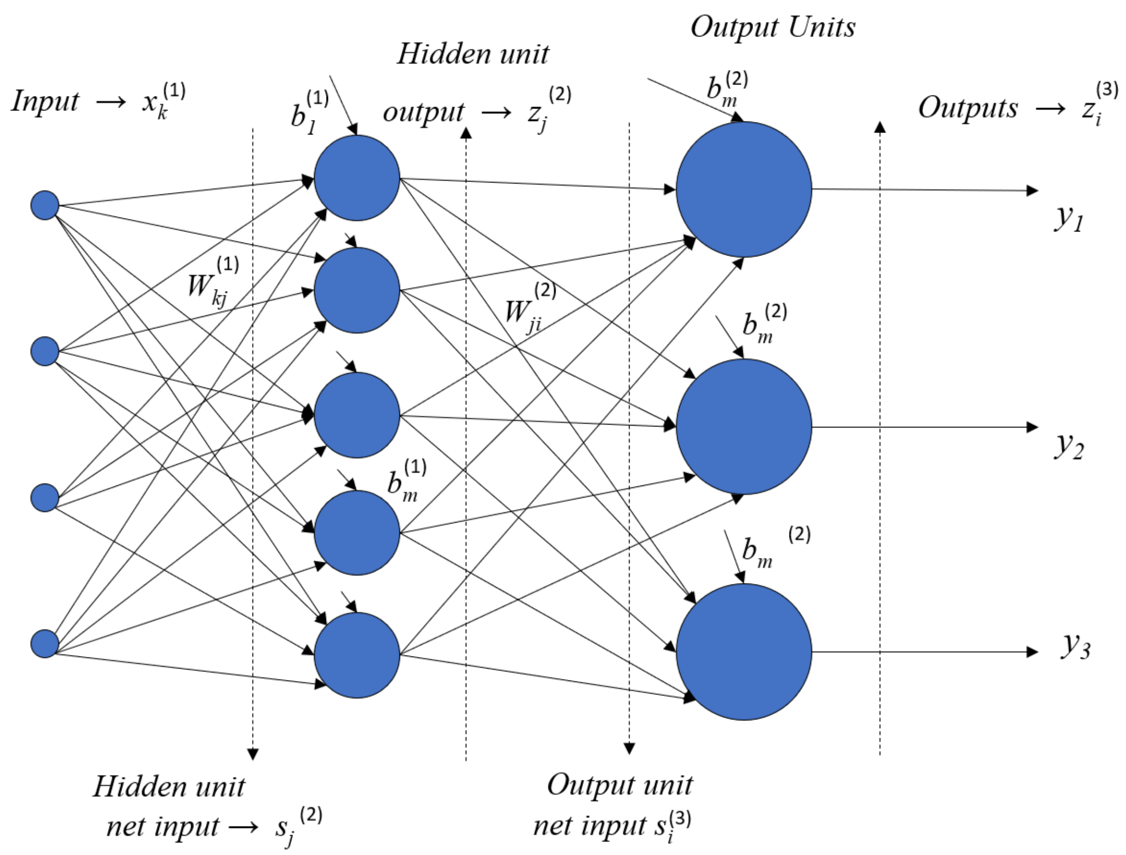


FIGURE 3.7: Three-layer feed forward neural network

which become the starting inputs for the following layer. The neurons in the last layer, or output layer, outcome the response of the network:

$$a = a^M = y \tag{3.25}$$

A sketch of multilayer perceptron is shown in Figure 3.7.

3.3.3.2 Activation function

In the back propagation algorithm, artificial neurons have in input parameters with associated weights and biases and the weighted sum of each "activated" unit, z, is the input value for the activation function. The activation function applies a transformation $f(z)$ to the end of establish whether or not a neuron should passes forward the value into the next layer.

A transfer function can be linear or non-linear function, however a linear function can learn only linear relation.

The simplest function is the *linear function*, $a = f(z) = z$. The output of such function is the weighted sum of inputs and is not confined between any range. The main disadvantages of such activation function are that the derivative is a constant value which determines a constant gradient descent. Moreover, being the activation linear, it is useless having more than one hidden layer. Generally this function is used in the output layer only, in order to permit neural network extrapolation since the function is everywhere defined.

The more used non-linear *Sigmoid function*, $\sigma(z) = \frac{1}{1+e^{-z}}$, is continuously differentiable and provides a fixed output range, $a \in (0, 1)$. However, when the value of z increases, the function becomes susceptible to vanishing gradient. This issue limits the learning accuracy. Sigmoid function is used especially in the last layer or binary classification models being the probability ranging in the output range of this function.

A modified version of sigmoid function is the Hyperbolic tangent activation function, $Tanh = \frac{2}{1+e^{-2z}}$. The output values are bounded between -1 and 1 enabling to consider values of different sign. This characteristic simplifies the optimization of neural network with more than one hidden layer. The main disadvantage of this function is that it still has vanishing gradient problem.

Nowadays, the most used activation function in regression problems is the *Rectified Linear Unit* (ReLU), $R(z) = max(0, z)$. The function outcomes 0 value for any negative input, while returns the input value for positive z (as a linear function). This function is still non-linear so the error can be back propagated and we can use multiple layers. Moreover, the function "rectified" the vanishing gradient problem, being the derivative 0 for negative input and 1 for positive input. It is less computational expensive when compared to Sigmoid an Tanh function. The main limitation is know as "dying RelU problem": for negative z the gradient will be 0 and, accordingly, weights will not get adjusted during the optimization process. Such neurons will stop responding to variation in input, becoming passive. A strategy to overcame this problem is using a modified version called Leaky RelU where the horizontal straight line for negative z has a small constant gradient.

3.3.3.3 Loss function

Loss function allows optimizing the network weight values using the gradient descent method. During a training process the objective is to minimize the distance between actual value and predicted value. Such distance is computed using a loss function. In regression problems, the most commonly used loss function is the *mean square error* (see Equation 3.12).

In some regression problems, the user may deal with distribution of target values containing outliers, e.g. large or small values far from the mean value of distribution. In these cases, using the *Mean Absolute Error*, $MAE = \frac{1}{n}\sum_{i=i}^{n} |y_i - \hat{y}_i|$, is an appropriate choice. This loss function results more robust to outliers.

3.3.3.4 Optimizer

The role of an optimizer is to iteratively update the weight and bias parameters to the end of minimizing the loss function.

The foundation of machine learning training and optimization is the *Gradient Descent* technique. The optimization procedure starts assigning a random value to each model parameter (Γ). The model is ran and the loss function is computed. The algorithm calculates the gradient of loss function ($g_t = \Delta L(\Gamma_{t-1})$) defining the direction to move the weights and bias to get a lower loss on the next iteration. A learning rate parameter (α), specified by user, control how much the coefficient should be updated at each iteration

$$\Gamma_{t+1} = \Gamma_t - \alpha g_t \tag{3.26}$$

When applied to non-convex loss function, gradient descent may converge into local minima. Moreover, since gradient descent needs to process all training samples before making the first update(full-batch optimization), this procedure results slow and computational expensive.

Stochastic Gradient Descent (SGD) decreases the computational time performing a parameter updating for each training example randomly selected. However, these frequent updates determine high variance in parameter values causing the loss function to intensely fluctuate.

Gradient descent with momentum accelerates SGD and reduces its oscillations introducing a momentum term γ that adds a fraction of previously updated vector V_{t-1} to the current parameter value:

$$\Gamma_{t+1} = \Gamma_t - \gamma V_{t-1} - \alpha g_t \tag{3.27}$$

Another strategy to speed up the optimization convergence is using an adaptive learning rate. Instead of updating all parameters Γ at once, using the same learning rate α, for every parameter $\Gamma(i)$, the algorithm computes different learning rate by dividing them by a decaying term, which is the sum of squared gradient estimate over the training phase (ϵ is a vector of small numbers to avoid dividing by zero):

$$\Gamma_{t+1} = \Gamma_t - \frac{\alpha}{\sqrt{\epsilon + \sum g_t^2}} g_t \tag{3.28}$$

Nowadays, the most used optimizer is the *Adaptive Moment Estimation (Adam)* [71]. The method computes adaptive learning rates for each parameter and, in addition, individual momentum changes. The method estimates first $\hat{m}_t$ and second $\hat{v}_t$ moments of gradient to adapt learning rate for each parameter:

$$\Gamma_{t+1} = \Gamma_t - \frac{\alpha}{\hat{v}_t} \hat{m}_t \tag{3.29}$$

with:

$$\hat{m}_t = \frac{(1-\beta_1) \sum \beta_1^{t-1} g_t}{(1-\beta_1^t)} \tag{3.30}$$

$$\hat{v}_t = \frac{(1-\beta_2) \sum \beta_2^{t-1} g_t^2}{(1-\beta_2^t)} \tag{3.31}$$

Betas are new hyper-parameters introduced by the method. Default values are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The moving average vectors $\hat{m}_t$ and $\hat{v}_t$ are initialized with zero value.

In the last few years, Adam reached so high optimization performance in many deep learning application that it is currently recommended as the default algorithm to use[72], [73].

It is worth noting that, when the training dataset is relatively small, the optimizers in the family of quasi-Netwon method, which replace the Hessian in Newton's method with an approximated Hessian, can outperform Adam. This is the case of Limited Memory BFGS [74] that approximates the Broyden-Fletcher-Goldfarb-Shanno (BFGS) using a limited amount of computer memory. This optimizer performed better than Adam in the deep learning application reported in Chapter 7.

3.4 Dimensionality reduction

Dimensionality reduction is one of most important sub-field of unsupervised learning. The dimensionality of a dataset is the number of features required to describe the regression problem. In a column-row format dataset, the dimensionality is the number of column, while each row identifies a training sample. The goal of a dimensionality reduction algorithm is to analyse the hidden structures of a dataset in order to reduce the number of required features and in so doing the complexity of the problem.

It is worth noting that dimensionality reduction differs from features selection. Both methods seek to reduce the number of attributes in the data set, but dimensionality reduction assesses this task creating new combinations of attributes (i.e., extracting new variables), whereas the selection of feature aims to include and exclude attributes already existing in the dataset without creation of any new feature.

Two most used algorithms for feature extraction are: *Principal Component Analysis* and *Projection to Latent Structures.*

3.4.1 Principal Components Analysis

PCA is a multivariate statistical method that reduces the dimensionality of the data while retaining most of the variation in the data set [75]. An orthogonal transformation allows to convert a set of n observations, containing possibly k correlated variables, into a new set of values of linearly uncorrelated variables defined as principal components, which are linear combinations of the original variables k [76].

The first principal component explains the largest possible variance representing the direction along which the samples show the largest variation. First component can be seen as the best fit line through the swarm of points. The algorithm determines the best fit line minimizing the distances between each point and this line (see Figure 3.8) generally using the Lapack implementation of Singular Value Decomposition (SVD). However, for large and redundant dataset SVD may be inefficient, while the non linear partial least squares (NIPALS) algorithm results more stable [77].

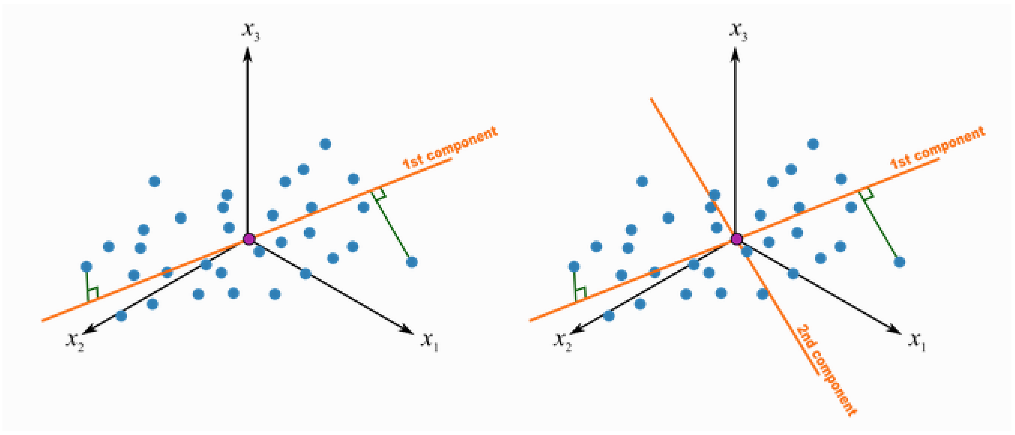


FIGURE 3.8: Geometric interpretation of PCA

The second component is computed under the constraint of being orthogonal to the first component and to have the largest possible variance [78]. The following components, constructed with the same criterion, account for as much of the remaining variability as possible. The amount of explained variance increases with the number of considered component. The scree plot with explained variance against component's number can be use to select the proper number of components. Formally, PCA is based on a decomposition of the data matrix X into two matrices T and P:

$$\begin{aligned} X &= TP^T \\ (n \times k) &= (n \times a)(a \times k) \end{aligned} \tag{3.32}$$

where the matrices T and P are orthogonal. P is the loading matrix, while T is the scores matrix. k is the number of features of original set, n is the number of samples and $a < k$ is the number of principal components considered. During the analysis, data inside X are

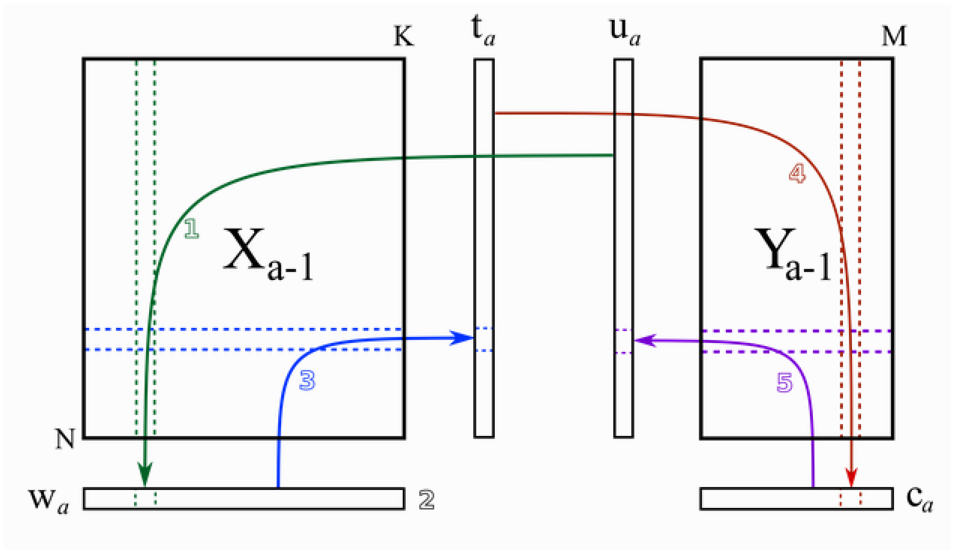


FIGURE 3.9: Nipals algorithm in PLS computation

considered being part of same group. There is no subdivision of features into input features and output features.

The loadings are the weights of each original feature when calculating the principal component. Loading values quantify the statistical importance of features for dataset variability. Highly correlated features have almost same weights in the loading vectors and appear close together in the loading plots, which are loading of a principal component plotted against the loadings of another component.

The scores represent the original data in a rotated coordinate system, the space of principal components. The scores are the extracted features able to describe the original set into the new lower dimension space. In so doing, the user can represents the original dataset by a reduced number of new features

3.4.2 Projection to Latent Structures

Projection to Latent Structures (PLS) acts on the data similarly to the PCA with the main difference that the variables of original data-set are divided in input features and output features, while the analysis aims to find relations between these two groups. Actually, PLS will find the multidimensional direction in the input variables space that defines the maximum multidimensional variance in the output variables space [79]. In its general form PLS creates orthogonal score vectors (also called latent vectors or components) by maximizing the covariance between these different sets of variables [80], [81]. The outcomes are one loading and one score vector for input and output features, respectively:

$$\begin{aligned} t_a &= X_a w_a \\ u_a &= Y_a c_a \end{aligned} \tag{3.33}$$

where t_a and w_a are score and loading vector for input features X, while u_a and c_a are score and loading vector for input features Y.

PLS model computes loadings and scores iteratively by means of Nipals algorithm (see Figure 3.9). First, the algorithm regresses each column of X_a onto the vector u_a. In the first step u_a is estimated by a column from Y_a. The coefficients of these regression are stored in $w_a = \frac{1}{u_a' u_a} X_a' u_a$. Columns strongly correlated with u_a have large weights in w_a.

Second, the weight vector is normalized, $w_a = \frac{w_a}{\sqrt{w_a' w_a}}$. Third, every row in X_a is regressed onto w_a and the regression coefficients are stored in $t_a = \frac{1}{w_a' w_a} X_a w_a$. Four, algorithm regresses every column in Y_a onto the score vector t_a and stores regression coefficient in $c_a = \frac{1}{t_a' t_a} Y_a' t_a$. Finally, each row of Y_a needs to be regressed onto c_a in order to compute for first time $u_a = \frac{1}{c_a' c_a} Y_a c_a$. This process is repeated until the vector u_a reaches convergence values. On such plot, the user can draw a threshold line ($t(a)$) computed by means of correlation coefficient onto the a-latent variable considered:

$$t(a) = m|w_a, c_a| + [M|w_a, c_a| - m|w_a, c_a|] \cdot [1 - corr(t_a, u_a)] \tag{3.34}$$

where m and M are the minimum and maximum value, respectively, while *corr* indicates the correlation coefficient. After building a PLS model, the most informative result can be achieved through the loading plots, in which the user superimposed a plot of input feature loadings on a plot of output feature loadings. This plot furnishes the latent variable relationship between the input feature space and the output feature space, in addition to relationship between input variables and between output variables.

3.5 Clustering

The second main application of unsupervised learning is Clustering analysis. The objective of clustering is finding different structures within the dataset to the end of grouping similar samples into the same group. Some of the most common clustering algorithms are: k-Means and DBSCAN.

3.5.1 k-Means

Hundred of algorithms have been developed to solve clustering problem, however k-Means remains the most popular clustering method. The main advantage is its simplicity: starting randomly selecting initial centres, the algorithm repeatedly assigns each sample point to the nearest center, and then the centres are recomputed according to sample assignation [82]. From a theoretical point of view, k-Means is not the best algorithm in terms of efficiency or quality, but in practice, speed and simplicity of k-Means cannot be beat [83].

The algorithm aims to group similar samples inside the same class. This similarity is parametrized using the squared euclidean distance, which, between a couple of points x_0 and x_1, is:

$$d(x_0, x_1)^2 = \sum_{j=1}^{m} (x_{0j} - x_{1j})^2 = \| \mathbf{x}_0 - \mathbf{x}_1 \|_2^2 \tag{3.35}$$

where j is the sample point dimension (i.e., feature column). Each point is assigned to the closest randomly selected centroid using this distance. Successively the algorithm computes *Sum of Squared Errors* within each cluster (or *cluster inertia*) as follow:

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{k} w^{(i,j)} \| \mathbf{x}^{(i)} - \mu^{(j)} \|_2^2 \tag{3.36}$$

where $\mu(j)$ is the centroid for cluster j, while $w(i, j)$ is 1 if the ith sample is inside jth cluster, 0 otherwise. 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 k-Means hyper-parameters the user must define are: number of clusters (m), maximum iterations, initial cluster centres. The initial number defines the number of times the algorithm will run with different centroid seeds. To select the proper number of clusters, user may use the *elbow method*, which consists plotting the ascending number of clusters m against the total variance explained. The k-Means algorithm reaches high performance if applied to dataset with a kind of spherical shape. As well known, a proper initialization of initial centroids is crucial for obtaining a good solution. In order to overcame the random selection of initial centroids, the *k-Means++* has been proposed in which the initial selection of centroids is based on the following phases: the first center is chosen uniformly at random among the data points. For each sample x_i, the distance between the point and the centroid is computed, $D(x_i)$. One new data point is randomly selected as second centroid using a weighted probability distribution, $P = f(D(x_i)^2)$. This procedure is repeated until all the centroids are initialized [84].

3.5.2 DBSCAN

The *Density-based spatial clustering of applications with noise* (DBSCAN) [85] is a clustering algorithm that uses the distance between nearest points to identify different group of data. DBSCAN is based on the assumption that clusters are dense regions in the data space separated by lower density regions. The key idea behind this algorithm is that for each point inside a cluster, the neighbourhood identified by a given radius (*epsilon*) has to contain at least a minimum number of points (*minimum points*). The user must set these two parameters before using DBSCAN algorithm. Each data point having a neighbour count greater than the minimum points is labelled as a cluster *core point*, while a sample is a *border point* if it has less than minimum neighbour points but it belongs to the neighbourhood of a core point. DBSCAN works through the following steps:

1. for each point x_i inside the dataset the distance between such point and other samples is computed. All points within distance epsilon are considered neighbours of x_i. The points having a neighbour number major or equal to minimum points are labelled as core points

2. each core point, if it is not already part of an existing cluster, generates a new cluster. The algorithm will find all the density connected points in order to assign them to the same cluster

3. the algorithm iteratively analyses the remaining unvisited points. Points that don't belong to any cluster are considered as outliers (i.e., noise).

The main advantage of this clustering algorithm is that the number of clusters to be generated is not a required input. Another important aspect is that DBSCAN can process any shape of cluster. Finally, this method can identify outliers.

On the other hand, DBSCAN does not perform well when applied over clusters with different densities and user needs to carefully select its required parameters. The value of minimum points increases increasing the size of dataset and generally this value must be at least 3. To select the proper epsilon the user may use the k-distance graph where the distance to the k minimum points is plotted against k value.

3.6 Model Evaluation

An essential step in machine learning is the model training. The user needs to evaluate how well the model fit the dataset and, more importantly, how well the trained model generalizes on unseen data. Generalization is the ability of model to outcome sensible outputs to sets of never seen inputs.

During training phase, the user need using data to incrementally improve the model prediction ability. To this end, the original dataset needs to be divided into training and validation set. The algorithm will be trained using the samples inside the first set, while the second set will be used to evaluate the model accuracy.

3.6.1 Training and Validation

The simplest evaluation method is the *holdout*. The dataset is randomly divided into three subset: training set, validation set and test set. The predictive model is trained using the training set samples, while the validation set is used as test platform for meta-model hyper-parameters tuning. Generally, not all algorithms require a validation set. Finally the test set, containing unseen data, is used to evaluate the future performance of a model. The results obtained during test phase characterize the model accuracy. The holdout approach is easily implementable and has high flexibility, but has high variability because different training and test sets may determine significantly different models.

A more accurate validation method used to overcome the limitation of previous strategy is *k-fold cross-validation*. This method requires to randomly divide the dataset into k different groups, also known as folds, each one having approximately equal size. One fold constitutes a validation set, and the method is fitted on the remaining *k-1* folds [86]. This process is iterated k times obtaining k different training and testing sets (see Figure 3.10). Common values generally used are $k = 5$ and $k = 10$. The estimation of accuracy is averaged over the results obtained for all training-test couples,

$$\hat{y} = \frac{1}{10}\sum_{i=1}^{10}\hat{y}_i \tag{3.37}$$

Cross validation significantly reduces bias, since the method uses most of data for training phase, and it also reduces variance, since all the data are also used during test phase. Finally, the interchanging of test and training samples improves the effectiveness and robustness of this validation strategy. Having to train and validate k different models, the cross validation strategy may result computationally expensive.

The objective of a train-validation process is obtaining a model you can trust when applied to predict never seen data. The highest risk during the training phase is *overfitting the dataset*. The key aspect of this phenomenon is the *information leaks*. When the user optimizes an hyper-parameter, information about the validation dataset are coming into the model. If this procedure is continuously repeated and the hyper-parameters iteratively

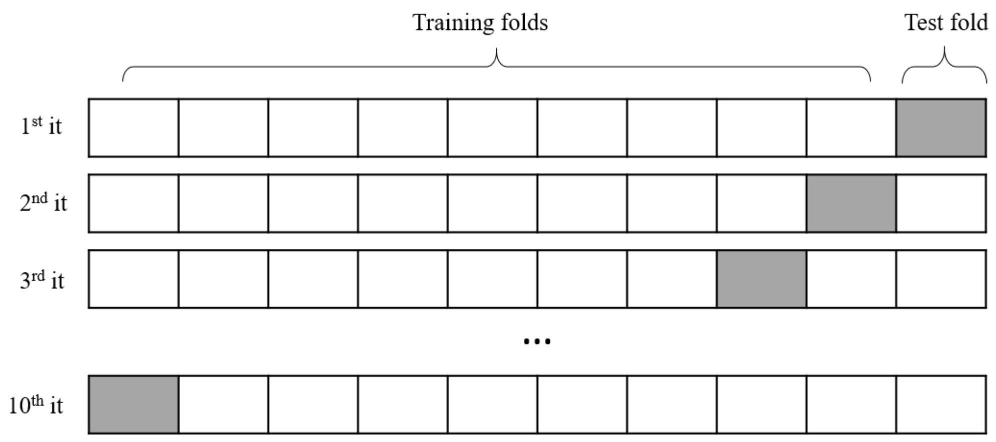


FIGURE 3.10: 10-fold cross validation

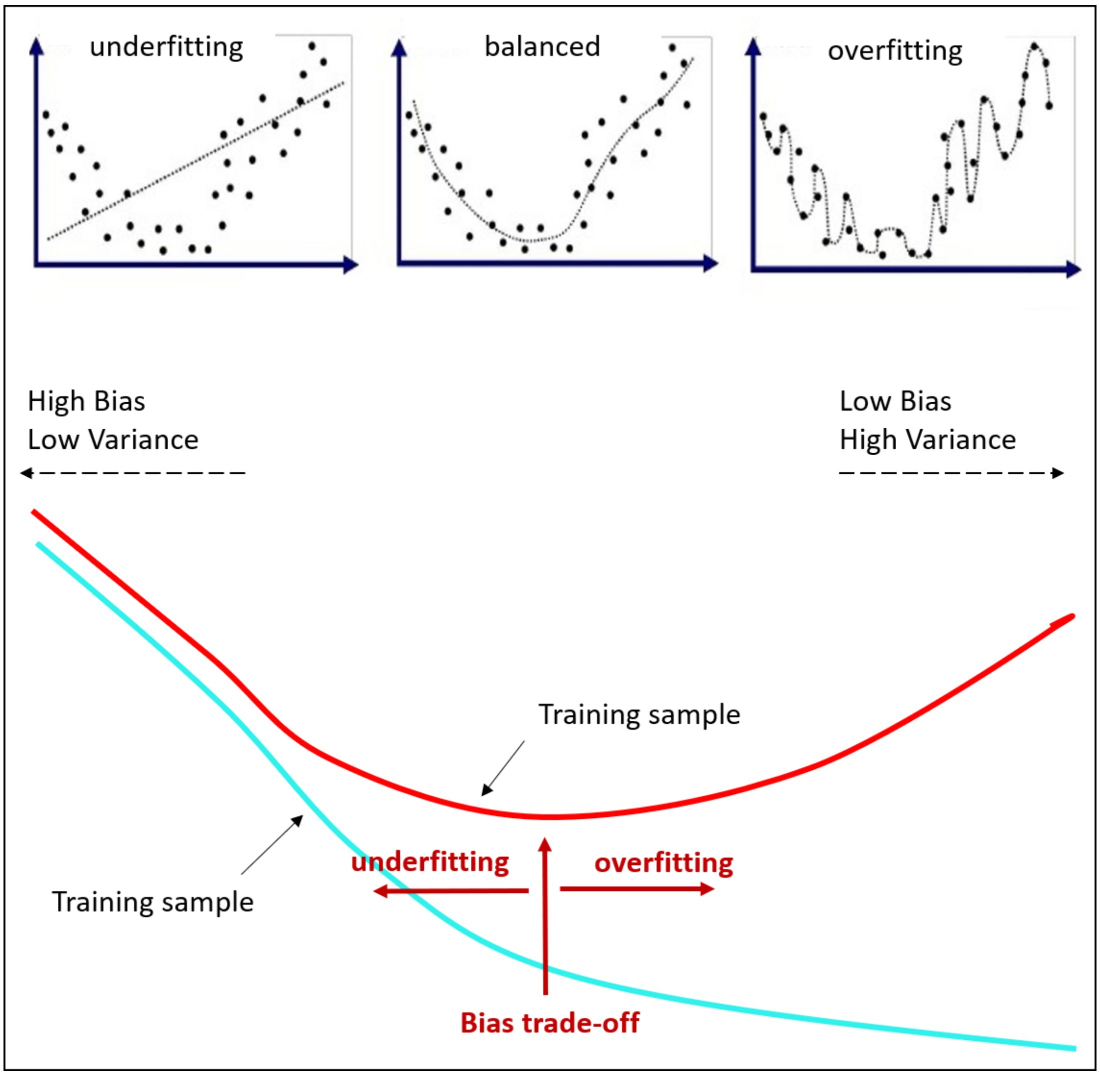


FIGURE 3.11: Bias-variance trade off

adjusted, at the end of this optimization the model will perform well on the validation set but it will not able generalizing over the test dataset. In such situation, the response of model applied on new input data will be extremely not accurate.

3.6.2 Overfitting and Underfitting

Dealing with overfitting and underfitting is unavoidable in any machine learning problem. The main responsible of such behaviour is the tension between optimization, which refers to adjust the model parameters to get best performance on training set, and generalization, the ability of model to perform well on new data. At the beginning of training phase, the model is in underfitting conditions. Optimization and generalization are correlated. The model still has to improve its prediction and iteratively decreasing the loss on training set, the loss on test set decreases (see Figure 3.11). In this preliminary phase the model is said to have high bias (i.e, model inability to capture the true relationship between data) and low variance (i.e., the model difference in fits between different datasets). After several training iterations, test loss stops improving. In this phase the model starts overfitting: the model is learning patterns that are specific of training set but that result misleading for new data prediction. When overfitting occurs, the model has low bias because the model fits so well the training set that there is no distance between predicted and real value. On the other hand, it has high variance because the prediction error becomes high when the model is applied to test dataset.

Ideally, the best solution to improve accuracy and generalization capability is to get more training data. If it is not possible, the user should modulate the quantity of information the model is allowed to store. This process is called bias-variance trade off (or model *regularization*). If a model has a limited storage capability, it will focus only on most prominent patterns, which have a better chance of generalizing well.

3.7 Problem-specific data flow

All the necessary phases (see Figure 3.12) needed to correctly develop and use any machine learning algorithm are listed in the following:

1. *Defining the goal*: the first step in any machine learning project is to correctly define the problem. The most powerful algorithm will perform poorly if applied to the wrong problem.

2. Define a set of features that influences the output- this phase is called "feature engineering". The success of all machine learning algorithms depends on how you present the data [87]. Data features directly influence the predictive models and the results one can achieve. Well engineered features improve the problem flexibility allowing to use less complex models, which result faster to run and easier to understand, keeping good predictive performance. In this preliminary phase you have to turn your inputs into things the algorithm can understand [88].

3. *Generation of training dataset*: all machine learning algorithms learn from data. Having the right training dataset according to the specific problem is a critical task. Generally, the process for preparing data before machine learning application involves the following steps: (i) collect data, (ii) pre-process data and (iii) transform data. All the available data should be evaluated to the end of selecting a subset of data that the user will be working with. Intuitively, one may think including all the available data is the better choice. Depending on specific problem, this may or may not be true.After the data collection, the user should pre-process data. Generally, in this phase the user need formatting the data (i.e., collecting data in a relational database or selecting a format that is suitable for specific application), cleaning the data, removing unnecessary or corrupted data or fixing missing data, and eventually sampling the data. In many application, taking a small sample of data may be useful for faster exploring and prototyping solutions before considering the whole dataset. The last step involves the transformation of data according to machine learning algorithm and the knowledge of specific problem. Common transformation are scaling (i.e., the preprocessed data contain attributes with mixtures of scales for various quantities or very different magnitude in the same scale. These features may be transformed to have the same scale such as between 0 and 1 for the smallest and largest value respectively.), feature decomposition, as the case in which the dataset includes features representing a complex concept that may be better exploited by a machine learning method when split into the constituent parts (for example, a feature is the date that have day and time but only the hour of a day is relevant to the specific problem), and feature aggregation, when several features can be joined into a single more meaningful feature.

4. *Design and optimization of a machine learning algorithm*: once data preprocessing has been completed, the user needs to select a machine learning algorithm in order

to solve the problem. This phase requires choosing, running and tuning the algorithms then the user needs defining a test harness to evaluate the performance of different algorithms tested against a fair representation of the problem to be solved. The outcome of testing multiple algorithms will be an estimation of how a variety of algorithms perform on the selected problem giving indication on which of them might be worth tuning and which of them should be discarded. Moreover, the results may furnish an indication of how learnable the problem is. To evaluate the algorithm ability in solving the problem, the user needs to measure the prediction performance made by a trained model on the test dataset. Each class of problem (i.e., classification, regression, clustering) has standard performance indicators which will score its meaningfulness to the domain of the user's problem. The preprocessed data need to be divided in test and training set. The algorithm will be trained on the test dataset and will be evaluated against the training dataset. Several selecting methods are available to define the sub-set just mentioned. They range from a simple random split of data (i.e., 70% for training, 30% for testing) to more complicated sampling method as the Cross Validation. Once the test harness has been defined, the user can quickly check a variety of machine learning algorithm and can select the more effective algorithms according to problem. At this point, the algorithm that perform better needs to be tuned in order to get better results. Modifying the machine learning parameters (or Hyper-parameters) can influence the prediction performance. If the algorithm is parametrized by n-parameters you will have an n-dimensional hypercube of possible configurations for the selected algorithm. The user can exploit several optimization methods to set the parameters, however, it is worth noting that the more tuned the parameters of an algorithm, the more biased the algorithm to the training data and test set leading to a less robust model that overfits the test set.

5. *Derivation of a learner*, once the algorithm has been tuned, the final model is obtained using the algorithm to make prediction on new data. Given new input data, the model outcomes the expected response that may be either assigning a label in a classification problem, or predicting a real value in a regression problem.

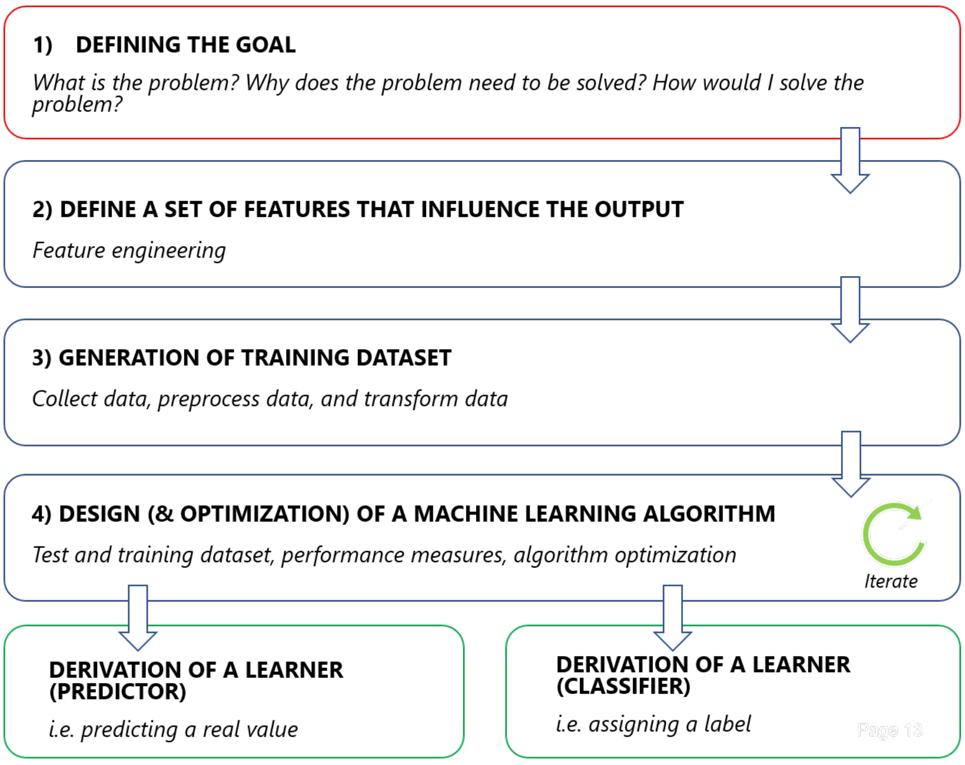

1) DEFINING THE GOAL
What is the problem? Why does the problem need to be solved? How would I solve the problem?
2) DEFINE A SET OF FEATURES THAT INFLUENCE THE OUTPUT
Feature engineering
3) GENERATION OF TRAINING DATASET
Collect data, preprocess data, and transform data
4) DESIGN (& OPTIMIZATION) OF A MACHINE LEARNING ALGORITHM
Test and training dataset, performance measures, algorithm optimization
Iterate
DERIVATION OF A LEARNER (PREDICTOR)
i.e. predicting a real value
DERIVATION OF A LEARNER (CLASSIFIER)
i.e. assigning a label

FIGURE 3.12: Methodology data flow

Chapter 4

Relevance of data treatment

"On fait la science avec des faits, comme on fait une maison avec des pierres: mais une accumulation de faits n'est pas plus une science qu'un tas de pierres n'est une maison."

Science is built of data the way a house is built of bricks; but an accumulation of data is no more a science than a pile of bricks is a house.

Henri Poincare

4.1 Importance of Data

The success of any machine learning application depends largely on training data "quantity and quality". Every machine learning campaign starts with *data mining*, the process of collecting data related to the specific problem the user wants to solve. These data may come from experimental observations, numerical investigations, existing databases or they can be generated by the user. Referring to the example reported in chapter 1 (see Figure 1.7), it is clear that even a deeply optimized model will perform bad if the quantity of training data is not sufficient.

Obviously, the quantity of data is not the only one critical aspect to be evaluated. If training data include non representative data (i.e., sampling bias), outliers or noise (e.g., data coming from poor-quality measurements), it will be harder for any machine learning algorithm to find underlying structures and perform well.

Another important aspect to be considered is the number of relevant features needed to describe each training sample inside the dataset. The prediction model will perform well only when training data does not contain too many irrelevant features. Irrelevant features increase the complexity of model and reduce the chance of generalizing well over unseen data. The process of defining a good set of features is called feature engineering. This process involves two main aspects:

- *feature selection* : selection of most useful features from a set of existing features (in the following section tools for feature selection will be introduced);
- *feature extraction* : creation of new more representative features combining existing features (i.e., dimensionality reduction methods as *PCA* and *PLS*).

Each data scientist should spend a great effort to improve the quality of such data before performing machine learning applications. This phase is known as *Exploratory Data Analysis* and starting from this analysis, the user will implement a series of actions (i.e., outliers removal, distribution shape improvement or irrelevant features reduction) in the *data preprocessing* (see Figure 4.1).

Generally, data treatment requires spending more time than the time required by the development of entire prediction model however this preliminary phase impacts on final result quality more than any hyper-parameter optimization.

4.2 Normalization

Preliminary step before any deeper analysis is the data normalization or scaling. The procedure aims to bring all the columns (or features) of a dataset into the same range. The most common normalization technique are *min-max normalization* and *Z score*.

The first strategy is a simple scaling procedure:

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

where z is the scaled feature vector. Normalization transforms data from the original range to a new range between 0 and 1.

Z score is a standardization procedure that transforms the data as follows:

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

where μ is the mean value of the original dataset and σ is the standard deviation. This transformation should be applied to data featuring a Gaussian distribution in order to obtain a a new centered Gaussian distribution with standard deviation of 1 (also known as *standard Gaussian distribution*).

4.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to data analysis that exploits several statistical techniques, most of which are graphical techniques, to: maximize insight inside dataset, uncover underlined structures, detect outliers and anomalies, extract the most relevant features.

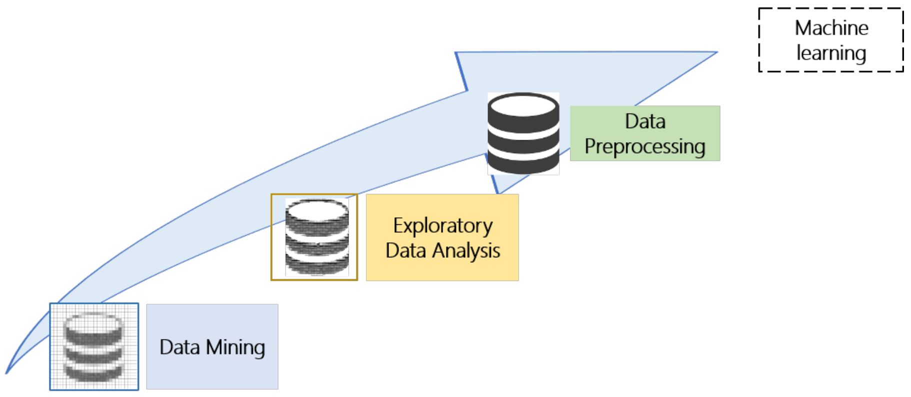


FIGURE 4.1: Data treatment flow

4.3.1 Removing Outliers: Box and Whiskers Plot

Box and Whiskers plot (or Boxplot) is a standardized statistical tool developed by [89] to display the distribution of data using five numbers: *minimum*, *lower quartile* (Q_1), *median*, *upper quartile* (Q_3) and *maximum*. The median is the middle value of the i-th dataset column. Lower and upper quartiles are the 25^{th} and 75^{th} percentile, respectively. They define the extreme sides of the box. Thus the box contains the 50% of samples and is called inter quartile range (IQR). The median identifies the line drawn inside the box. The whiskers extends from the extreme sides of the box to minimum and maximum values, which are:

$$\begin{aligned} min &= Q_1 - 1.5 \times IQR \\ max &= Q_3 + 1.5 \times IQR \end{aligned} \tag{4.3}$$

It is worth noting that minimum and maximum are not the lowest or biggest value into the dataset. All samples smaller than minimum or bigger than maximum are outliers. In a boxplot they are drawn as individual points in-line with whiskers (below minimum or upper maximum). Figure 4.2 shows the sketch of a boxplot. Boxplot is generally used to detect outliers, which are samples having values too much out of range to be considered simply anomalies or borderline configurations. More likely, outliers may come from errors occurred during the dataset generation (i.e., errors during the experimental campaign).

Moreover, boxplot allows visualizing the data distribution through their quartiles, which has the advantage of taking up less space if compared with either histogram or density plots. This is useful when comparing distribution of multiple features. It is worth noting that reducing outliers from a dataset acts also on the shape of distribution. In statistics, the *Kurtosis* indicator is used to describe the extension of the distribution tails. Long tails indicate the presence of outliers and such tails affect the symmetry of data. Removing outliers means reducing tail dimension.

4.3.2 Distribution Shape Improvement

A lot of statistic methods assume a Gaussian (or normal) distribution of data: the well known "bell shaped" curve. When dealing with data having such distribution, the parametric methods are powerful and generally they perform better. This is the reason why

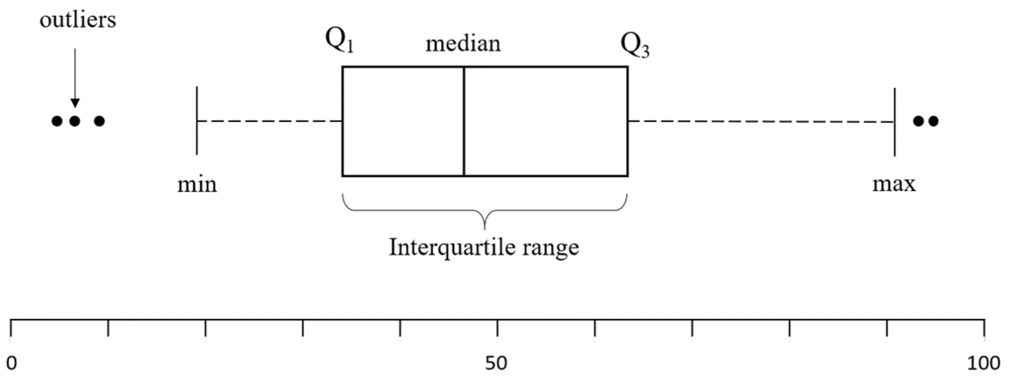


FIGURE 4.2: Box and Whiskers plot

data scientists incentive to use normal distribution if possible. This need becomes even more important if we are trying to solve a machine learning problem.

Generally, there are many reasons why the dataset may not be Gaussian. The most common reason is that the size of dataset is to small. Another reason may be the presence of extreme values that may generate long tails on a distribution. Even if the data shape is not Gaussian, there are several techniques that can be applied to reduce the distribution asymmetry (or *skewness*) making a distorted data sample more normal.

4.3.2.1 Data Skewness

The skewness measures the degree of distortion from the Gaussian distribution. Skewness quantifies the lack of symmetry in data statistical shape. A symmetric distribution has skewness equal to 0. In such distribution, mean and median coincide. In non symmetric distributions the data trail off more sharply on one side respect to the other. A non symmetric distribution can be positive skewed (also known as right skewed) or negative skewed (also known as left skewed). Positive skewed data have a long tail that extends to the right side of distribution. The mean and median are greater than the mode, which is identified by the highest histogram inside the distribution, i.e., the most observed value. Negative skewed data present an opposite trend with mean and median lower than mode, while the distribution tail is longer on the left hand left side. Figure 4.3 summarises the different categories.

Generally, a distribution with a skewness value above +1 or below -1 is said *highly skewed*. When such value lies between +0.5 to -0.5 the distribution is *moderately skewed*. The problem of dealing with skewed distribution is that statistical methods generally assume approximately normal distribution of data. Using such methods on skewed data may figure out misleading answers.

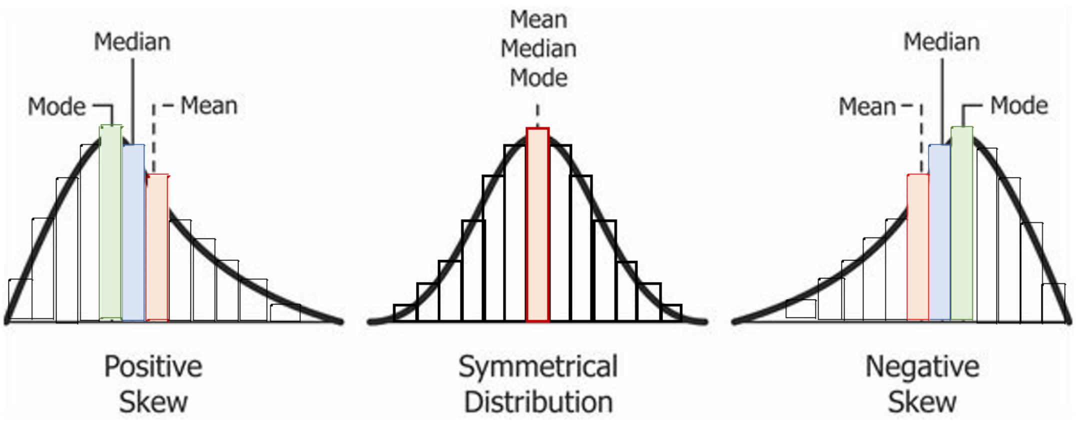


FIGURE 4.3: skewness

A solution to this problem is transforming data to make a skewed distribution more Gaussian-like. Each data value will be replaced by a new number, which is function of the original value, to improve the effectiveness of statistical analysis.

4.3.2.2 Transformation Functions

Before performing any transformation, the user should know the category to which skewed data belong.

To improve positive skewed data the most common transformations are: *square root*, *cube root* and *logarithm*. The cube root transformation, $x' = x^{\frac{1}{3}}$, can be applied to both positive and negative values. The transformation has a great effect on distribution shape, but is less powerful than logarithmic transformation. Square root, $x^{'} = x^{\frac{1}{2}}$, can be applied to positive values only. The user need evaluate the values in each columns before using such transformation. Logarithm transformation, $x^{'} = log(x)$, is the more powerful strategy to improve skewness among transformation, but can be applied only to strictly positive values.

To improve negative skewed data, we can use the *square transformation* as well. Square transformation, $x^{'} = x^2$, can be use to moderately reduce left skewness. Detailed recommendations regarding the use of traditional transformations have been reported by [89].

Traditional transformations are simple and computational inexpensive, however they present limits regarding the applicability range and the effectiveness of non-normality reduction. A new family of transformations have been developed to overcome these limits, the *power transformations*. Power transformations are functions that raise numbers to an exponent. Several traditional transformations are members of this class, however the idea behind the new strategy is using a potential continuum of transformation that provides a wide range of opportunities for customize a transformation to the target dataset [90].

The first of such functions is the Box-Cox transformation [91]. In particular, the two-parameters version of such transformation is defined by

$$\Psi^{BC} = \begin{cases} (x+\lambda_2)^{\lambda_1} - 1)/\lambda_1 & (if \quad \lambda_1 \neq 0) \\ log(x+\lambda_2) & (if \quad \lambda_1 = 0) \end{cases} \tag{4.4}$$

where the λ parameters are estimated using the *profile likelihood* statistical method. This transformation can be applied regardless of whether a distribution is positive or negative skewed. Moreover, it can be used for both positive and negative x by introducing the parameter λ_2 that has not been previously included in the the first formulation.

Another family of distributions has been more recently proposed by [92]; the Yeo-Johnson transformation. This transformation can be applied without limitations on x and exploits several properties of Box-Cox transformation. The Yeo-Johnson includes the following functions:

$$\Psi^{YJ} = \begin{cases} [(x+\lambda)^{\lambda} - 1]/\lambda & (if \quad \lambda \neq 0, x \geq 0) \\ log(x+1) & (if \quad \lambda = 0, x \geq 0) \\ -[(-x+1)^{2-\lambda}] & (if \quad \lambda \neq 2, x < 0) \\ -log(-x+1) & (if \quad \lambda = 2, x < 0) \end{cases} \tag{4.5}$$

When applied to strictly positive x, the Yeo-Johnson transformation is similar to Box-Cox applied to $(x + 1)$. If x is strictly negative, the transformation results equal to Box-Cox transformation of $(-x+1)$, but with power $(2-\lambda)$. For all other values the function results a mixture of the above cited transformations.

4.4 Data dimension

Often when training-set of a machine learning model comes from an existing database, the user have to deal with a plethora of features used to describe each sample inside the dataset. Generally, all the features might not be necessary in building the prediction model and, in same cases, they also reduce its prediction performance.

A required step before building a machine learning model is the feature selection. This process has a huge effect of model prediction capability. It is worth noting that, as already mentioned, the feature selection process does not involve the generation of new feature. It simply select the most relevant features according to a specific criteria: the feature correlation coefficient.

4.4.1 Correlation Matrix Analysis

Correlation is a statistical technique to quantify the relationship between two quantitative, continuous variables. Correlation coefficient measures both strength and direction of linear dependence between each couple of features in the dataset.

Correlation coefficient, also known as Pearson correlation, can be obtained by dividing covariance (cov) of a couple of variables by the product of their standard deviations (σ):

$$\rho_{XY} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{4.6}$$

This procedure scales the covariance value into the range from -1 to +1, which is the correlation value range. Directly correlated features have a correlation coefficient +1, while the value of this coefficient reaches -1 if the features are inversely correlated. Finally, not correlated features have near zero value. Generally, features having high value of the correlation coefficient are more linearly dependent and hence produce nearly the same effect on either the dataset variability or the dependent variables.

Correlation matrix can be rapidly analysed by means of heat map (see Figure 4.4). It is a graphical representation in which each element inside the matrix is represented using a colour. The heat map uses a warm-to-cool colour spectrum to show the value of correlation between features.

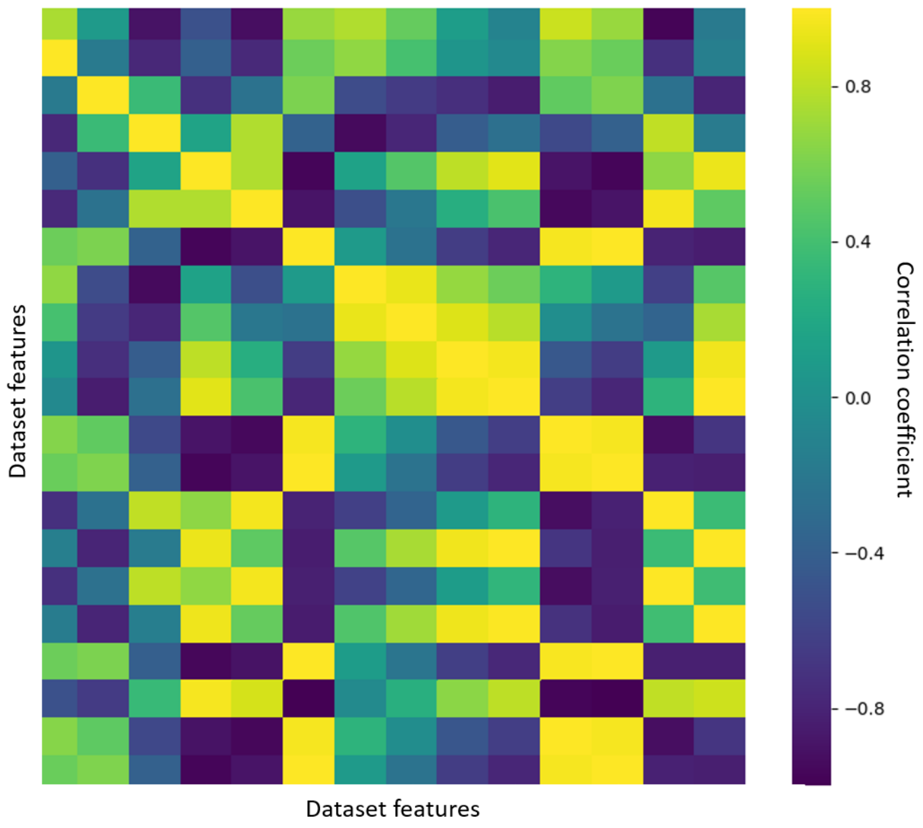


FIGURE 4.4: Heat map representation of correlation matrix

Chapter 5

Multi-Dimensional Extension of Balje Chart for Axial Flow Turbomachinery Machine Learning Based

5.1 Introduction

Following machine learnt application aims to explore the axial flow turbomachinery design space to build a set of multi-dimensional Balje charts, which will provide new design procedures that allow defining the blade geometry parameters as hub-to-tip ratio, blade count, solidity, twist distribution. The multi-dimensional Balje charts are built using correlations between rotor performance and enriched geometric and kinematic parameters.

To find such correlations, firstly the axial flow rotor-only design space has been explored using AxLab and a population of axial flow fans has been derived. To improve the fidelity level of fan performance data, AxLab has been previously enhanced exploiting a meta-model for deflection prediction of airfoil in cascade configuration.

Successively, to the end of reducing the dimensionality of hyper-surface of solutions, the principal component analysis (PCA) has been applied allowing to identifying a set of linearly uncorrelated variables, which are responsible for the majority of dataset variability. These variable are the geometric features that have a greater effect on rotor performance.

Finally, the application of Projection to Latent Structures (PLS) allows to link this geometric parameters to Balje dimensionless parameters (i.e., specific speed and diameter) improving the amount of design information available using Balje chart.

5.2 Methodology

5.2.1 Dataset: Axial flow fan population

An enhanced version of Axlab which implements a new deviation model based on artificial neural network (see Annex A) has been used to explore the design space and derive a hypersurface of solutions of axial flow fans performance, directly related to geometrical and operational fan parameters. Diameters of tested fan population ranged between 0.2m to 2.5m. Each individual has hub-to-tip-ratio (HR) values between 0.25 and 0.75, while solidity values ranging between the extreme values used in the DOE to train the meta-model (see Annex A). Several chord distributions were considered (constant, variable) to test a wide range of blade aspect ratio values. Furthermore, the individuals are characterized by different pitch distributions selected according to free, mixed and forced vortex models allowing to test the blade in different loading conditions. In addition, we tested the individual at different pitch dispositions obtaining an envelope of fan characteristic curves to investigate whether the best efficiency operation occur at the blade angle different to the design value. The geometric and operative ranges of the fan tested during the design space exploration are reported in Table 5.1. In the table, Twist identifies the difference

Shroud diameter	0.2 - 2.5 m
Hub-to-tip ratio	0.25 - 0.75
Number of blades	4 - 16
Hub pitch disposition	15 - 60 deg
Mid-span solidity	0.25 - 1.15
Twist	5 - 45 deg
RPM	600 - 3600

TABLE 5.1: Geometric and operative ranges of exploited fans

between $Pitch_{hub}$ and $Pitch_{tip}$. Both the variable and the constant distribution of the NACA 4-digit airfoil camber have been chosen for the blade stacking line of the tested fans. The correct number of blades was selected according to the desired solidity, while the fan rotational speed was fixed observing the constrain imposed to tip speed Utip. We set $U_{tip} \leq 150$ m/s to allows adopting the incompressible flow assumption. Individuals were tested between 0.08 and 0.45 Mach number values. The blade tip clearance has been set as one percent of blade tip radius value for all tested individuals. The total number of performed simulations was equal to N=7313. AxLab requires less than two minutes to compute each simulation returning the fan performance curves and span-wise distributions of velocity and flow angles for the peak pressure flow rate (Q_{pp}) and peak efficiency flow rate (Q_{pe}).

Figure 5.1 shows the peak efficiency operating points of the tested population. We can see how fan designs tend to be disposed along the diagonal line following the natural tendency to associate high pressure with high flow rate. In Figure 5.2 the same population is shown on the specific speed (Ns) and diameter (Ds) plane, according to Balje diagram.

5.2.2 Big Data Analysis

The population inside the dataset was analysed by means of Principal Components Analysis (PCA) and Projection to Latent Structure (PLS) methods.

PCA analysis is applied to the main representative geometric and operative parameters of fan population: fan tip diameter (Dt), mid-span solidity, blade number (z), aspect ratio ($AR = h/l$, where l is the mid-span chord length and h in the blade height), hub-to-tip ratio ($HR = r_{hub}/r_{tip}$), twist ($\gamma_{hub} - \gamma_{tip}$), blade rotational speed (RPM), volume flow rate at peak efficiency (Q_{pe}), pressure ratio at peak efficiency (P_{pe}). During the analysis, we considered up to the third principal component, accounting for the 78% of data-set variability as shown in the elbow chart.

Following step of analysis required the use of PLS. The Balje specific speed (Ns) and diameter (Ds) are considered output variables of PLS process, while information related to the blade geometry, not provided by the use of Balje chart, in particular for mixed and forced vortex blading, as the hub to tip ratio, the solidity or the blade height are considered input features for PLS analysis aimed at finding the relation between these geometric variables and the cited dimensionless parameters enriching the amount of information available from Balje chart. Actually, the main responsible variables for flow rate (Q) and pressure rise (P) values are Dt and RPM, however, also the blade geometry exerts an influence on Q and P. Thus, according to the definition of Ns and Ds (see chapter 2),

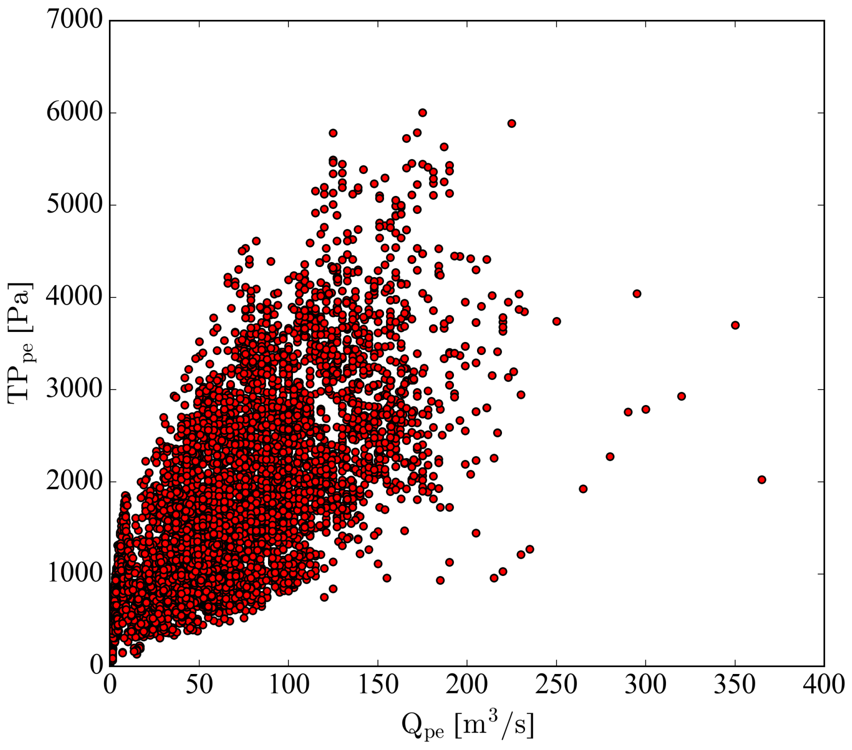


FIGURE 5.1: Tested fan population on P-Q plane of peak efficiency

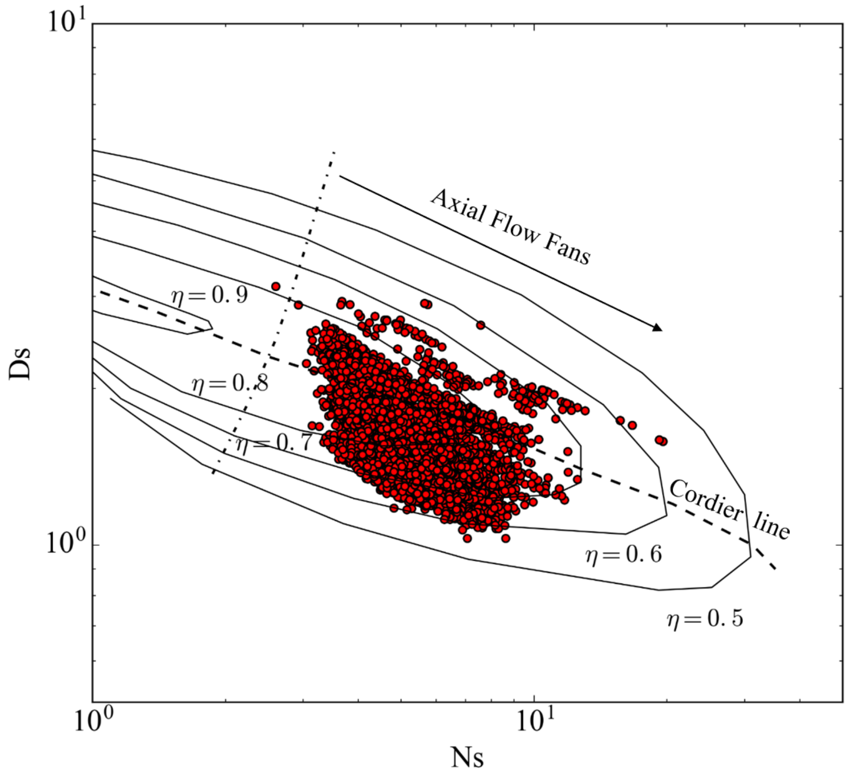


FIGURE 5.2: Representation of tested fan population on Balje chart

specific speed and diameter are affected by size (Dt) and operation (RPM) of the fan in an explicit manner, but also by geometrical effects via Q and P parameters.

5.3 Results

5.3.1 PCA Analysis

Figures 5.3 - 5.4, 5.5 - 5.6 and 5.7 - 5.8 show three pairs of representations of the loadings (i.e., the p geometric and operation variables) contribution to the first and second, first and third, and second and third respectively.

Figure 5.3 shows that Dt, solidity, Z and RPM are the higher magnitude loadings for the first component. The rotational speed is the only positive loading, whereas the others are all negative. The inverse correlation between Dt and RPM, laying on the same direction with opposite orientation, is an expected behaviour related to the constraint applied to the rotational speed values of fans in the design space. This is an acknowledged information but is useful to assess the quality of this first approach. AR, defined as the ratio between h and chord length, and HR have significant loading on component 2, positive and negative respectively, blade twist shows appreciable loading magnitude as well, so this component seems the one characterizing the blade geometry. The vector diagram in Figure 5.4 clearly shows that HR is completely uncorrelated with the fan size and rotational speed being the direction of this parameter almost orthogonal to the direction of the other cited parameters.

The loading plot reported in Figure 5.6 shows the influence of original variable on first and third component. Twist has the largest positive loading on component 3, so this component is strictly related to blade loading level and in particular to the swirl velocity distribution defined by the selection of vortex model. In this work, the vortex flow around the span was selected according to the free, mixed and forced vortex assumptions obtaining different amount of twist between the root and tip section of each tested blade. Actually, the blades designed according to the free vortex theory present the largest amount of twist value for a specific blade height. This angle gradually decreases moving through mixed and forced vortex assumption.

The last loading plot reported (Figure 5.8) allows visualizing the influence of the original variables on residual variability not considered in the previous analysis, further showing the deep relationship between twist angle and the third principal component. This figure also shows that parameters influencing the blade geometry (i.e., AR, HR and twist) affect the blade loading level as well (i.e., the third component). It is also interesting to note that AR and HR are equally important for both the second (blade geometry) and the third component (blade loading), but their contribution is opposite for the second component (for given chord length, a high HR tarnishes the blade length pulled up by a high AR).

5.3.2 PLS Analysis

Table 5.2 reports the loading coefficients (w) exerted by the input and output variables on the PLS components. In particular, given its importance on the third PLS component, the blade height h has been considered in the statistical analysis as an additional input parameter.

To facilitate the data interpretation, input and output variables are differently coloured in the following bar graphs of the loading vectors.

Figure 5.9 shows the PLS loading vector on first component underlining the relationship between HR, solidity and z and their influence on Ns and Ds. The cited inputs result inversely proportional to Ns while they are directly proportional to Ds, suggesting that high values of these parameters determine increases in Ds while decrease the Ns value.

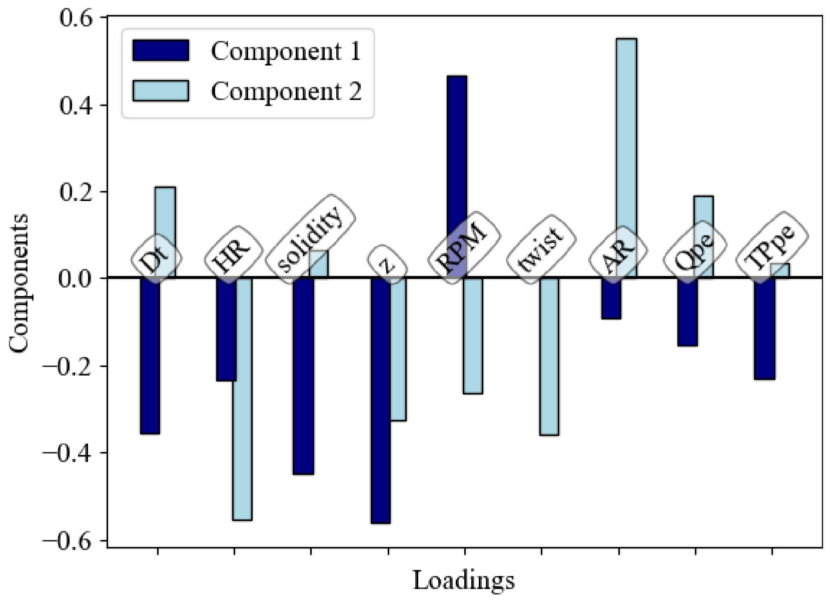


Figure 5.3: First and second component loading bar

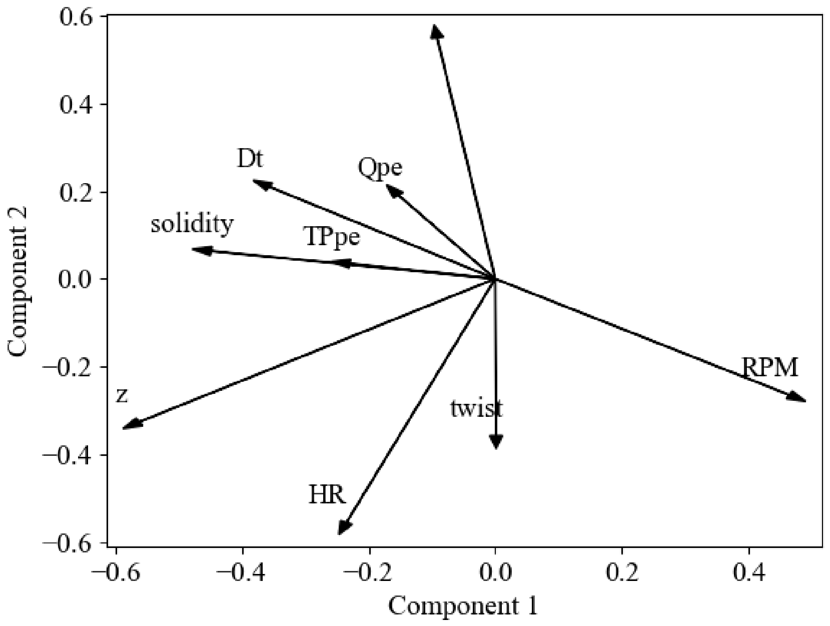


Figure 5.4: First and second component loading plot

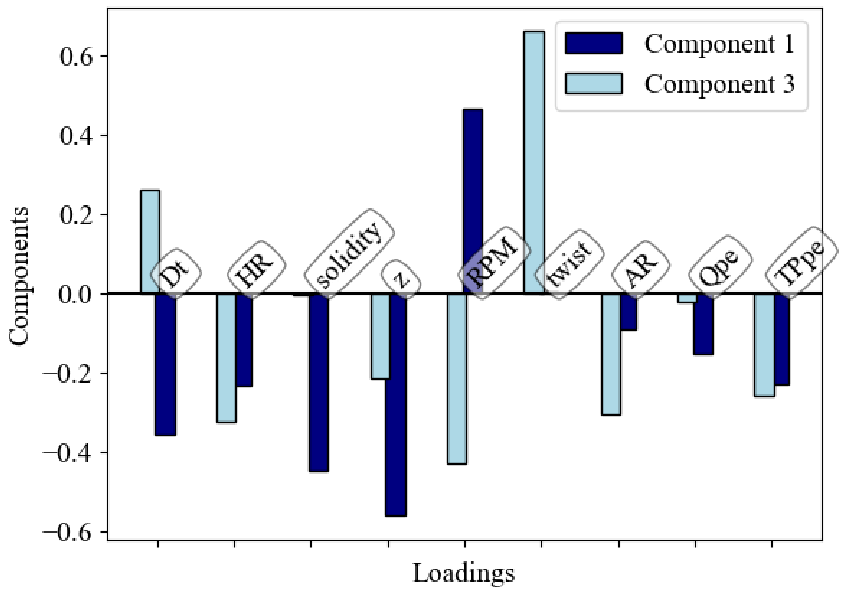


FIGURE 5.5: First and third component loading bar

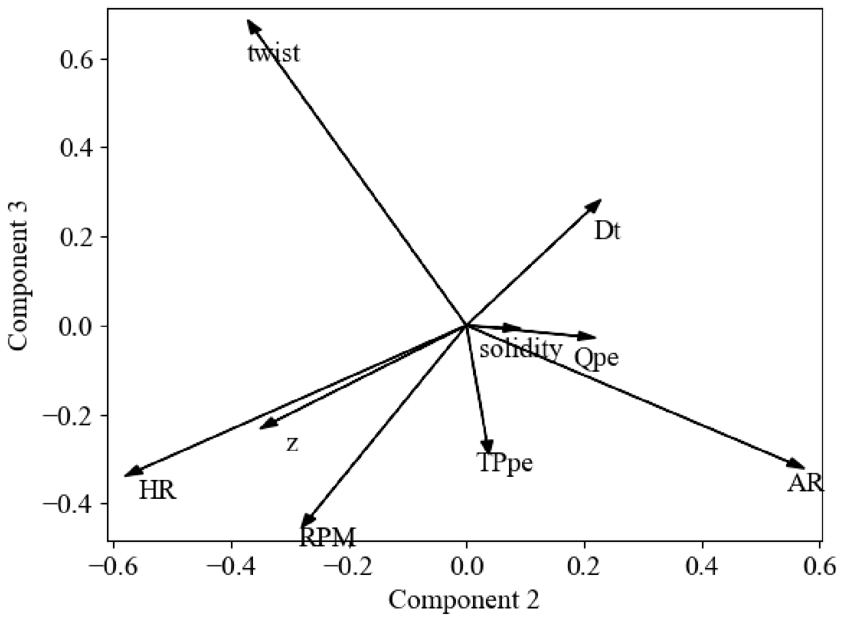


FIGURE 5.6: First and third component loading plot

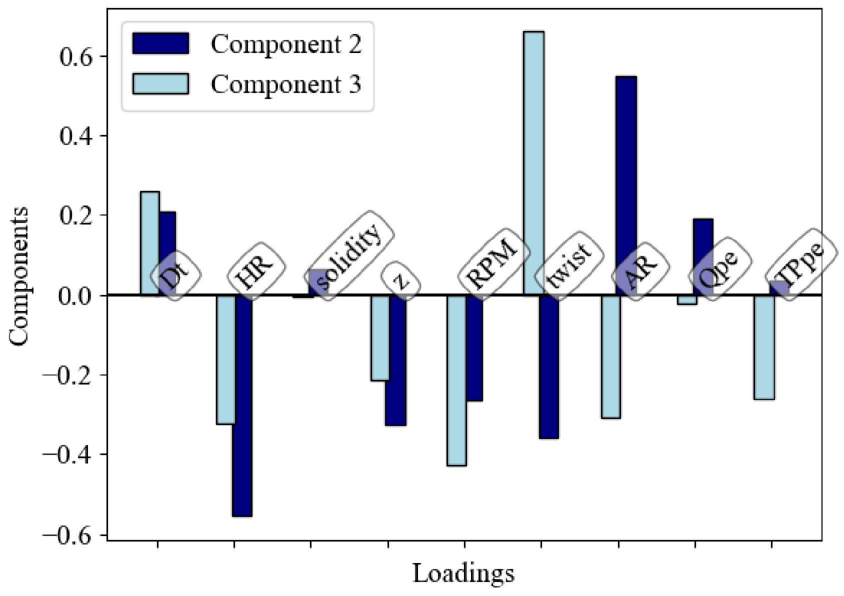


FIGURE 5.7: Second and third component loading bar

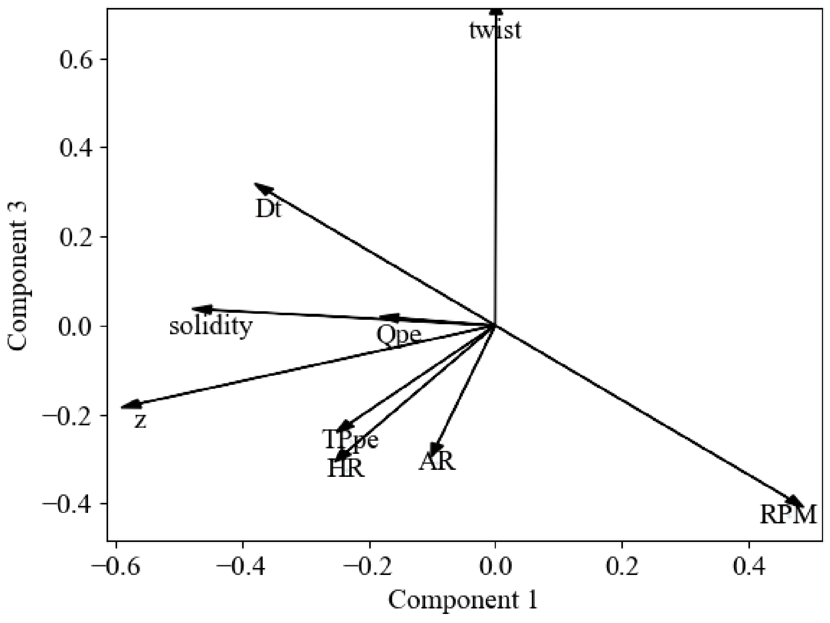


FIGURE 5.8: Second and third component loading plot

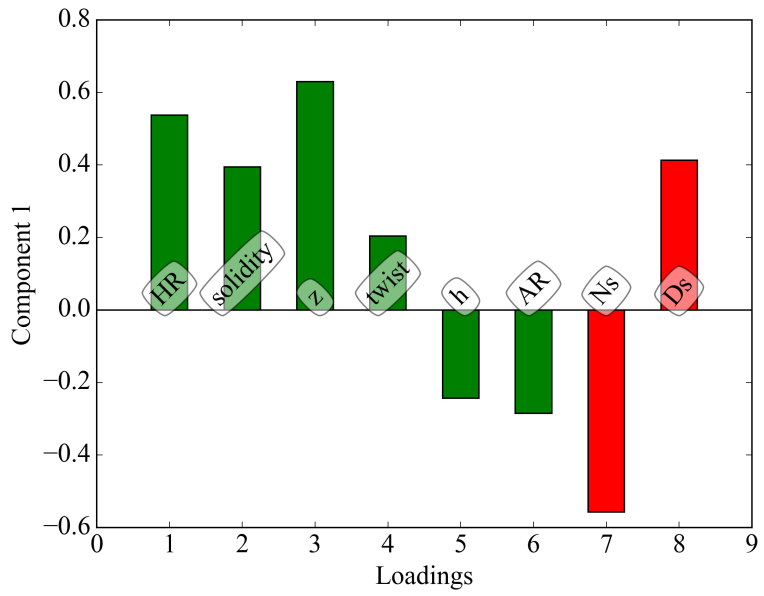


Figure 5.9: PLS loading vector on first component

In Figure 5.10 we report the loading vector on second component accounting for the remaining data set variability. Looking at the loading coefficients, we can see a significant relation of twist angle and aspect ratio on Ns and Ds. In particular, the twist angle results directly proportional to the dimensionless parameters, while the aspect ratio having an opposite trend results indirectly proportional As we can see in this last bar graph (Figure 5.11), the loadings of output variables is low suggesting to not consider further components.

	Component 1	**Component 2**	**Component 3**
w_{HR}	0.5373409	0.0630834	-0.418244
w_{σ}	0.3945273	-0.3643465	0.473071
w_{z}	0.6297442	-0.2317783	0.096296
w_{twist}	0.204328	0.60159	0.430652
w_{h}	-0.2433161	-0.3081109	0.709880
w_{AR}	-0.2845599	-0.6468832	0.060962
w_{Ns}	-0.5575367	0.2214975	-0.007182
w_{Ds}	0.4129333	0.1372413	0.117609

Table 5.2: PLS loading coefficient

5.3.3 Multidimensional Balje Charts

The relationship between the fan geometric variables and the specific speed and diameter, reported in the PLS analysis, are here used to define three composed parameters (CP) successively used to better drive the exploration of Balje chart. In particular, the PLS loading coefficients are used to develop linear combinations of the geometric parameters having the major effect on Ns and Ds: each geometric parameter in the linear combination is multiplied by the corresponding loading coefficient (see Table 5.2). Hence, the geometric combination is normalized dividing by the absolute value of the sum of the loadings. The

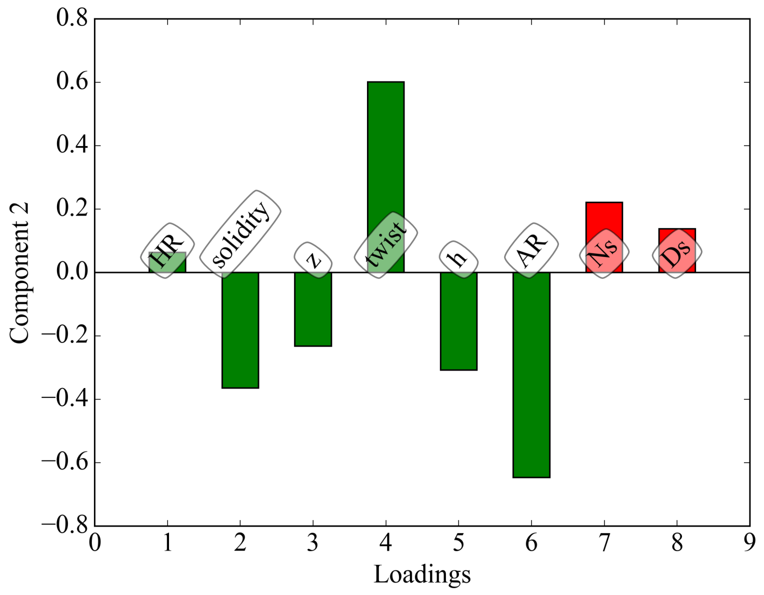


FIGURE 5.10: PLS loading vector on second component

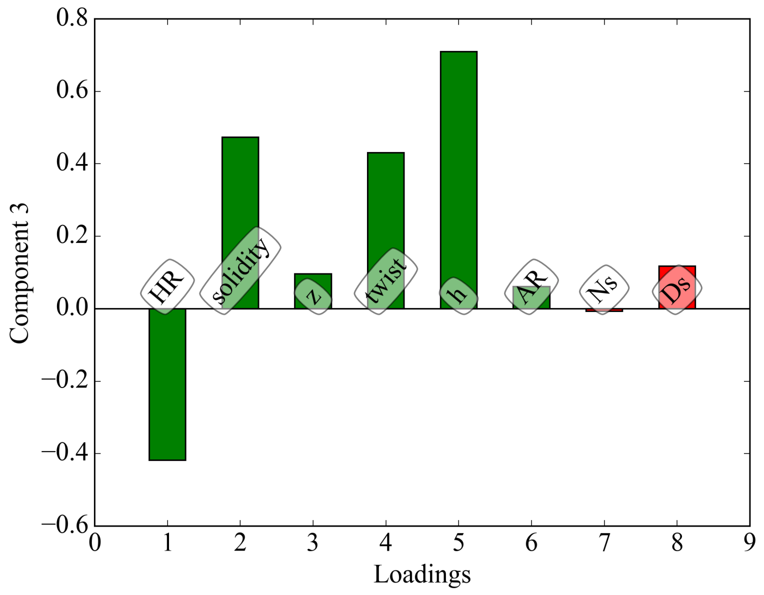


FIGURE 5.11: PLS loading vector on third component

composed parameters allow the creation of two-dimensional contour graphs on the Ns-Ds plane. In particular, we decided to group HR and sigma in the same chart because they are measure of two different "solidities": HR is a blade solidity connected with the amount of flow rate evolving through the fan while σ is a cascade solidity that take into account number of blade and the chord distribution along the span. Furthermore, the pair HR and z was selected according to their loading values on the first PLS component (the dependence between solidity and z suggested to avoid grouping this parameter pair). Finally, grouping twist and AR, the most influencing parameters of second component, we defined a measure of the blade global load. In particular, the twist angle is a measure of the blade loading, related to the vortex model adopted. Furthermore, the aspect ratio takes into account the amount of blade surface available to produce the fluid deflection according to the cited blade loading. Figure 5.12 shows the multidimensional extension of Balje chart using the linear combination of HR and solidity (CP1). We can see how the CP1 values decrease moving through higher Ns values while high values of the linear combination are confined between Ns=4 and Ns=5. In a preliminary design phase, after the selection of the desired

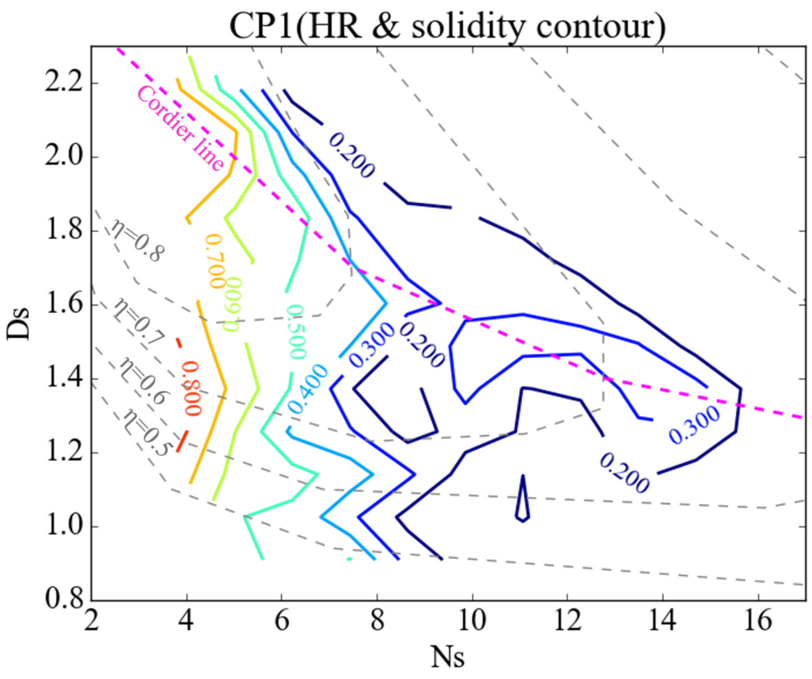


FIGURE 5.12: Multidimensional Balje chart with HR and solidity contour

Ns and Ds, this chart allows to identify the value of HR and solidity combinations. The CP1 expression used to identify one of the two geometric variables knowing the value of the other is reported in the following Equation 5.1:

$$CP_1 = \frac{w_{HR,1} \cdot HR + w_{\sigma,1} \cdot \sigma}{abs(w_{HR,1} + w_{\sigma,1})} \tag{5.1}$$

where $w_{HR,1}$ and $w_{\sigma,1}$ are the PLS loading coefficients exerted on the first PLS component by HR and solidity respectively.

Moving to Figure 5.13, the extension of Balje chart using the contour of linear combination of HR and z (CP2) is reported. The contour lines are characterized by a vertical trend reaching lower values moving through higher Ns. Following Equation 5.2 shows the CP2 definition:

$$CP_2 = \frac{w_{HR,1} \cdot HR + w_{z,1} \cdot z}{abs(w_{HR,1} + w_{z,1})} \tag{5.2}$$

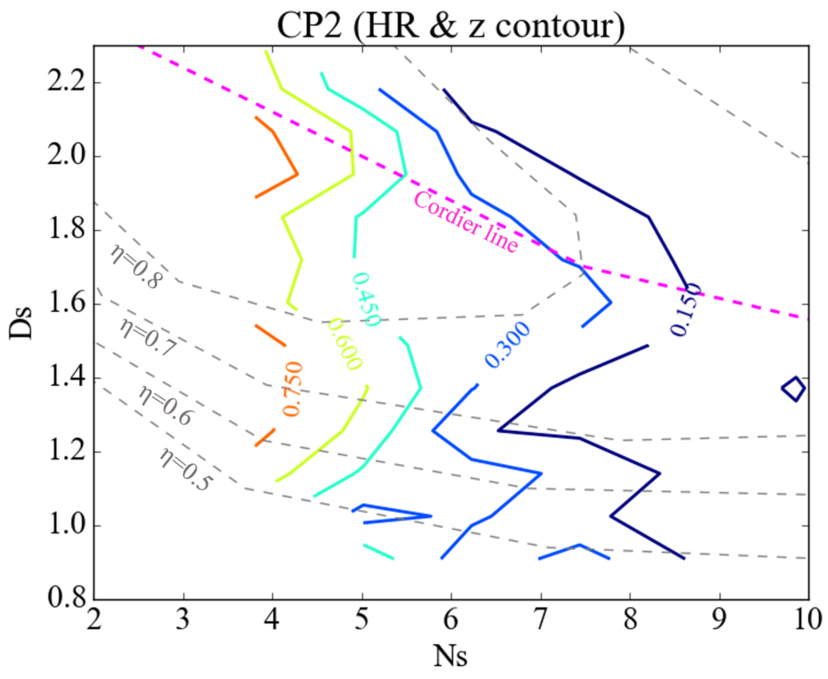


FIGURE 5.13: Multidimensional Balje chart with HR and z contour

as seen in the previous case, $w_{HR,1}$ is the loading exerted on the first PLS component by HR while $w_{z,1}$ is the loading of z on the same component. The last multidimensional

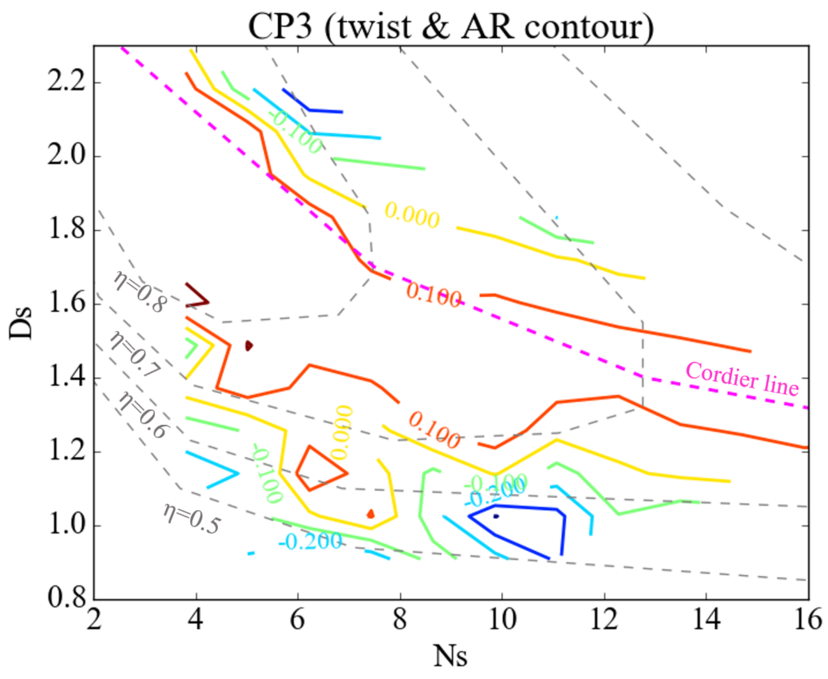


FIGURE 5.14: Multidimensional Balje chart with twist and AR contour

Balje chart extension is obtained enriching the Ns-Ds plot with the contour of the linear combination of twist angle and aspect ratio (Figure 5.14). Following the CP3 expression

(Equation 5.3):

$$CP_3 = \frac{w_{twist,1} \cdot twist + w_{AR,1} \cdot AR}{abs(w_{twist,1} + w_{AR,1})} \tag{5.3}$$

where $w_{twist,1}$ and $w_{AR,1}$ are the PLS loading coefficients exerted on the first PLS component by HR and solidity respectively. CP3 is a measure of the blade global load. An interesting behaviour regards the disposition of CP3contour lines that are parallel to the Cordier line. In particular, the Cordier line lays in the range of highest CP3 suggesting taking that level of twist and AR combination to obtain the highest reachable efficiency according to the desires Ns and Ds. Because of the inversely correlation between twist and AR loading negative contour lines values are including in the contour plot. Using the previous multidimensional Balje chart in a combined approach the designer could identify the values of HR, solidity, blade number, twist angle and AR starting from the selection of the desired Ns and Ds enriching the amount of data available during the preliminary design process.

5.4 Conclusions

The axial fans design space was explored using an axis-symmetric solver to derive fan performance and the data collected have been used to build a hypersurface of 7313 performance solutions directly related to geometrical and operational fan parameters. The hypersurface was then explored using multivariate statistical techniques. The work has been carried out: i) to confirm some literature findings on the relations between fan performance and the most representative geometric parameters; ii) to further extend the literature findings enriching the amount of information available using the Balje chart during the fan preliminary design phase.

The results of the several statistic techniques used in this work allowed to identify the variables responsible for most of data set variability. Tip diameter, solidity and RPM resulted in a deep influence on the first component, that we can see as measure of fan size and operative conditions. The aspect ratio and the blade high resulted intensely correlated with the second component, which measures the geometry of fan blade. Instead, the third component through its dependence on twist appeared associated with the blade load coefficient and the flow vortex model.

A further analysis applied to the parameters mostly affecting the data set variability permitted to weight the relationships existing between the main fan geometrical parameters and the specific speed and diameter.

A close connection between hub ratio, solidity, twist and the Balje dimensionless parameters has be found and used to added further dimensions to the Balje chart enriching the information available during the preliminary design with new relations accounting the blade geometry.

5.5 Annex A

A dataset with high fidelity data is needed to explore the design space of axial fans and derive a multidimensional Balje chart. For this work, the data were generated using an axi-symmetric code, and therefore the fidelity of the dataset is of paramount importance for the final results of the analysis. In particular, many design parameters concur to define the blade solidity and to increase the fidelity of the dataset we needed to implement an accurate method to derive lift and drag performance of a linear cascade at different Reynolds numbers, angle of attack and angle of cascade. Therefore, a preliminary part of this work was the development of a meta-model for lift and drag coefficient prediction of NACA 4-digit profiles in cascade configuration, as the AxLab axi-symmetric code requires lift and drag coefficients to estimate the flow exit angle

The derivation of this meta-model was carried out with a design of experiments that drove a series of CFD computations and then required to train an artificial neural network to predict lift and drag performance of the cascade. The complete procedure is described in the following.

5.5.1 Design space

A cascade aerodynamic performance meta-model for the unidirectional NACA 4-digit series commonly used in industrial fan applications has been developed. Due to the minor influence of the airfoil thickness on the blade aerodynamic performance, in this work we decided to fix the profile thickness at 12% of chord length (third and fourth digits of the NACA code) to limit the number of trials needed for meta-model development. Only the first and second digits of the NACA profile have been modified. To explore a significant range of camber angles, the first digit levels ranged from 1 to 6 ,while the second digits spaces from 2 to 6 (i.e., all the profiles between the NACA1212 and the NACA6612 were included in our design space). These airfoils feature values of camber angle and camber line shapes commonly used in the axial fan applications. Different cascade setups are tested to take into account the blade-to-blade flow interaction by changing the cascade pitch angle (γ), solidity ($\sigma = l/t$) and the Angle of Cascade (AoC). The computations are performed at different Re number values.

5.5.2 Data sampling

A series of numerical simulations were needed to derive the deflection meta-model. Using a Design Of Experiments (DOE), the region of interest of the factors (see first column in Table 5.3) is covered optimally by the chosen experimental setting maximizing the accuracy of the information obtained by experiments. DOE also allows to investigate factor interactions, being much more rigorous than traditional methods of experimentation such as one-factor-at-a-time and expert trial-and-error [93]. Among the several DOE techniques available in the literature, full factorial design allows testing all possible combinations of factors. However, the number of trials required by this approach geometrically increases with the number of factors. Conversely, the Central Composite Design (CCD) is a fractional factorial design (with centre points) which adds other samples (the star points) to estimate the curvature of the design space [94] with a limited number of required trials. Among the three CCD available, in this work we used a rotatable Central Composite Inscribed (CCI) design of experiment with axial points located at factors levels -1 and 1 and factorial points fixed at a distance $1/\nu$ from the center points. The rotatability is assured by setting $\nu = (2^h)^{0.25}$, where ν identifies the distance from design space center to a star point [95]. The factor ranges and the corresponding tested levels are reported in Table 5.3). Being the design space described by six factors, the total number of computations required was equal

Factor Name	**-1**	$-1/\nu$	**0**	$1/\nu$	**1**
First digit	1	2	4	5	6
Second digit	2	3	4	5	6
Pitch	15	21.68	37.5	53.31	60
Solidity	0.15	0.32	0.75	1.17	1.35
AoC	-4	1.02	6	13.02	16
Re	$2x10^5$	$4.67x10^5$	$1.1x10^6$	$1.73x10^6$	$2x10^6$

TABLE 5.3: Tested levels of design factors

to $N = 2^h + 2h + 1 = 77$. These trials are sufficient to derive the meta-model. However, to obtain a better description of the cascade performance especially in the near stall region and at the zero lift angle of cascade, several trials at additional AoC levels were added to the original DOE. Accordingly, the total number of two-dimensional simulations performed was increased up to $N = 140$. The additional points allow to better capture the curvature of the cascade polar curves.

5.5.3 Numerical model

The two-dimensional computations required to develop the meta-model were carried out using the C++ open-source code OpenFOAM v.18.06 [96]. In this work, the simpleFoam solver was used to carry out steady state computations of incompressible flows. The Spalart Almaras [97] turbulence model was applied. Velocity and $\tilde{\nu}$ equations were solved with smoothsolver, while the Generalized Algebraic Multi grid solver was used for the pressure equation. The computational domains used involve a single blade-to-blade passage with

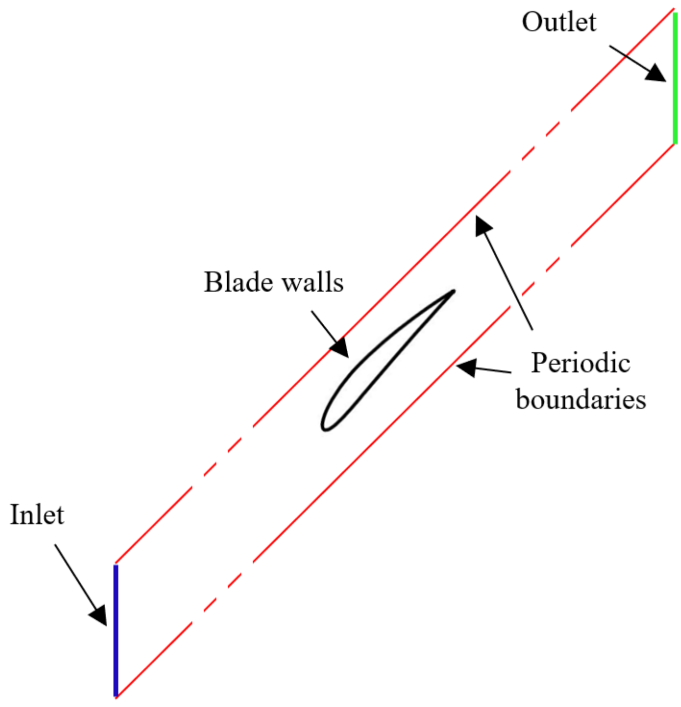


FIGURE 5.15: Computational domain

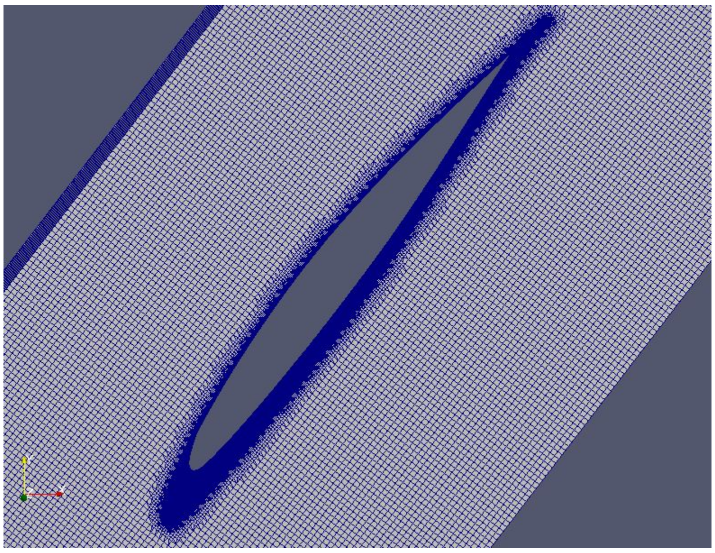


Figure 5.16: Detail of computational domain generated using snappy-HexMesh

periodic boundary conditions applied in pitch wise direction. The OpenFOAM snappy-HexMesh [96] mesh generator was used to systematically generate the hex-dominant grids (the computational domain is reported in Figure 5.15 and Figure 5.16). We used computational domains characterized by the same geometric feature already used and validated in [98].

Each grid extends 30 chords in the up-stream direction and 60 chords in the down-stream direction. The mean grid dimension is 340k cells, small variations were observed depending on the solidity of the case. A wall spacing of 0.02mm assures a y+≈ 1 for all the grids. The range of mesh quality indicators are reported in Table 5.4. For all simulations,

	Min	**Max**	**Average**
Area ratio	1	1.6	1.08
Aspect ratio	1	109	12
Skewness	0	0.58	0.04
Min incl. angle	25	90	83
$y+$	0.02	0.79	0.3

Table 5.4: Mesh quality indicator

we imposed a constant velocity vector at the inlet, which is defined according to the angle of cascade. The desired Re number is obtained adjusting the ν values, while inlet value is fixed to $\tilde{\nu} = 3\nu$. Velocity non-slip conditions are imposed over the walls [99]. Each of 140 simulations required approximately two hours. The total computational time needed was about 11 days.

5.5.4 Artificial Neural Network

The meta-model developed was based on an artificial neural network with multilayer perceptron (MLP) structure. Following a series of tests carried out to identify the best MLP configuration to ensure accuracy limiting the computational costs, the architecture with two hidden layers and 30 neurons was selected. The neural network includes a sigmoid activation function for the hidden layers and a linear function on the output layer (Figure 5.17).

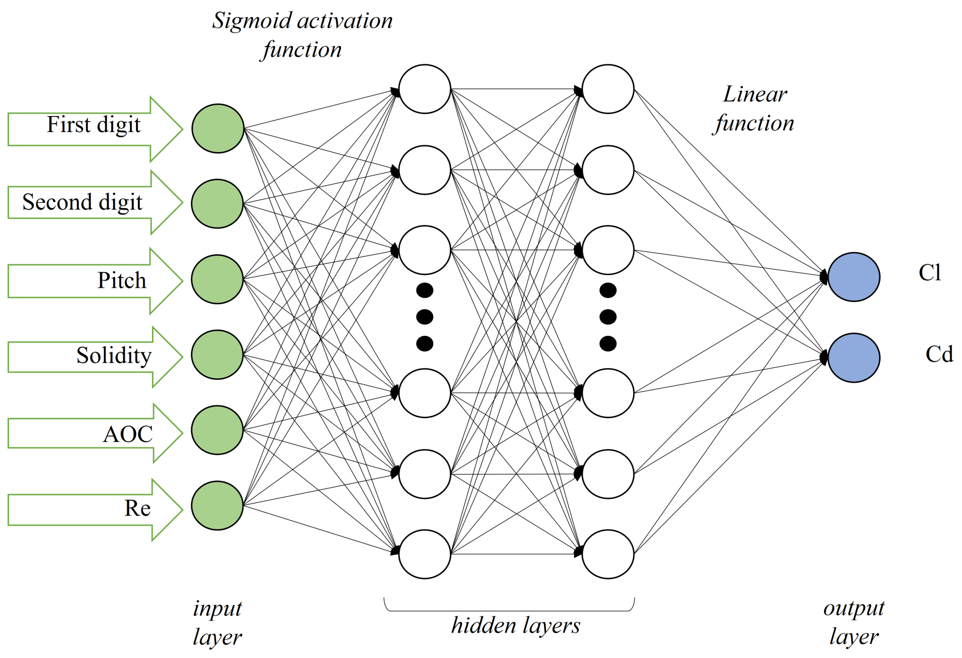


FIGURE 5.17: Multi layer perceptron structure

A Bayesian optimization algorithm was used to train the neural network, randomly initializing weights with Gaussian function and biases using a linear function. We set the learning rate to 0.2e-2 and the training epochs to 3000. The sampling dataset was split into a training dataset containing 80% of the total trials and a test dataset with the 20% remaining. To evaluate the neural network accuracy and avoid over-fitting the MSE for test and training data was computed along the training epoch (Figure 5.18). The magnitude of MSE values obtained in the training phase is practically similar between training and test data subsets, showing a good balance between over-fitting and under-fitting. The model is sufficiently accurate to explain the variance of data set furnishing a good prediction and at the same time presents an adequate flexibility level useful to avoid data over-fitting.

5.5.5 Improved solver validation

The accuracy of the solver, improved by the introduction of deflection meta-model, has been assessed comparing the performance characteristics predicted by the axi-symmetric code with the experimental data of a single stage axial flow fan for CSP air-cooled condensers. The experimental setup respects the international standard ISO5801:2007. The

Casing diameter	1542 mm	
Hub-to-Tip ratio	0.3	
Minimum Tip Clearance	6.2 mm	
Number of blades	8	
	Hub	***Tip***
Solidity	1.03	0.29
Chord	180 mm	174 mm
Re	6×10^5	1.5×10^6

TABLE 5.5: CSP fan geometry

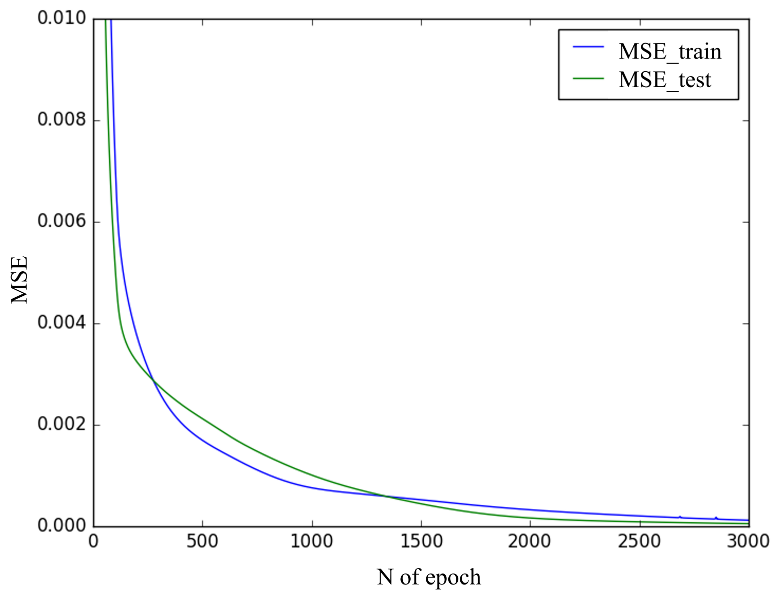


Figure 5.18: Meta-model cost history in terms of MSE distribution

geometric features of selected fan (Table 5.5) are completely included in the sampled design space used during the training of the meta-model artificial neural network based. We can see an appreciable improvement of the AxLab meta-model driven prediction of fan total pressure and power curves (Figure 5.19, Figure 5.20). Using the enhanced version of the solver, the gap between predicted total pressure rise and power and experimental data significantly decreases, especially in the fan stable operation range. The standard version of AxLab, neglecting the blade-to-blade interaction, overestimates the blade deflection capability resulting in high total pressure rise and high power prediction. The overestimation in pressure rise prediction decreases from the 15% of standard AxLab prevision to less than 1.5% of the AxLab meta-model driven. A similar result has been observed in the fan power prediction.

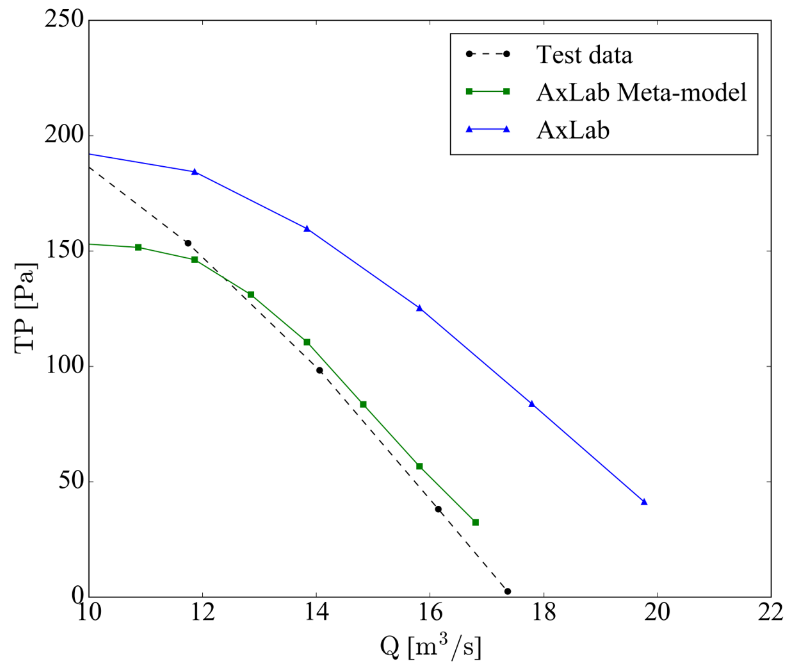


FIGURE 5.19: Total pressure curves predicted using AxLab against experimental data

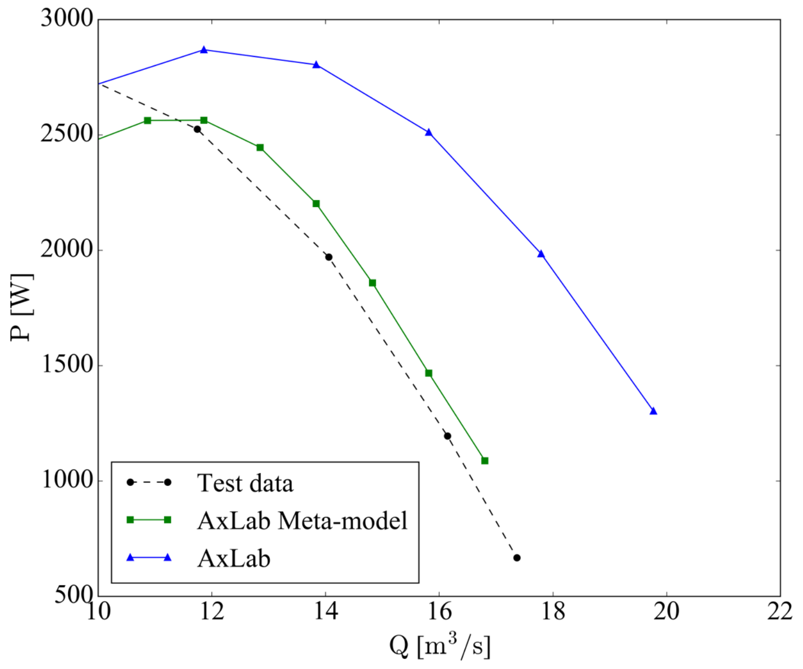


FIGURE 5.20: Power characteristics predicted using AxLab against experimental data

Chapter 6

Surrogate Model Based Optimization of Truly Reversible Fans

6.1 Introduction

The aim of this chapter is assessing how exploiting surrogate-based techniques in optimization problems concerning the aerofoils aerodynamic. We will address the following questions: how meta-model techniques affect results of the multi-objective optimization problem, and how these meta-models should be exploited in an optimization test-bed. In particular the object of this optimization strategies are truly reversible axial fans.

Reversible single-stage axial fans are largely employed in tunnel and metro ventilation systems, to supply and extract air from the tunnels. State-of-the-art solutions include the use of dedicated fans for air supply and extraction, but in general the use of reversible fan rotors, characterized by truly reversible blade sections, has been established as the most convenient [100], [101]. Traditionally, truly reversible aerofoils have been generated by flipping the pressure side of a non-cambered aerofoil and joining the suction side trailing edge with the pressure side leading edge and vice versa (see Figure 6.1). Therefore, the aerofoil has periodic aerodynamic features every 180 degrees of angle of attack. However, the unnecessary thickness in the trailing edge, coming from this approach, reduces the performance when compared to standard aerofoils used for turbomachinery blades. As a global effect, the average aerodynamic efficiencies of reversible fans reach just up to 95% of the correspondent non-reversible geometry [102]. Since 2012, the new trend in mandating

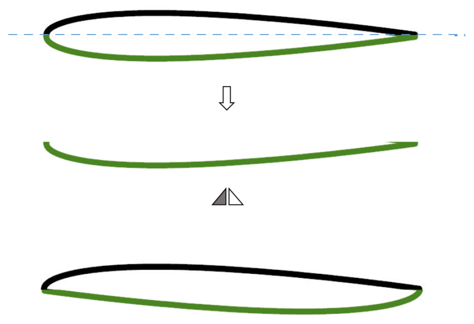


FIGURE 6.1: Reversible aerofoil generation

minimum efficiency grades asked fan designers for stricter efficiency constraints during the design of a new reversible fan. In the past, reversible airfoil solutions relied mainly on two-dimensional cascades wind tunnel data [103] or fully three-dimensional annular cascades data [104]. Nevertheless, experimental reversible cascade data are seldom available to industrial fan designers.

Within this context, we propose a study on the multi objective optimization problem (MOOP) to obtain a set of optimized aerofoil shapes for the use in reversible fan blading. In particular, we investigate the possibility to speed up the MOOP based on NSGA-II algorithm by means of surrogate models (SMs). In this view, meta-models created by a least square interpolation surface and artificial neural network were used to replace the aerodynamic performance predicted by XFoil. Even if the use of boundary element method (BEM) can lead to non-optimal solutions, it is still a good replacement (mostly customary in design phase) of an expensive tool such as CFD to perform a preliminary study on this topic and to define guidelines [98]. Each meta-model was used in two different optimization strategies namely: (i), a simple-level characterized by a no evolution control (NEC) approach; (ii), an indirect fitness replacement (IFR) bi-level. To have the maximum control on each step of the MOOP solution, the entire optimization framework and the meta-models development were in-house coded in Python [105]. In conclusion, differences between the use of different meta-models and different optimization strategies are discussed by means of quality metrics typical of optimization schemes.

6.2 Multi Objective Optimization with Evolutionary Algorithms

Whether Multiobjetive Optimization with Evolutionary Algorithms (MOEA) is assisted or not by surrogate models, there are main aspects common to both approaches. To mention but a few: the choice of the objective functions, the geometry (aerofoil) parametrization and the optimization algorithm.

6.2.1 Objective Functions

The selected objective functions for the multi-objective optimization problem (MOOP) were the aerodynamic efficiency (ϵ) and the stall margin (α), defined as:

$$\epsilon = \frac{C_{LS}}{C_D}, \qquad \alpha = \frac{AoA(max(C_L)) - AoA(C_{LS})}{AoA(max(C_L))} \tag{6.1}$$

where C_L is the lift coefficient, C_D is the drag coefficient and *AoA(max(*C_L*)* and *AoA(*C_{LS}*)* are the angle of attack. According to the given definition of stall margin α, each aerofoil polar was computed between 0 deg and the angle of attack where the C_L reaches its maximum.

Aerodynamic efficiency was selected in this study to represent the key performance indicator of profile losses in relation to the aerodynamic loading of the blade section. Stall margin provides a measure of how much the design point is far from stall. When optimized geometries data are used in a quasi 3D axi-symmetric code, for either design or performance prediction of an axial fan, as in the work [98] or [25], stall margin is also a measure of solution reliability. The α objective function has the additional purpose to prevent tendency that optimized elements have to move their maximum aerodynamic efficiency in proximity of *AoA(max(*C_L*))*. Aerodynamic efficiency, lift coefficient and therefore stall margin are not only functions of aerofoil geometry (g), where g is a vector that univocally defines the aerofoil geometry, but also of Reynolds number (*Re*):

$$\epsilon = \epsilon(Re, C_{LS}, g), \qquad \alpha = \alpha(Re, C_{LS}, g) \tag{6.2}$$

The objective is to get a set of optimized geometries under specified Re and C_{LS} that can be used to replace the aerofoils sections of an existing blade geometry. Such new set of geometries g' is considered optimized under specific Re and C_{LS} if aerodynamic efficiency ϵ' and/or stall margin α' are higher than the $\epsilon$and α of the current set:

$$\begin{gathered} \epsilon = \epsilon(Re, C_{LS}, g\prime) = \epsilon', \quad \alpha = \alpha(Re, C_{LS}, g') = \alpha' \quad and \\ C_L(Re, AoA', g') = C_{LS'} \end{gathered} \tag{6.3}$$

Such aerofoils collection can be used to modify an existing fan blade to obtain better fan efficiency or more stable working conditions at a given design point. In fact when velocity diagrams, aerodynamic loading ($\mathrm{C_{L,i}}$), Reynolds number (Re_i) and blade solidity ($sigma_i$) are prescribed at each blade radius, it is possible to determine the aerofoil stagger (γ_i)(Figure 6.2) that results in the blade working at optimized conditions ϵ' and α', $\gamma_i = \beta_{m,i} - AoA'_i$, where "i" identifies the generic blade section along the span. The restaggered blade with optimized aerofoils will result in a deflection ($\theta = \beta_1 - \beta_2$) close to the design deflection.

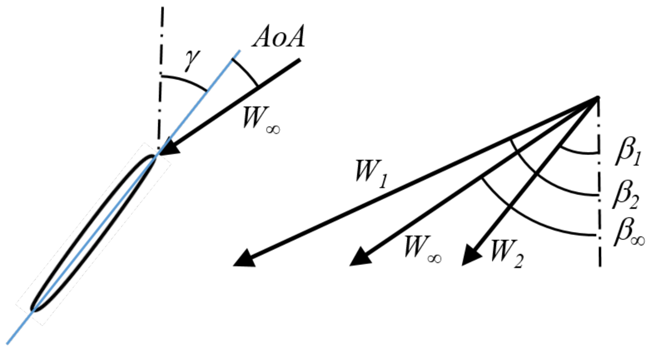


FIGURE 6.2: Definition of blade angle of attack (AoA), stagger angle (γ) and relative velocities

6.2.2 Aerofoil parametrization

As the present study concerns truly reversible aerofoils, the parametrization of only one surface of the profile defines univocally the aerofoil geometry. Selection of the parametrization scheme among all available methods represents a crucial point, because it strongly influences the whole optimization process, as described by Wu [106], Kulfan [107] and Samareh [108]. For optimization purposes the most important features of a parametrization scheme are: (i), to provide consistent geometry changes; and (ii), to produce an effective and compact set of design variables [108]. These two features are fundamental during the optimization loop because they simplify generation and simulation of geometries, and avoid data overfitting in the meta-model. Two parametrization schemes were tested: Sobieczky [109] and the B-Splines parametrization [108]. These schemes were chosen for their simplicity and the limited amount of design variables required to reproduce a set of given aerofoil geometries. B-Splines parametrization was selected according to its capability to reproduce the suction side of a set of 40 reversible aerofoils selected between either those most commonly used in ventilation industry, or those generated by a simple rearrangement of the suction side of several generic non-cambered aerofoils. A curve fitting algorithm, developed by Schneider [110], finds values of the parametrization coefficients for the given profiles that allows to obtain the minimum deviation between a selected reversible aerofoil and the corresponding parametrization.

According to the Nelder-Mead search algorithm [111], the 6^{th} degree B-Spline parametrization guarantees a good balance among aerofoil geometrical features and the number of design parameters. B-Spline parametrization uses the coordinates of 6 points: two points define the beginning and the end of the aerofoil suction side and are forced to have coordinates (0,0) and (1,0). A second set of points indicates the direction of the surface tangents at the leading edge and at the trailing edge, respectively. For those points, x was again forced to 0 and 1 respectively, while y coordinates are the first two degrees of freedom (Y_{a1} and Y_{a4}). Third couple of points are used to control the suction surface between leading and trailing edge adding other 4 degrees of freedom ($X_{a2}, Y_{a2} and X_{a3}, Y_{a3}$). Figure 6.3 shows the set of control points and relative parametrization (dashed line) after the curve fit to reproduce the given geometry (solid line). The error observed applying the B-Spline

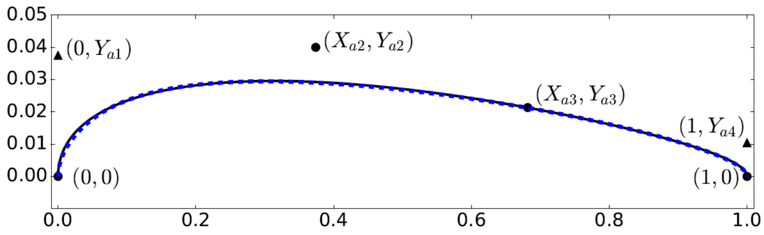


FIGURE 6.3: Control points used to reproduce the given geometry (solid line) with a 6th degree B-Spline parametrization (dashed line)

parametrization is less than 10^{-3} for each of the initial set of 40 reversible aerofoils. It is computed as the distance between the suction side (solid line in Figure 6.3) and its parametrization (dashed line in Figure 6.3). Table 6.1 reports the error range:

min error	4.29×10^{-5}
max error	7.26×10^{-4}
mean error	2.91×10^{-4}

TABLE 6.1: B-Spline parametrization error range

6.2.3 Optimization algorithm

The MOOP has been solved using the non-dominated sorting genetic algorithm (NSGA-II) proposed by Deb. Starting from the parent population P_t, taken as an initial set of $N = 40$ individuals (geometries), the algorithm generates an offspring population Q_t of the same size N. Then the two populations are combined together to form an R_t population of size $2N$. The offspring Q_t is generated using the following criterion: four elements are generated using crossover, mutation and breed interpolation from first two most promising elements in P_t. Among them, first two are random mutations, a third element is generated by scattered crossover and the genes of the last element are averaged between parent's genes. The remaining 36 elements of Q_t population are obtained by scattered crossover or mutation of randomly selected elements in Pt. The Deb non-dominated sorting approach is used to pick individuals from the last non-dominated front. An advantage of this approach is that solutions compete each other also in terms of how dense they are in the fitness space. Thus, diversity is explicitly considered when forming the offspring population. Furthermore, elitism is ensured by non-dominated sorting of both parents and offspring and inclusion of non-dominated fronts in the new set of elements. NSGA-II has demonstrated to be able to find much better spread of solutions and better convergence near the true Pareto-optimal front compared to the Pareto achieved with different evolutionary algorithms [43].

6.2.4 Test matrix

We decided to select a series of operating conditions on the envelope of reversible fans for tunnel and metro applications [112]. A matrix of 25 optimization cases was defined by means of the selection of 5 values of Re, and five values of C_{LS} reported in Table 6.2. Each optimization starts from the same initial population (P_t) of $N = 40$ reversible aerofoils described above.

Re	300000	675000	1050000	1425000	1800000
C_{LS}	0.1	0.3	0.5	0.7	0.9

Table 6.2: Re and C_{LS} values used in the test matrix.

6.3 Surrogate Model-Based Optimizations

6.3.1 Data Sampling

In both the optimization frameworks, the initial meta-models were trained by means of selected samples inside the design space. In this work, the design space includes s=8 parameters (i.e. 6 airfoil geometry parameters, Re and C_{LS}), and the vector of independent variables reads as:

$$F = (X_{a2}, Y_{a2}, Y_{a1}, X_{a3}, Y_{a3}, Y_{a4}, Re, C_{LS}) \tag{6.4}$$

Instead of testing all possible combination of variables (factors), we used CCI design of experiment, with ν set to 4 to keep rotatability. As a consequence, the number of factorial runs is $2^8 = 256$. The operating ranges for all the factors and the levels at which they were tested are shown in Table 6.3. The total number of experimental trials was equal to

Variable	**-1**	$-1/\nu$	**0**	$1/\nu$	**1**
X_{a2}	0.0140	0.0282	0.0330	0.0377	0.0519
Y_{a2}	0.0217	0.0423	0.0491	0.0560	0.0765
Y_{a1}	0.0126	0.0332	0.0401	0.0470	0.0676
X_{a3}	0.0049	0.0136	0.0164	0.0193	0.0280
Y_{a3}	0.2761	0.3430	0.3653	0.3876	0.4545
Y_{a4}	0.0265	0.0599	0.0710	0.0821	0.1155
Re	300000	862500	1050000	1237500	1800000
C_{LS}	0.1	0.4	0.5	0.6	0.9

Table 6.3: Tested levels of design variables

$N = 273$. All 273 examples were tested by means of XFoil to calculate the vector of design objectives, $O = (\epsilon, \alpha)$. Notably, the entire sampling task required approximately 1.5 hours on a 4 cores i7 laptop.

6.3.2 Least Square Method

The model fitness function was approximated using a second-order polynomial response surface, which reads as:

$$\begin{aligned} O = \zeta_0 + \zeta_1 F_1 + \zeta_2 F_2 + ... + \zeta_S F_S + \\ \zeta_{1,1} F_1^2 + \zeta_{2,2} F_2^2 + ... + \zeta_{S,S} F_S^2 + \\ \zeta_{1,2} F_1 F_2 + ... + \zeta_{S-1,S} F_{S-1} X_S + q \end{aligned} \tag{6.5}$$

The polynomials quantify the relationship between the vector O of objective functions, ϵ and α respectively, and the factors ζ_i are the regressors and q is an error associated with the model. This first surrogate model is suitable for optimization processes and can handle noisy optimization landscapes, as reported in [113],[114].

6.3.3 Artificial Neural Network

The neural network developed was a multilayer perceptron (MLP). A series of tests on the architecture of MLP was carried out to identify the best trade-off architecture to ensure accuracy and limit the computational costs. Following these tests the selected configuration MLP included one hidden layer, 25 neurons, a logsig activation function on the hidden layer and a linear function on the output layer. Test were made by using 10 to 60 neurons, 1 and 2 hidden layer for a total of 6 tests on a single layer network and 36 tests on a double layer. The architecture with one hidden layer and 25 neurons had the best surrogate model coefficient of determination.

To avoid converge into a local minimum and detect overfitting the sampling dataset was re-organized into 5 different datasets; each was then split into a "training dataset" containing 80% of the total samples and a "test dataset" with the remaining samples. The ANN was then trained on each dataset and the accuracy was evaluated by means of the coefficient of determination R_2, calculated above the entire normalized prevision set: ϵ and α.

6.3.4 Pareto Front Properties

To further analyse the SM assisted multi-objective optimization performance, the quality of results was assessed using the following Pareto front properties. Namely: (i), the number of Pareto optimal solutions in the set; (ii), the accuracy of the solution in the set i.e. measured by closeness to the theoretical Pareto-front solution; and (iii), the distribution; and (iv), the spread of the solutions [115]. To quantify these properties we computed a set of performance indices. Specifically: (i), Overall Non-dominated Vector Generation Ratio ($ONVGR$); (ii), Hyperarea (H); (iii), Spacing (Sp) and Overall Pareto Spread (OS).

$ONVGR$ is a cardinality-based index derived from the one defined in [115], and it is linked to the number of non-dominated solutions in the Pareto front. To compute this index, all the elements that populate the optimized solution sets (from all optimization framework: MOEA and SMs assisted optimizations) were combined in a unique Pareto front for each test case, according with the non-dominated sorting algorithm of NSGA-II. $ONVGR$ defines the percentage of unique Pareto front elements that are generated from metamodel-assisted optimization or using XFoil as fitness function. H identifies the area of objective space subtended by the Pareto solution set. Sp identifies the distance between the elements that populate the solution set. OS quantifies how widely the optimized solution set spreads over the objective space. In a two objectives optimization, it is computed as the ratio of the rectangle that is defined selecting the good and bad points (according to each objective functions) and the rectangle that has two vertices on the Pareto front extreme points. A solution set presenting higher OS value is characterized by wider spread.

6.4 Results

Standard MOEA results are first illustrated. Results from the developed SMs are then presented and compared against MOEA.

6.4.1 MOEA

The fitness function selected to compute the objective functions during NSGA-II algorithm was based on BEM modelling of the airfoil (i.e XFoil). Figure 6.4 shows the objective function values of the initial given population P_t for $Re = 1.05 \times 10^6$ and C_{LS} ranging between 0.1 and 0.9. We analysed the initial population using XFoil with the aim to

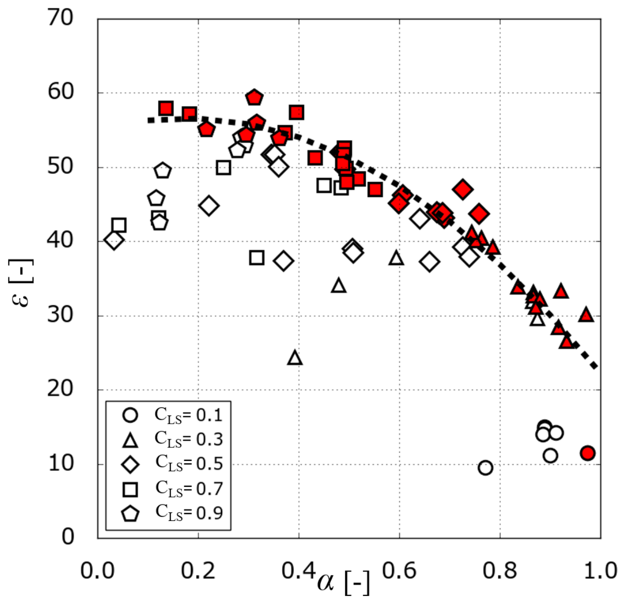


FIGURE 6.4: Original fitness function objectives ϵ and α for $Re = 1.05 \times 10^6$ and different values of C_{LS}. Filled symbols are used to interpolate a technological frontier (dotted line) for the baseline aerofoil geometries

find an initial frontier. We sorted the tested individuals according to fast non-dominated sorting algorithm suggested by Deb [38] obtaining the front depicted by filled symbols. Any optimization technique should move this imaginary frontier (dotted line) towards higher efficiencies and stall margins. Figure 6.5 shows results for all Re and C_{LS} combinations of the MOEA test matrix after 20 generations. Each graph, illustrates the initial frontier (dotted line) and the final frontier obtained by a selection of elements on the Pareto fronts. Irrespective to the configuration, the frontier shifts as a result of a major improvement in efficiency lowering the stall margin.

6.4.2 No Evolution Control (NEC) or single level approach

MOOP was solved using NSGA-II algorithm assisted by developed meta-models. SMs accuracy in terms of coefficient of determination R^2 for least square method and artificial neural network is reported in Table 6.4. In this method, there is no call to the original fitness BEM function-based, and the optimization was stopped after 20 generations. To assess the results of the SM assisted optimization, solutions on the Pareto fronts were retested with XFoil. Figure 6.6 shows the results of such an assessment for $Re = 1.05 \times 10^6$ and $C_{LS} = 0.5$. Irrespective to the value of Reynolds number and C_{LS}, the use of SMs to drive the optimization result inadequate with NEC approach. In fact, a consistent discrepancy was observed between the aerodynamic performance meta-model predicted (black dots) and the performance obtained testing the same individuals with the BEM function (white dots). A geometrical analysis made on few geometries of the SMs Pareto fronts,

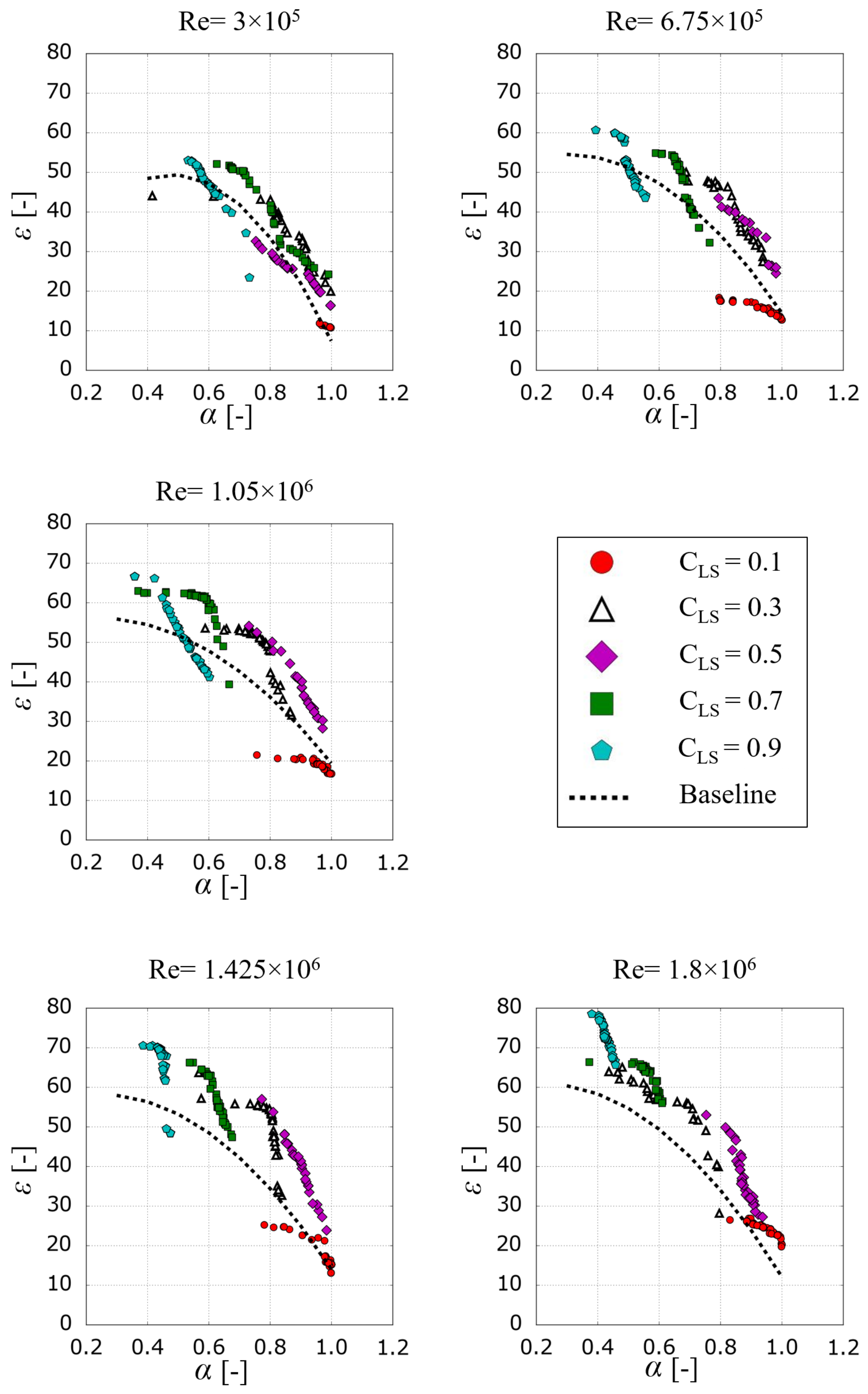


FIGURE 6.5: Pareto fronts for Re from 3×10^5 to 1.8×10^6 and C_{LS} from 0.1 to 0.9

confirms that the geometrical trends, previously found with MOEA, are not reproduced. In addition, as described in Figure 6.6, each Pareto front given by SMs provides similar

R^2		
Output	*LSM*	*ANN*
ϵ	0.9620	0. 9656
α	0.9480	0.9865
Global	-	0. 9906

TABLE 6.4: Surrogate models coefficient of determination for NEC approach

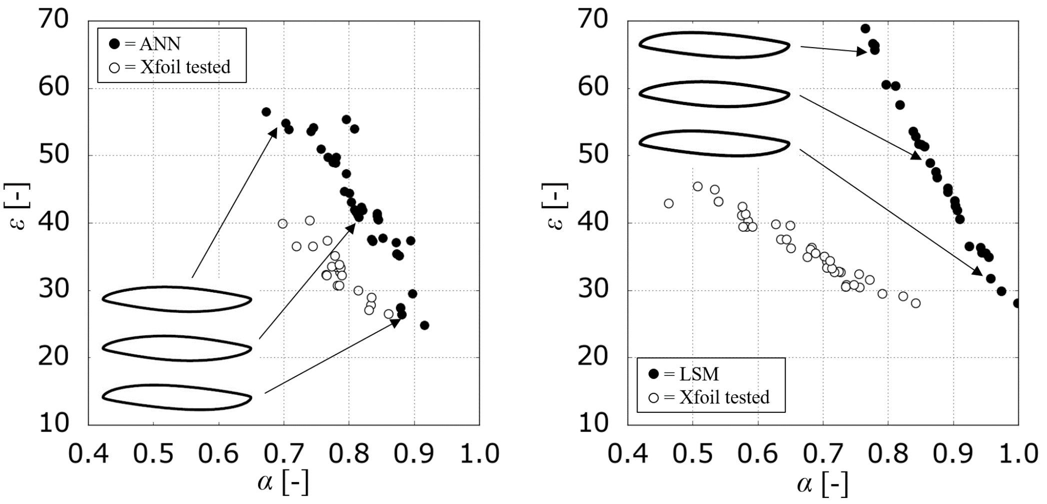


FIGURE 6.6: Final iteration (in black dots) and its XFoil verification (white dots) for NEC approach for artificial neural network (left) and least square method (right) for $Re = 1.05 \times 10^6$ and $C_{LS} = 0.5$

aerofoil shape in contrast to the flow physics, which suggest to find thinner aerofoils in zones with high efficiency and low stall margin and conversely an increased thickness in the zone with low efficiency and high stall margin.

To quantify the underlined departure of SMs optimization, the prediction capability (P_c) was plotted against the number of iterations in Figure 6.7 (i.e. $Re = 1.05 \times 10^6$, $C_{LS} = 0.5$). Notably, the prediction capability Pc is an index similar to the coefficient of determination R^2 computed as:

$$P_{C,\epsilon} = 1 - \frac{\sum_n \epsilon_{BEM,i} - \epsilon_i}{\sum_n \epsilon_{BEM,i} - \bar{\epsilon}_{BEM}} \tag{6.6}$$

where $\epsilon_{BEM,i}$ is the aerodynamic efficiency of the ith individual of Pareto front tested with BEM function; ϵ_i is the efficiency predicted by the surrogate model. Using the same criteria we have computed $P_{C,\alpha}$. Irrespective to the SMs, after an initial phase of 5 or 6 iterations in which the meta-model prediction is above 90%, P_c quickly drop for both the objectives ϵ and α. Despite P_c trends in ANN and LSM are similar, the optimization behaviours are totally different. In the test matrix ANN tends to empathize a geometrical feature that generate distorted or unrealistic geometries that are hard or impossible to simulate by means of XFoil, while this trend was seldom recorded for LSM. The inability of meta-model based optimized geometry to replicate the behaviour of aerofoils is strongly connected to the incorrect prediction of elements in the intermediate regions of the sampled design space for SMs instruction. This circumstance was already documented in highly dimensional problems (as in the present case owing to the selected aerofoil parametrization) when using single level NEC approach [45], [116].

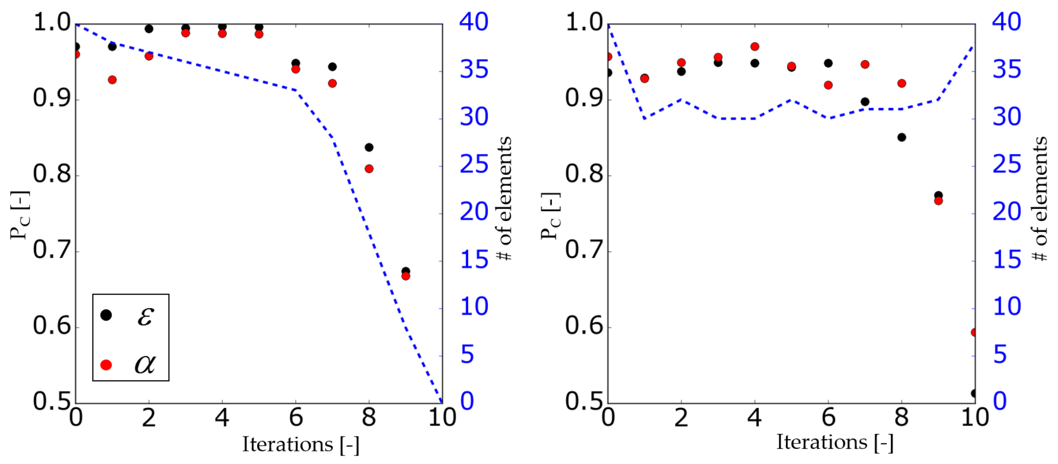


FIGURE 6.7: Prediction capability progression and optimized elements generated during the first 10 generations for both objective functions for $Re = 1.05 \times 10^6$ and $C_{LS} = 0.5$. Left:ANN; Right:LSM.

6.4.3 Indirect Fitness Replacement (IFR) or Bi-level approach

In view of the poor performance of NEC approach, the IFR framework was implemented for the solution of the MOOP. In this case, the initial sampling was used to train the first iteration of both meta-models. Then meta-model assisted NSGA-II was invoked for 5 generations. The genetic algorithm was kept identical to the original formulation used in the reference MOEA optimization. After the NSGA-II completion, the 40 best new members are evaluated using the original fitness objective function and then stored to enrich the SMs training pool. To this end, a limit of 10 calls to the objective function was set, to control the computational cost of the optimization process. Figure 6.8 illustrates the results of the final iteration loop again for the combination $Re = 1.05 \times 10^6$, $C_{LS} = 0.5$, already used in the NEC exploration. In the IFR approach, aerodynamic performance predictions are in good agreement with XFoil predictions, with a significant gap reduction between Pareto fronts estimated by the meta-models and by the original fitness function. In addition, it is possible to infer that the aerofoil flow physics is well represented, in both SM-based optimization, by the change in the aerofoils geometry. Thinner aerofoils populate the upper left region of Pareto front where the efficiency increases to the detriment of stall margin as expected and confirmed by MOEA analysis. Figure 6.9 shows the overall comparison of SMs assisted optimizations against the standard MOEA for five different configurations of the test matrix. Notably, in Figure 6.9 the dotted lines indicate the base-line aerofoil performance for the specified Re and C_{LS} values. To give additional hints about the ANN based surrogate, Figure 6.9 also shows the results of a reduced complexity level artificial neural network with only 10 neurons per hidden layer. Since the results obtained with the IFR approach demonstrates that the complexity of SMs had low impact on the quality of the obtained Pareto fronts, we performed the optimization with a simplified version of the ANN to reduce the computational effort required. These results give at a glance the indication of the quality of SMs assisted optimization implemented in a IFR test-bed, using half of the computational effort required by the MOEA. It is also

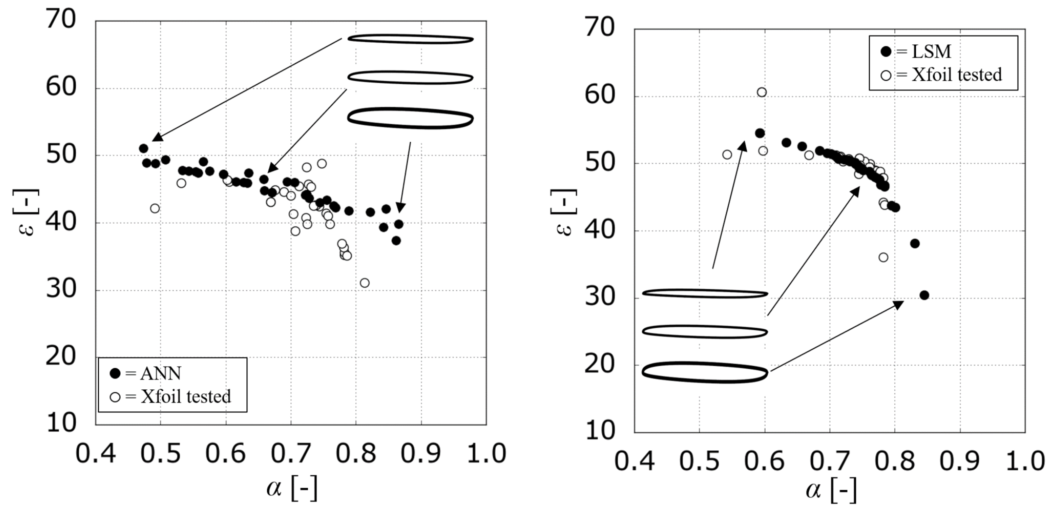


FIGURE 6.8: Final iteration (in black dots) and its XFoil verification (white dots) for IFR approach for artificial neural network (left) and least square method (right) for $Re = 1.05 \times 10^6$ and $C_{LS} = 0.5$

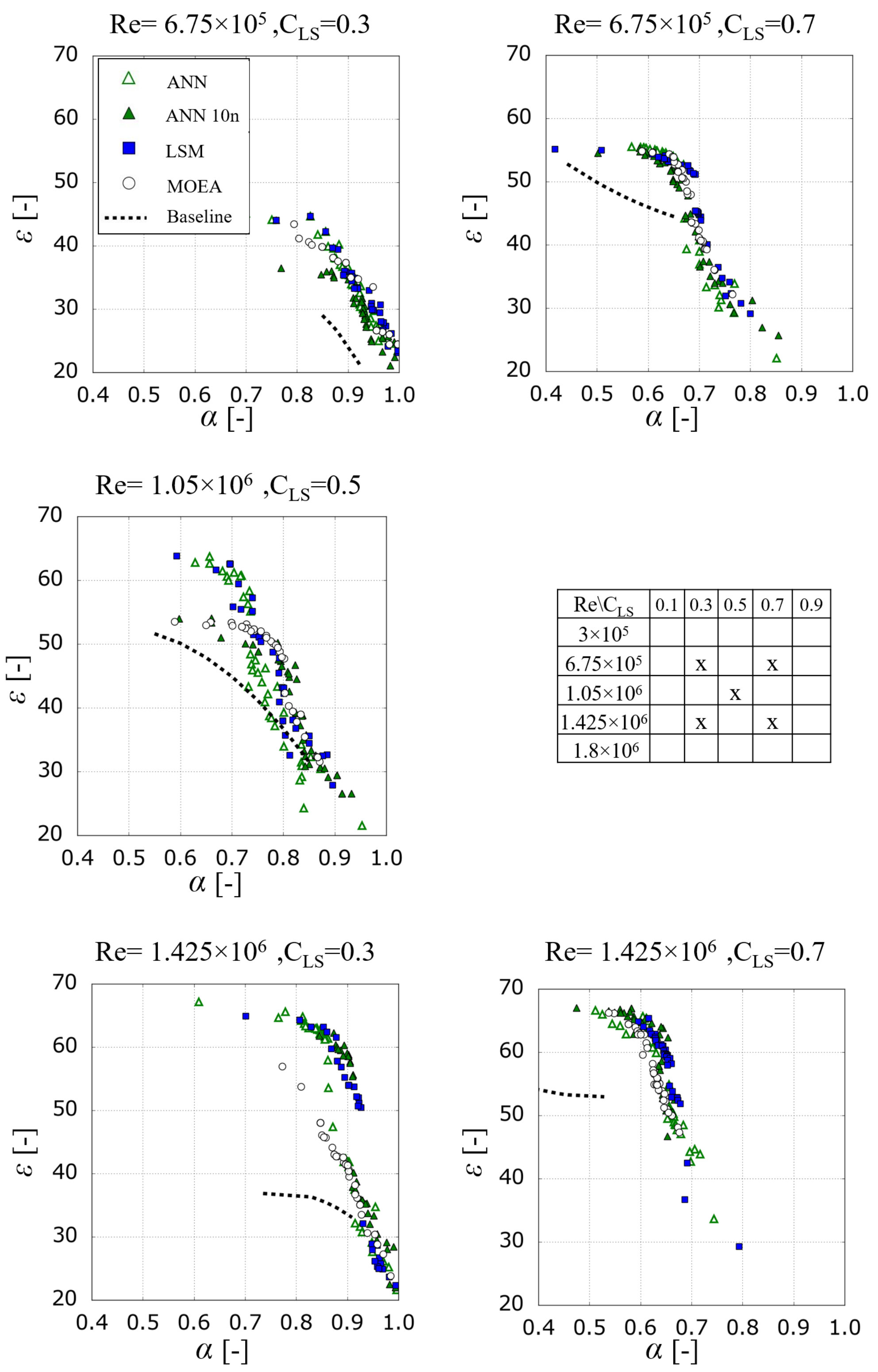
worth noting that, when comparing the two meta-models, the quality of the Pareto front solutions are close indicating a reduced influence of the optimization process to the specific meta-modelling algebra.

Table 6.5 reports the values of the indexes described in section 6.3.4. In each column, the best case is in **bold**, and the worst one in *italic*.

According to the values of *ONVGR* listed in Table 6.5, the SMs assisted optimizations resulted in a larger number of non-dominated elements in the set if compared to MOEA. The only exception concerns the central test case in the matrix ($Re = 1.05 \times 10^6$ and $C_{LS} = 0.5$) where MOEA optimization reaches the 33.3 % of non-dominated elements. However, we can obtain the same result using LSM as fitness function.

As per *H*, the use of ANN determines wider dominated area in the objective space, although LSM has very close values for the same test cases. As observed above, also in this case, worse results have been obtained using the MOEA approach. Concerning *SP* values, the analysis shows that ANN, even with 10 neurons, have in general Pareto fronts characterized by the minimum distance between the elements that populate the solution. The values of *OS* tend to confirm that ANN Pareto sets present wider spread over the objective space, in line with the *H* quality.

As a final comment, it is impossible to say that the overall quality of solution sets for MOOP cannot be accounted by means of a single performance index, especially if the shape of Pareto front is extremely different. In the end, since in this study the declared final objective was to move as much as possible the technological frontier on the α-ϵ chart, the effectiveness was assessed by means of *ONGVR* index concluding that SMs-assisted optimization can be equivalent or even better in this case with respect to that based on XFoil and surrogate models.



Re\C$_{LS}$	0.1	0.3	0.5	0.7	0.9
3×10^5					
6.75×10^5		x		x	
1.05×10^6			x		
1.425×10^6		x		x	
1.8×10^6					

FIGURE 6.9: Pareto fronts for five different test matrix configurations

Re	C_{LS}	*Fitness*	*ONVGR*	*H*	*Os*	*Sp*
675000	0.3	ANN	28.6	**43.0**	0.29	0.57
675000	0.3	ANN 10n	10.7	41.7	**0.32**	**0.28**
675000	0.3	LSM	**46.4**	42.4	0.20	**0.28**
675000	0.3	MOEA	14.3	41.6	0.15	0.40
675000	0.7	ANN	**43.8**	42.8	0.36	1.62
675000	0.7	ANN 10n	10.4	**43.0**	0.31	**0.26**
675000	0.7	LSM	37.5	41.8	**0.38**	0.66
675000	0.7	MOEA	8.3	40.4	0.15	0.60
1050000	0.5	ANN	11.9	**53.3**	**0.51**	0.80
1050000	0.5	ANN 10n	21.4	47.1	0.35	0.27
1050000	0.5	LSM	**33.3**	53.1	0.42	0.72
1050000	0.5	MOEA	**33.3**	45.2	0.24	**0.24**
1425000	0.3	ANN	18.4	**62.1**	**0.68**	1.28
1425000	0.3	ANN 10n	**52.6**	58.8	0.23	**0.29**
1425000	0.3	LSM	28.9	61.3	0.48	0.62
1425000	0.3	MOEA	0.0	53.0	0.27	0.97
1425000	0.7	ANN	32.0	45.8	**0.31**	0.62
1425000	0.7	ANN 10n	28.0	44.1	0.15	**0.06**
1425000	0.7	LSM	**40.0**	**47.8**	0.27	1.65
1425000	0.7	MOEA	0.0	43.8	0.10	0.30

TABLE 6.5: Pareto Fronts quality indexes for different optimizations techniques

6.5 Conclusions

This work presents an investigation of surrogate-based optimization of truly reversible profile family for axial fans. The Multi-Objective Optimization Problem (MOOP) is based on the NSGA-II, an Evolutionary Algorithm able to find multiple Pareto-optimal solutions in one simulation run. The MOOP has been firstly solved by using the XFoil software, avoiding the modelling and use of any SM. These results have been compared with those obtained by implementing different SMs in the MOOP, considering two different optimization frameworks, namely NEC and IFR. The use of SMs to assist MOEAs is a complex matter which requires an exhaustive analysis of the entire optimization framework and how the SMs are embedded in the considered optimization algorithm. This work has shown that, for this problem, an IFR approach has to be preferred to a NEC approach. In fact, a NEC approach has produced false Pareto-fronts in all the tested configurations. In particular, the NEC optimization has been not able to explore the entire design space in between the sampling points. The genetic algorithm led the optimizer to candidate solutions that were, during every iteration, more and more distant from the corresponding true value. This interpretation was suggested by the decreasing value of prediction capability associated to the increasing of unrealistic geometries and thick profile. An IFR approach has produced more reliable results, providing a good prediction capability during the iterations and, hence, reducing the distance between estimated and true Pareto fronts. The optimization was able to cover the entire design space, providing a geometry along the Pareto fronts similar to those experienced by the MOEA optimization. The IFR method has required a more complex and computationally expensive optimization framework based on a bi-level approach, with the infill criterion based on the results of the GA. An analysis of the ONVGR index have shown that, in most cases, the IFR optimization was able to produce better results (individuals) than the XFoil-based optimization. Regarding the SMs used in this work (a LSM and ANN with different number of neurons) IFR results show the independence of the MOEA to the SM considered. This is considered an important outcome being a LSM easier and faster to model if compared to an ANN. On the other side, NEC approach results do not present any relevant differences between the two SMs, even though different prediction accuracies were reached. The different reliability of the results obtained adopting a NEC and IFR approach, scales down the role and importance of the initial sampling in SMs based optimization. In fact, it is evident that a NEC approach is totally favourable to exploitation, being, on the contrary, not able to correctly explore the design space. An IFR approach, which involve an infill criterion, overcomes the difficulties and limitations related to the correct initial sampling creation, by iteratively adding optimized solutions evaluated with XFoil to the initial sampling. This approach ensures a better balance between exploration and exploitation of the design space. Results shown for the IFR approach demonstrated that, in the selected cases, a consistent reduction of MOOP computational cost is possible. In the specific case, according to the metric selected to judge the Pareto frontiers, the calls to the expensive objective function were reduced of 50% producing in all cases a reliable set of better solutions in comparison with the standard MOEA approach. In conclusions, this study clarified the strong impact of the optimization framework on the reliability of the obtained Pareto fronts in a SM-based design optimization for an industrial fan benchmark problem. With an IFR approach, the use of the considered SMs can significantly reduce the computational time needed to solve the MOOP by reducing the call of the fitness function, while ensuring the creation of the same of even better Pareto fronts if compared to those obtained by using only XFoil. The solutions obtained by IFR method present higher values of Pareto quality indexes if compared with MOEA (see Table 6.5). The major improvement is associated to the ONVGR index, which asses that the elements populating the Pareto front obtained with the SMs assisted optimization dominate the solution obtained with the MOEA.

Chapter 7

Machine-learnt topology of complex tip geometries in gas turbine rotors

7.1 Introduction

The purpose of this work is to develop a data driven strategy to explore the influence of tip configurations on performance in high-pressure turbine stages and to generate new designs. The data driven strategy has been developed exploiting an existing database of optimized HPT blade tips [58]. First, we performed exploratory data analysis (EDA) to improve the quality and readability of CFD-based optimization dataset. To this end, a set of statistical methods have been used to reduce outliers and skewness of data avoiding the presence of redundant information.

Then we have developed a new continuous parametrization of the tip surface by means of principal component analysis and unsupervised learning algorithms (i.e., clustering methods). The new tip representation allowed describing the surface with greater freedom obtaining high fidelity levels and using only few continuous variables. In particular, the continuous representation overcomes the limits of the topological approach emerged in [58].

Finally, two meta-models with different complexity level, respectively based on back-propagation and inductive ML algorithms, have been developed to predict HPT aerodynamic efficiency and tip heat transfer. Such meta-models have been used in a preliminary design process generating new tip surface geometries, which aim to control the overtip leakage flow and the turbine stage performance.

This work also clarifies the correlation between several subdivisions of tip surface (i.e., pressure side outer rim, suction side outer rim, trailing edge, internal part) and turbine stage performance and flow structures behind rotor trailing edge.

7.2 Case study

The dataset of HPT blade tip geometries, used in the present work, resulted from the multi-objective optimization study carried out by De Maesschalck and colleagues [58]. In particular, we exploited the computations coming from the squealer-like tip parametrization strategy [58]. The dataset included $n = 803$ three-dimensional simulations, in which a periodic section of the blade row is modelled while only the top 6% of the rotor span can change shape. The tip surface is fully parametrized with 214 different blocks, that can be either part of a rim or a cavity, and the block status is described by a binary code (see Figure 7.1). A more detailed description of the discrete parametrization can be found in [58]. Objective functions were tip heat load, which is defined as the heat transfer integrated over the entire rotor surface exposed to the tip deformation, and the rotor aerodynamic

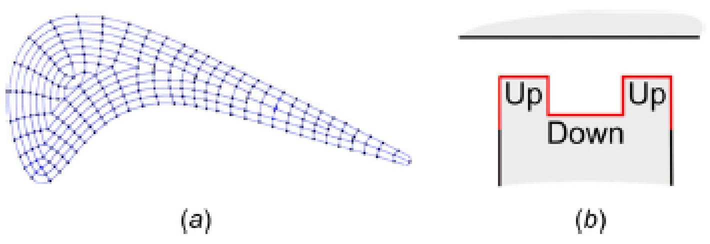


FIGURE 7.1: Block discretization for the squealer-like optimization: (a) block discretization pattern and (b) graphical example of the block-up/block-down tip surface generation

efficiency. For each explored individual, 78 performance parameters were available to describe the flow field with a particular focus on kinematic/dynamic field, heat transfer and secondary flows. All the steady-state RANS simulations were performed by means of a commercial software using the k-ω SST model for turbulence closure. As a consequence, each sample is therefore described using 214+78=292 parameters and the matrix of dataset $\boldsymbol{X}$ reads as:

$$\boldsymbol{X}_{n\times 292} = (b_1, b_2, ..., b_{214}, Var_1, Var_2, Var_{78}) \tag{7.1}$$

where, "*Var*" are the aforementioned continuous performance variables (i.e., aerodynamic efficiency, tip heat load, vorticity, mass-flow etc.) and "b" are the binary variables identifying the tip surface cell status: $b = 0$ if the block is part of a cavity, $b = 1$ if is part of a rim.

7.3 Dataset Pre-processing

Pre-processing makes use of multivariate statistics to detect and remove outliers, improve variable probability distributions and reduce dataset dimension.

7.3.1 Dataset quality

Box and whisker plot computation allows detecting anomalies using quartile discrimination within the dataset. Specifically, the whiskers extend 20% from the inter-quartile range .From this quartile-based discrimination, 122 samples resulted outside the range of acceptance at least on one of the performance variables. In this number, 64 outliers were found to be consequence of errors occurred during the optimization chain (mostly related to the numerical domain generation). The remaining samples were found to be linked to borderline configurations of the cells describing the tip surface. For this reason, they have been kept inside the dataset preserving the quantity of achievable information. The distribution of dataset variable box and whisker plots, before and after the removal of outliers is shown at a glance in Figure 7.2. The second pre-processing focused on dataset density plots based on the kernel density estimation [117], [118], in a view to correct the skewness of performance variable distributions in order to compensate any departure from normality. The Yeo-Johnson transformation function was used to correct dataset improving the probability distribution of such variables. Figure 7.3 compares the original pattern distributions against the transformed ones obtained with use of symmetry recovering Yeo-Johnson and provides the ex-ante ex-post skewness values. The pre-processed dataset matrix, after outlier removal and shape distribution recovery, now reads as:

$$\boldsymbol{X}_{k\times 292} = (b_1, b_2, ..., b_{214}, \tilde{V}ar_1, \tilde{V}ar_2, \tilde{V}ar_{78}) \tag{7.2}$$

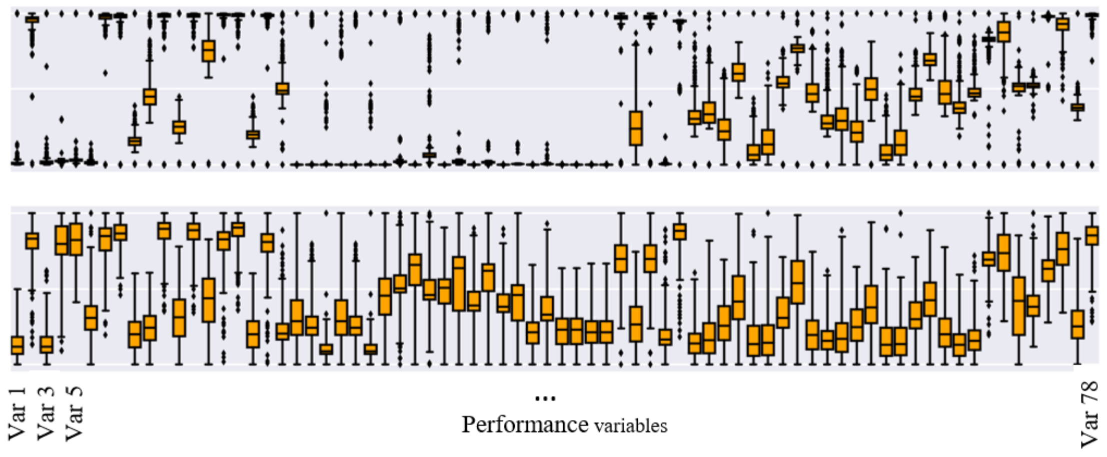


FIGURE 7.2: Box and Whiskers plots with outliers (upper) and without outliers (lower)

where, k = 735 is the number of valid samples, and $\tilde{Var}$ are the performance variables after Yeo-Johnson transformation.

7.3.2 Dataset dimension

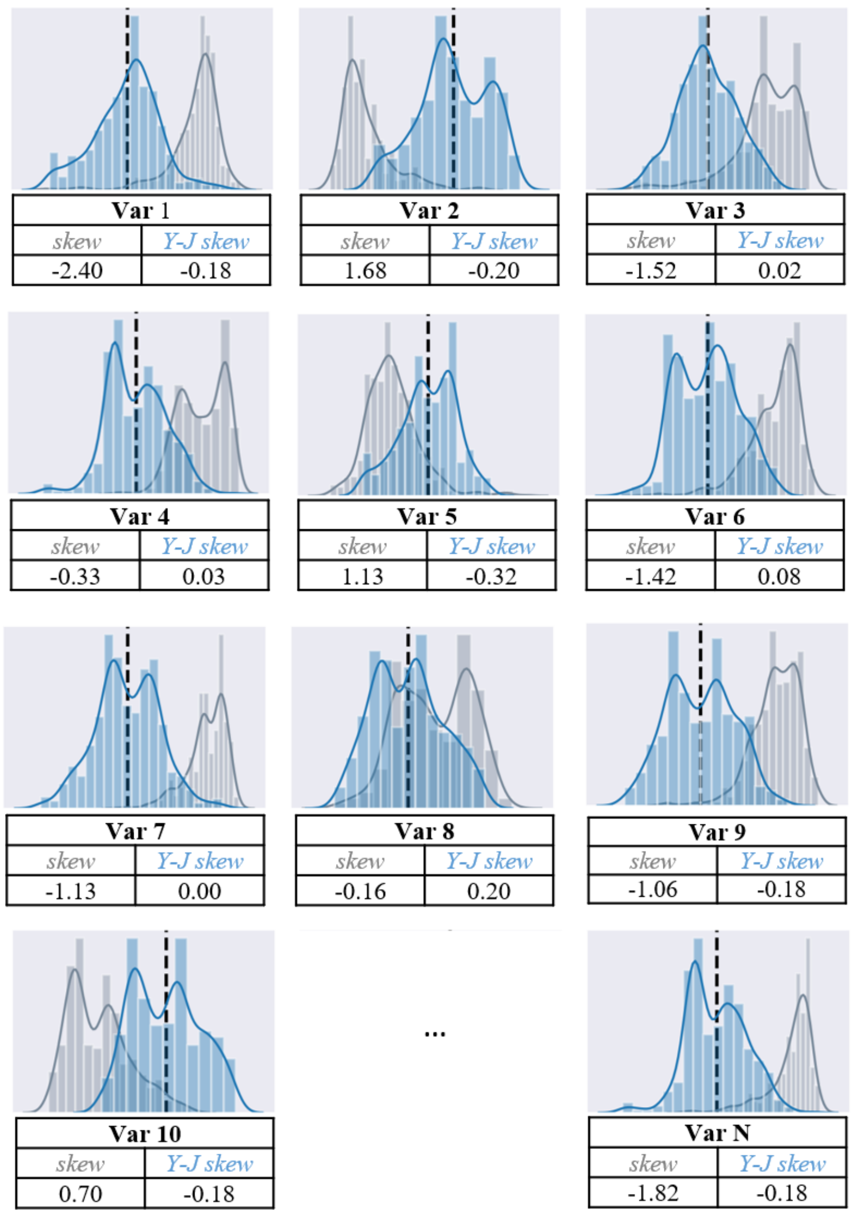
The reduction of the dataset dimension was, then, undertaken to eliminate unnecessary or duplicated variables. It is worth nothing that this redundancy reduction differs from dimensionality reduction approaches, e.g. typically tackled with unsupervised classification methods like principal components analysis. Both methods seek to reduce the number of attributes in the data set, but dimensionality reduction asses this task creating new combinations of attributes, whereas the selection of variables here performed aims to include/exclude attributes already in the dataset without any feature creation. To avoid redundancy and improve the readability of data, the data matrix $\boldsymbol{X}$ has been explored using heat map graphical analysis. This method is a representation of the Pearson correlation computed between each pair of variables (performance parameters). Figure 7.4 illustrates the heat map obtained for the original dataset $\boldsymbol{X}$. The output of such correlation analysis resulted in a further reduction of the dataset. Notably, all the variables in the heat map having a Pearson correlation coefficient higher that 0.96 or lower than -0.96 have been removed from the dataset because they are providing redundant information. According to the original work of Pearson [89], absolute correlation coefficients higher than 0.95 feature closely related variables. In this work, we set the threshold to 0.96 to reduce the risk of eliminating useful information. The number of variables needed to describe the flow field passes from 78 to 26 performance parameters, as a consequence the matrix of data reads as:

$$\boldsymbol{X}_{k\times 240} = (b_1, b_2, ..., b_{214}, \hat{V}ar_1, \hat{V}ar_2, \hat{V}ar_{26}) \tag{7.3}$$

here $\hat{V}ar$ identifies the performance variables after reducing the dataset dimensions.

7.4 Statistical model of the tip shape

The strategy used to derive a statistical model of tip shape was based on the use of factorial analysis and unsupervised learning algorithms. Specifically, PCA has been used to decrease the dimensionality of the problem. The first eight lower-order PCA components were considered accounting for more than 85% of cumulative explained variance. The PCA



	skew	Y-J skew
Var 1	-2.40	-0.18
Var 2	1.68	-0.20
Var 3	-1.52	0.02
Var 4	-0.33	0.03
Var 5	1.13	-0.32
Var 6	-1.42	0.08
Var 7	-1.13	0.00
Var 8	-0.16	0.20
Var 9	-1.06	-0.18
Var 10	0.70	-0.18
Var N	-1.82	-0.18

FIGURE 7.3: Data shape improvement applying the Yeo-Johnson function

analysis was used to develop a new parameterization of the tip surface that can reproduce each individual (i.e., every possible configuration of the 214 cells) using 8 parameters, i.e. the first eight latent variable scores. Figure 7.5 illustrates the proposed statistical tip shape representation. Specifically, three profiles (A, B and C respectively) are shown as examples of tip topologies obtainable using the shape representation. The main advantage, resulting from the use of such a representation, regards the generation of new individuals when coupled with genetic operators (e.g. crossover, mutation and random generation).

One of the drawbacks in the original representation, based on almost 200 binary variables, was the generation of ineffective shapes due to "chessboard" like configuration of the cells. The reduced-order but continuous parameterization, on the contrary, overcomes this problem because the PCA score values Ui define a topology basis which already combines the overall tip geometry (not just local information) and turbine rotor aero-thermodynamic performance. In addition, the effectiveness of crossover and mutation operators increases

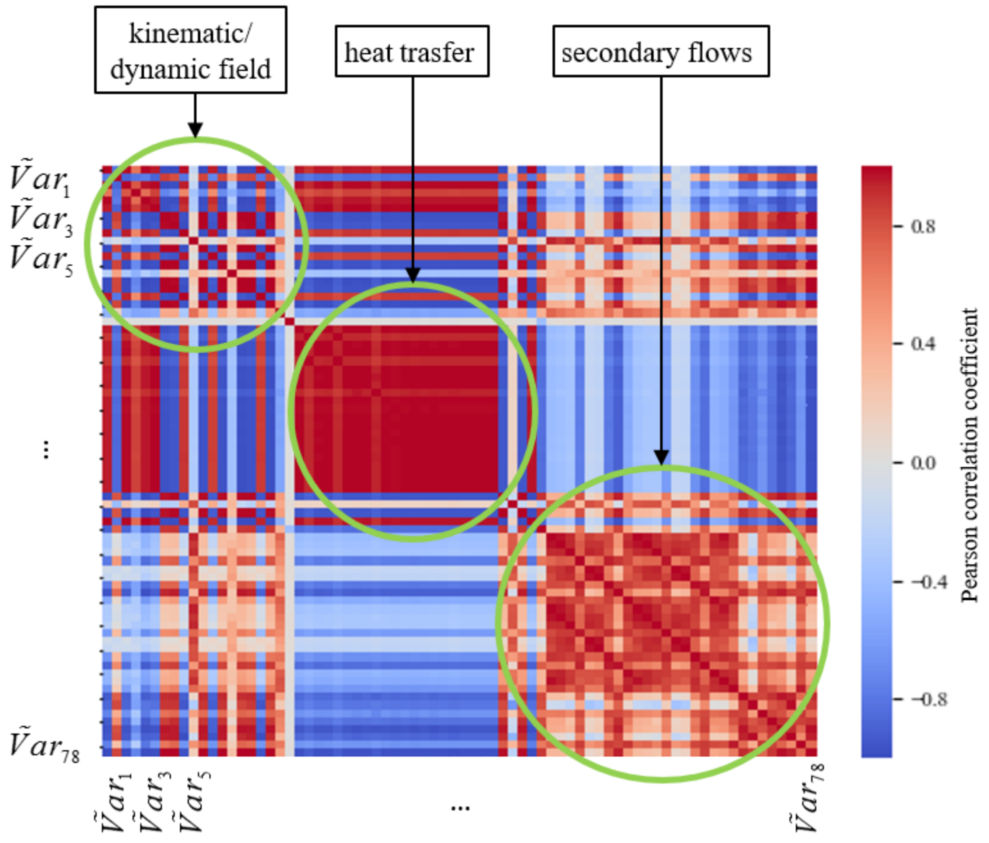


FIGURE 7.4: Heat map based on the correlation matrix

too because the recombination and mutation will involve only the most relevant genes.

To quantify the validity of the proposed topology approach, Table 7.1 illustrates the error in inverse transformation from the statistical PCA representation to recover the binary ones. Notably, with a mean error of 0.22% (i.e., less than one cell is misrepresented on average).

Max error	**Min error**	**Mean error**
2.33% (<5 cells)	0% (0 cell)	0.21% (<1 cell)

TABLE 7.1: Statistical representation error rate

7.5 Pareto front exploration methodology

To the end of exploring the original Pareto front design space, we developed an optimization strategy based on the use of NSGA-II algorithm. In particular, the statistical shape model coupled with genetic operators (i.e., crossover and mutation) is used to generate new individuals exploiting the advantages mentioned in the previous section. The obtained individuals, described by the reduced ordered parameters, are then back transformed to the original binary parametrization in order to be tested. Instead of using computationally intensive CFD simulations, we developed two metamodels for tip surface performance prediction.

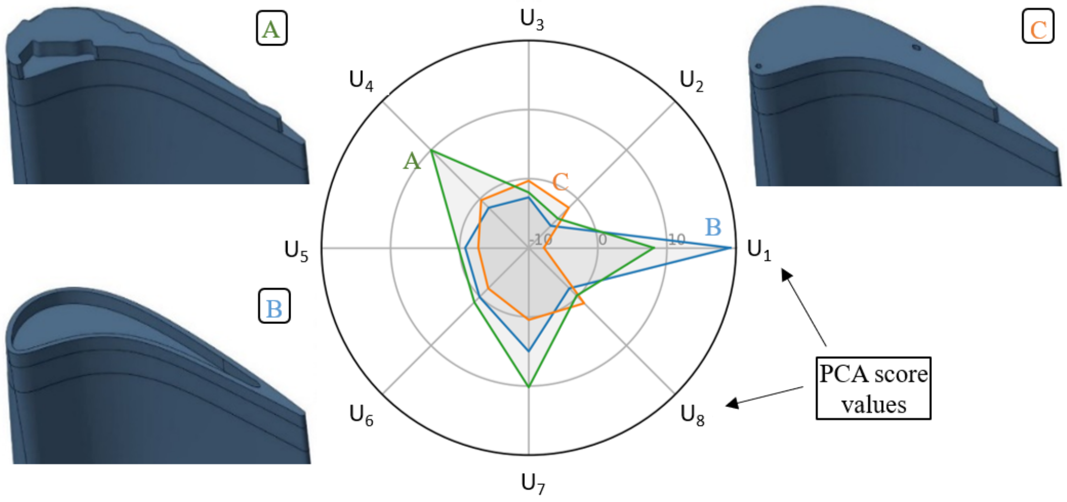


Figure 7.5: Tip surface continuous representation

7.5.1 Metamodeling performance prediction

In order to explore the use of the novel statistical tip shape modeling, two meta-models, namely an artificial neural network (ANN) and a gradient boosting regressor (GBR), have been developed to enrich the available Pareto front in terms of aero-thermodynamic performance metrics.

The ANN is a multilayer perceptron trained using the back propagation technique. The network has 2 hidden layers and 35 neurons on each layer. The hyperbolic tangent and linear activation functions have been selected for the hidden and output layers respectively. The weights have been optimized according to the Limited memory BFGS (L-BFGS), an optimizer in the family of quasi-Newtonian methods outperforming widely used Adam algorithm when applied to our dataset.

The GBR has been selected according to its attitude in handling binary coded data [59]. The boosting based meta-models has 1500 predictors, while the least absolute deviation has been selected as lost function. It is a highly robust loss function solely based on order information of the input variables [119].

The cross validation has been applied during the training phase. In the following, we report the regression results obtained using both metamodel to predict efficiency and tip heat load. The GBR meta-models reaches the highest prediction capability during the test phase (performance indicators are listed in Table 7.2). For both meta-models, great part of the error observed during the test phase was related to the prediction of samples in the range of low efficiency (see Figure 7.6). coming from the original optimization design of experience. The database has few samples in that design space region.

7.6 Results

Firstly, unsupervised methods have been used to unveil tip geometry performance correlation.

Successively, ANN and GBR based meta-models are used to carry out the optimization of HPT tip geometry following two criteria. The first genetic exploration is meant to extend the efficiency-heat load Pareto front with new individuals. The second application, then,

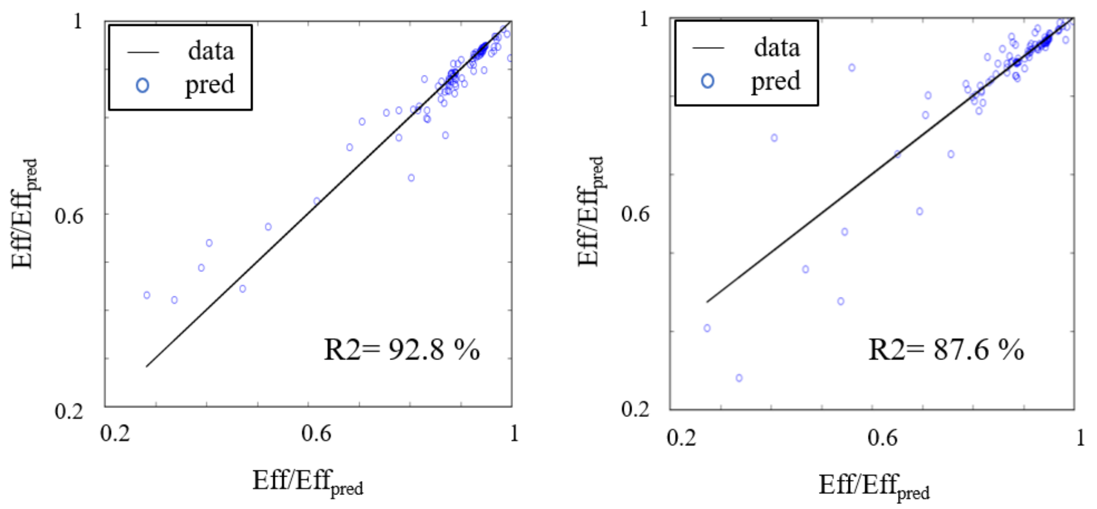


FIGURE 7.6: Efficiency prediction results obtained during test phase using GBR (on the left) and ANN (on the right)

Meta-models	**Predicted Output**	**Explained Variance**	**Root Mean Squared Error**	R^2
GBR	Eff	0.927814	0.036145	0.927796
GBR	THL	0.890007	0.066322	0.889368
ANN	Eff	0.875673	0.047439	0.875621
ANN	THL	0.839559	0.081541	0.832768

TABLE 7.2: Statistical representation error rate

aims to design individuals able to control the secondary flow keeping optimal aerodynamic performance. In both applications, the meta-models are used to obtain optimized profiles targeting specific areas of the design space.

7.6.1 Geometry-performance correlation

The PCA loadings, which quantify the statistical significance of each cell, have been used to group approximately 200 cells (i.e., the original discrete geometry description) into 10 clusters. A graphical representation of loadings is illustrated in Figure 7.7 where V1, V2 and V3 are the loading values on the first three components respectively. Irrespective to the 8 principal components initially considered in PCA analysis, only the first three components have been considered to correlate geometry and performance because they allow to clearly catch the functional zones of tip surface (i.e., pressure side outer rim, suction side outer rim, front and rear inner surface).

Contours revel that the tip surface may be divided into distinct regions characterized by a well-defined loading value. These regions are concentric (V1 and V2) contours while they follow the camber-line (V3 contour). In details, the first and second components determine the shape of tip surface inner part. Moreover, V2 is responsible for the trailing edge geometry while V3 determines the outer rim of the tip surface in the suction side.

Since cells with similar loading value on the first three principal components (i.e., neighbouring cells in V1, V2 and V3 space, reported in Figure 7.7) features comparable

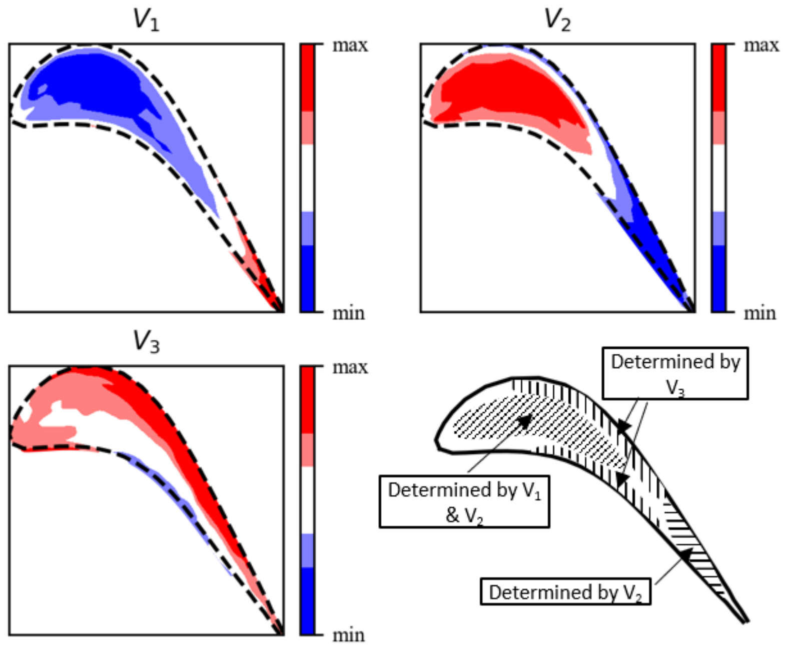


FIGURE 7.7: Loadings contours on first (a), second (b) and third (c) principal component

(or identical) influence on the dataset variability, k-Means++ clustering algorithm has been used to collect them in 10 clusters (Ai, with "i" $\in [1, 10]$). The cluster number was suggested by the elbow method minimizing the cluster sum of square [120], namely the sum of squared deviations from each observations to cluster centroid. Each cluster identifies a confined tip surface portion (i.e., leading and trailing edge, pressure-side, suction-side, internal surface) illustrated in Figure 7.8. To evaluate the effect of each area portion Ai (or cluster) on rotor performance, we firstly computed the cluster filling factor (F_{Ai}) for each profile in [58] using Equation 7.4:

$$F_{Ai} = \frac{\sum_{j=1}^{n^\circ cells} b_j g Area_j}{\sum_{j=1}^{n^\circ cells} Area_j} \tag{7.4}$$

where $Area_j$ is the area of the j-th cell in Ai, while b_j is the cell status. Successively, projection to latent structures (PLS) has been applied to quantify the correlation between F_{Ai} and the couple efficiency-tip heat load. In this work, the tip geometry, in the form of F_{Ai}, is considered PLS input feature, while efficiency and tip heat load are output features.

From PLS analysis (Table 7.3) follows that the first three latent variables have a strong correlation between input and output scores. This means that is possible to find correlations between the loading coefficient of input and output features in all the three latent variables. From first component plot, Figure 7.9, it is clear that there is a direct correlation between the tip pressure side (F_{A4} and F_{A5}) and rotor efficiency. The efficiency is deeply connected with the blade outer rim in the middle zone of the blade surface, where the pressure rise between pressure and suction side increase increasing mass flow evolving in that over-tip region. As for the pressure side, the increase of the features related to the internal

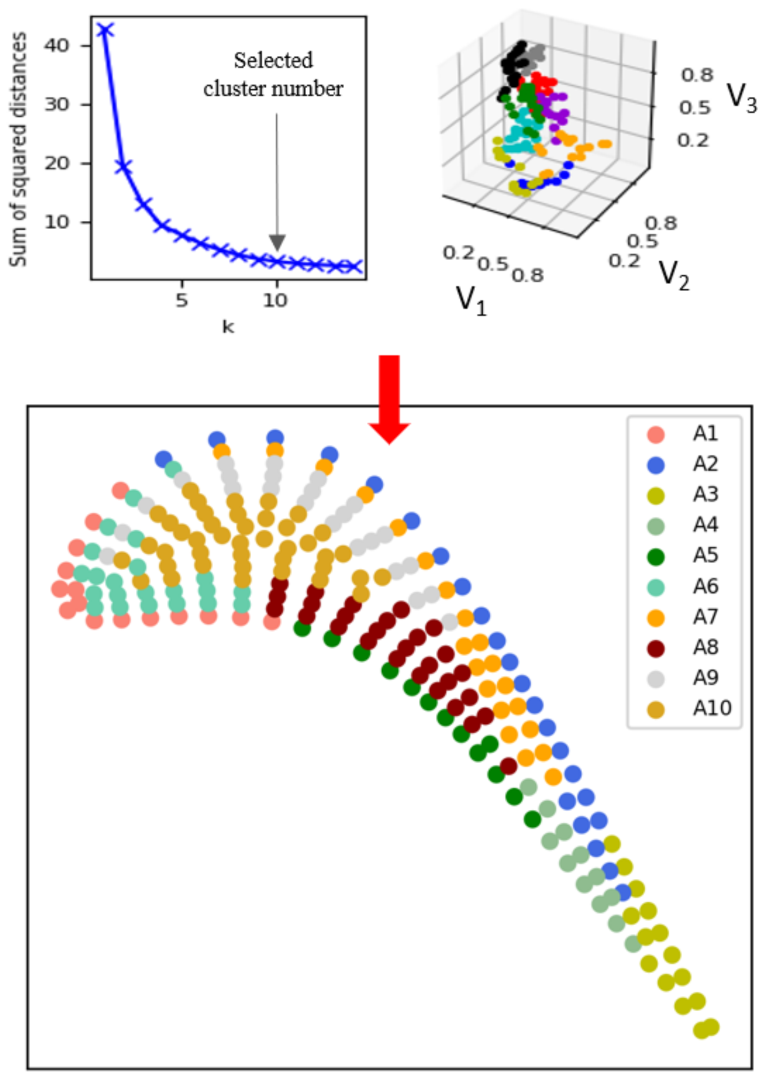


FIGURE 7.8: Elbow method and K-Means++ clustering method (upper part). Cluster representation on tip surface (lower part)

zone of the tip surface (F_{A8}, F_{A9} and F_{A10}) has a direct correlation with the efficiency value, while there is an inverse proportionality between the same geometric features and the tip heat load in the over-tip region of the blade. A rim in this area decrees the tip heat load. As for the aforementioned tip area sub-divisions, even the leading edge of the tip surface (F_{A1} and F_{A6}) results directly correlated with efficiency while there is an opposite correlation with tip heat load. All the other loading are lower than the threshold value (dashed line) and must be considered not relevant for this analysis (see Equation 7.5).

$$Tr(LV_a) = min(abs(X_{loading}, Y_{loading})) + [max(abs(X_{loading}, Y_{loading}))] \cdot \\ [1 - corr(X_{scores}, Y_{scores})] \tag{7.5}$$

Looking at the loading coefficients on the second component reported in Figure 7.10, we could see how increasing the filling density of the suction side (F_{A2}) leads to increase both efficiency and tip heat load, as well as an increment of F_{A3} and F_{A4} values. In particular, an open trailing edge (F_{A3}) leads to reduce the tip heat load with a lower effect on efficiency value. Between the loading values on third component, Figure 7.11, only the pressure side (F_{A5}) reaches the threshold values resulting directly correlated with efficiency.

Latent variable	**Correlation between X and Y scores**
1^{st}	75%
2^{nd}	68%
3^{rd}	39%

TABLE 7.3: Correlation between latent variables

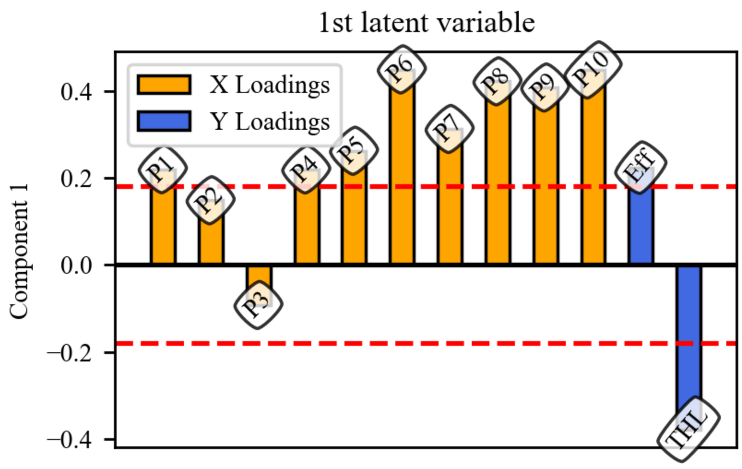


FIGURE 7.9: PLS loading on first component

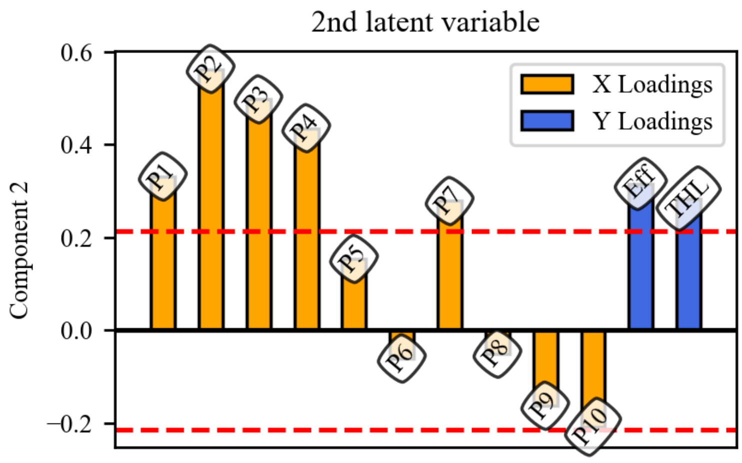


FIGURE 7.10: PLS loading on second component

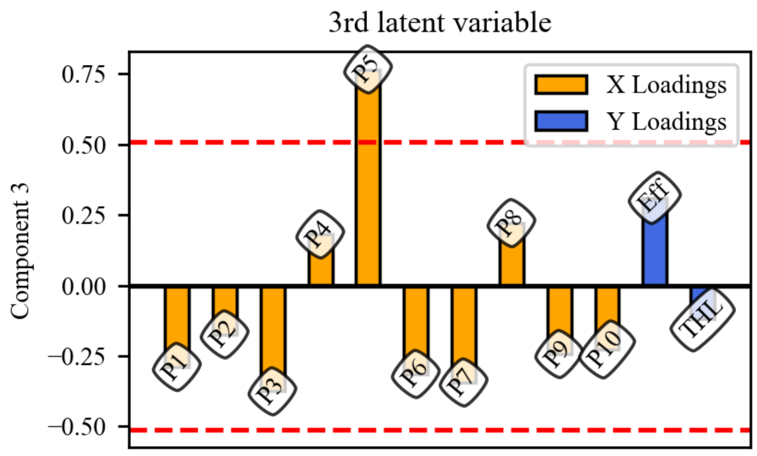


FIGURE 7.11: PLS loading on third component

7.6.2 Pareto front enrichment

The first Pareto front exploration makes use of two profile figure of merits, namely aerodynamic efficiency and normalized tip heat load. It is worth noting that the tip heat load is normalized by the heat flux over the flat tip blade and, for consistency, the aerodynamic performance of this flat tip profile are used as the reference case for the second objective function and for other performance indicators throughout the result section. Figure 7.12 reports the enriched Pareto front. Specifically, four new profiles have been generated. Profiles A and B with the goal of obtaining new best efficiency designs with alternative routes to CFD-based optimization, and profiles C and D to explore the design space filling up the discontinuities observed in the dataset Pareto front. Figure 7.12 reports all the tip configurations (open circles) tested by De Maesschalck and co-workers [58] highlighting the dominating solutions (triangles) and the new individuals (squares) obtained using the statistical topology-based optimization. Notably, the empty squares represent the metamodel predictions, while the solid squares are the results of CFD simulations. To give quantitative hints about the relative quality of CFD simulations Table 7.4 gives a summary of CFD prediction errors against the reference metamodel performance.

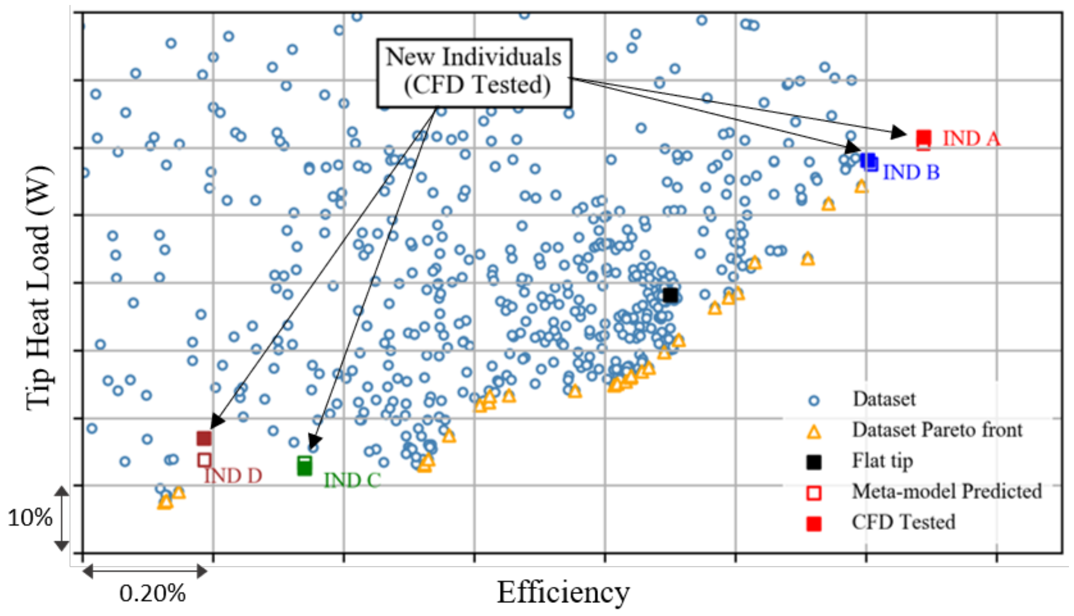


FIGURE 7.12: Pareto front enrichment

Ind	Variable	Prediction error [original - predicted]
A	Eff	0.00127
A	THL	0.60319
B	Eff	-0.000416
B	THL	0.01654
C	Eff	0.000126
C	THL	-0.3197
D	Eff	-0.00023
D	THL	1.45584

TABLE 7.4: Meta-model prediction error

The new geometries have been obtained by means of meta-model driven NSGA-II. ANN model resulted in design A and B, while GBR in individuals C and D. To assess the performance prediction quality, the new profiles have been numerically tested. From the error rate listed in Table 7.4 results that the error for efficiency prediction is lower than 0.2% for all new profiles while the error for THL prediction is lower than 2% for A,B and C. The prediction error for D is below 5.2%. The error is relatively low compared to max variability observed in the design space. Individual A and B overcame the efficiency value of the previous best individuals of the original dataset. In particular, A improves the performance parameter of 0.12% extending the dataset Pareto front towards higher efficiency values. In the bottom left region of efficiency-tip heat load plane, in which the profiles are characterized by lower values of tip heat load and efficiency, we can clearly see a reduced number of tested profiles and a discontinuity in the dataset Pareto front. That region has been explored and individual C and D have been developed to fill up the aforementioned discontinuity.

7.6.3 Vorticity control

Figure 7.13 plots the downstream vorticity as a function of aerodynamic efficiency for every individual in the population of tip geometries [58]. We can see the dataset Pareto front, previously observed in the objective function plane, and the new individual E developed to decrease the generation of vorticity keeping high value of aerodynamic efficiency, in particular a value near to the highest efficiency observed in the dataset. Individual E, developed according to GBR meta-model, resulted a geometric variation of an existing profile located in the dataset Pareto front. Specifically, the meta-models modified specific zones of the tip surface chancing the mass flow evolving in the over-tip region of the blade. To have an indication of the impact of these geometric variations on the downstream

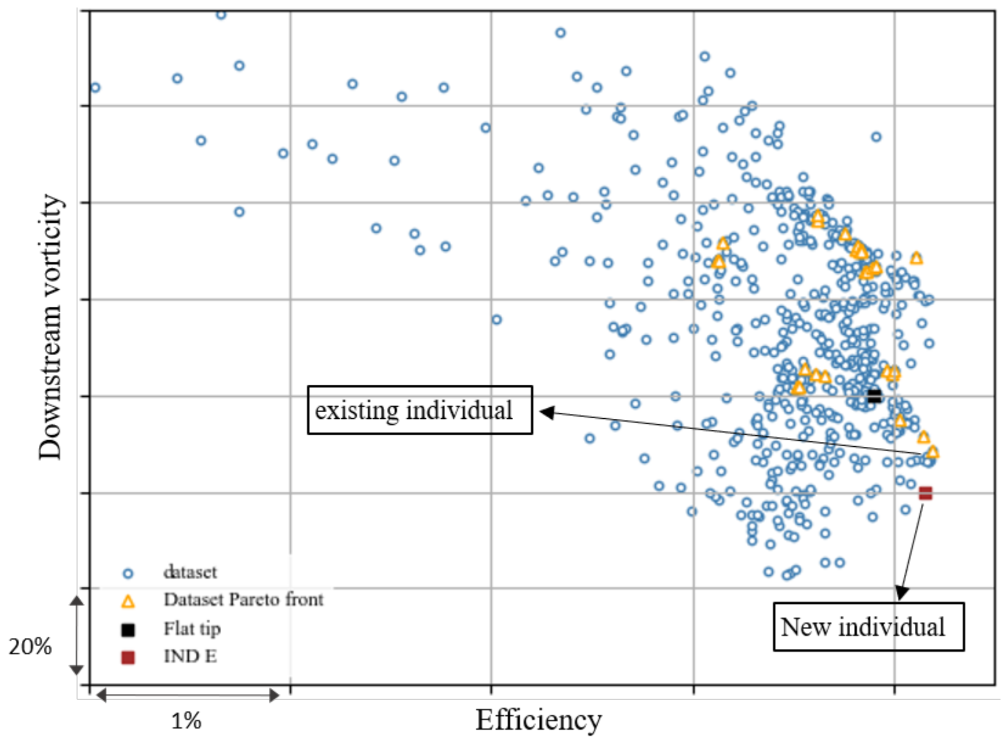


Figure 7.13: Downstream vorticity in function of aerodynamic efficiency

flow structures Figure 7.14 and Figure 7.15 compare the vorticity contours of the closest individual in the original Pareto front against individual E. In both contours, we can see an established vortex patter consisting of the tip leakage vortex (TLV) that results clearly separated from the upper passage vortex (UPV).

The changings observed in individual E, which affect only the upper 50% of the blade passage (both contours have the same vortex structure in the first 50% of rotor channel from the hub), significantly mitigate UPV and TLV strengths. The E tip geometry resulted in 15% decrease of the downstream vorticity in combination with nearly 0.1% drop in the aerodynamic efficiency giving the best compromise between secondary flow vorticity and efficiency.

The persistence in E contour of these separated counter-rotating vortices depends on whether the alterations of E tip geometry do not concern the external outer rim of the tip surface, so their impact on mass flow evolving through the tip channel and fluid exit velocity, main responsible of the observed flow structures, is limited. Surely, this is a desired behaviour related to the second objective of preliminary design: aerodynamic efficiency. As we previously see from the PLS analysis, the rotor tip outer rim strongly influences the efficiency.

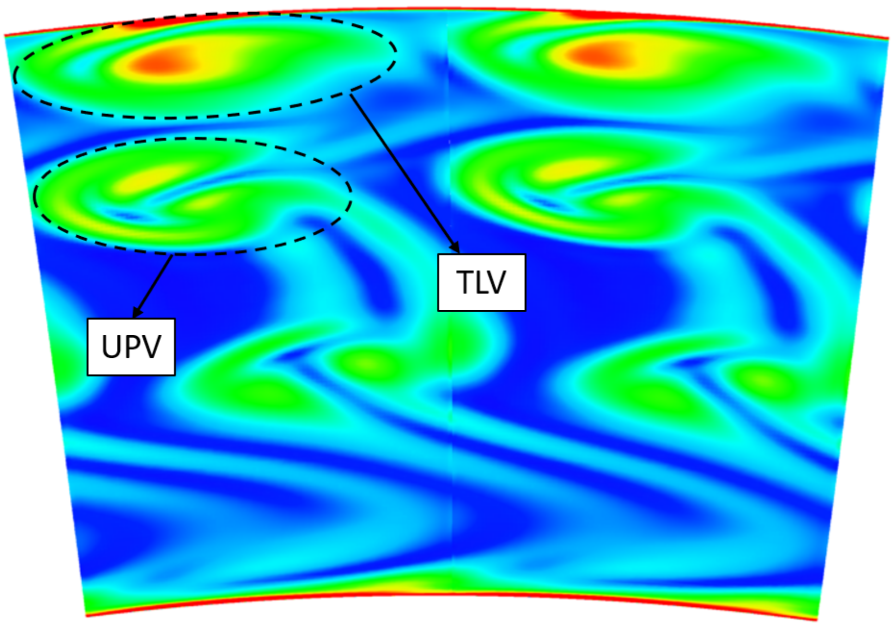


Figure 7.14: Downstream vorticity contours of existing individual

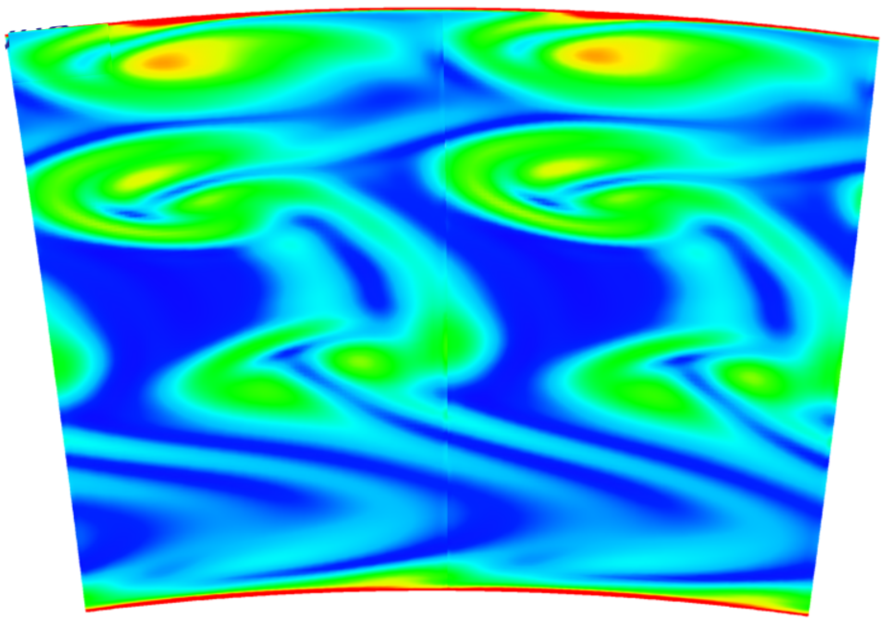


FIGURE 7.15: Downstream vorticity contours of existing new individual E

7.6.4 Conclusions

The current work developed a data driven based strategy to investigate the influence of tip geometry on performance in HPT blade and exploiting the acquired learning generating new profiles.

An existing database of 803 squealer-like blade tip has been explored. EDA has been conducted on database to improve quality and readability of data. Specifically, the application of Box and whisker plots allowed identifying and removing 63 outliers. Yeo-Johnson transformation improved the statistical shape of data reducing the performance variable skewness and then, the heat maps analysis based on the Pearson correlation reduced the dataset dimension removing unnecessary information.

To overcome the limitations arising from the use of topological parametrization, a new statistical representation of the tip geometry has been developed using the PCA analysis. The new parametrization, that describes the tip surface using the scores of the first eight low order principal components, decreased the amount of geometric variables keeping a fidelity level above 99%. Furthermore, the statistical representation improved the genetic operator effectiveness reducing the generation of infeasible shapes.

Two meta-models, namely an ANN and a GBR, have been developed to predict efficiency and tip heat load. Both meta-models have been trained using cross validation. ANN determined better performance during the training phase, avoiding overfitting. GBR reached the highest prediction capability during the test phase.

The geometry-performance correlation has been explored by means of unsupervised learning methods. First, the k-Means++ clustering algorithm, applied to the loadings of previous PCA analysis, has been used to identify tip surface areas in which cells featured comparable influence on dataset variability. Second, PLS analysis quantified the effect of those areas on efficiency and tip heat load. The presence of outer rim improves the efficiency while a rim in the internal part decrees tip heat load level. The trailing edge geometry resulted less influential on performance.

Aiming to generate new profiles with alternative routes to CFD-based optimization, two applications of ANN and GBR meta-models have been reported. First, four new profiles have been developed to spread dataset Pareto front. Individuals A and B outperformed previous best efficiency geometry while C and D fixed a design space discontinuity. Second, individual E has been designed to control secondary flow keeping efficiency to the maximum. E improved reduced downstream vorticity by 15% reducing the TLV strength with a negligible efficiency decrease.

Chapter 8

Conclusions

This thesis proposed new strategies for the implementation of elements from classical industrial turbomachine design, performance analysis and optimization into advanced design procedures inspired by Industry 4.0 initiatives. Great effort has been put on interweaving Machine Learning and Deep Learning techniques with old design and performance analysis procedures. The learning algorithms have been deeply analyzed and described, highlighting their advantages and limits, and evaluating the role they played into the transformation of all manufacture activities experienced during the last few years by turbomachinery community.

In the first chapter Machine Learning and Deep Learning have been introduced and briefly described. The chapter clarified the new programming paradigm produced by the cited learning approach within the artificial intelligence landscape. Final part of this chapter reported a review of machine learning application in turbomachinery applications.

The key concept of turbomachinery machine-learnt design has been introduced in the second chapter. Firstly, the classical approach to design, analysis and optimization has been described, than the role of machine learning has been deeply discussed suggesting the potentiality coming from the implementations of smart technologies within each step involved by traditional procedure.

All the machine learning and deep learning algorithms implemented in this work have been described in detail in third chapter, while the importance of data analysis as preprocessing tool to improve data quality before any deeper analysis has been proven in the fourth chapter.

In the last three chapters were given three machine-learnt strategies, each one involving a step inside classical design approach. In the application reported in fifth chapter axial flow turbomachinery design space has been explored using and artificial intelligence based meta-model for performance prediction. The meta-model has been introduced in AxLab, in house axi-symmetric flow solver, improving its prediction accuracy. The dataset of tested rotor performance has been analyzed using principal component analysis to reduce the dimensionality of the problem, while projection to latent structure allowed to find correlations between geometric parameters and rotor performance. Such correlations have been used to developed multi-dimensional Balje charts enriching the information available during the preliminary design with new relations accounting the blade geometry.

The application reported in sixth chapter answered the following question: how exploiting surrogate-based techniques in optimization problems concerning the aerodynamic of truly reversible profile family for axial fans. The Multi-Objective Optimization Problem (MOOP) was based on the NSGA-II optimization algorithm. The optimization has been initially solved by using XFoil software. These results have been successively compared with those obtained by implementing different surrogate models in the MOOP, considering two different optimization frameworks, namely direct fitness replacement (DFR) and indirect fitness replacement (IRF) approach. The work showed how the last cited approach should be preferred determining more reliable results. The IFR ensured the creation of the same of even better Pareto fronts if compared to those obtained by using only XFoil, while the calls to the expensive objective function were reduced of 50%.

Last application, reported in seventh chapter, data driven based strategy to investigate the influence of tip geometry on performance in HPT blade and exploiting the acquired learning generating new profiles. An existing dataset of optimized tip has been exploited using unsupervised learning algorithm to generate a new statistical shape representation of tip surface that allowed overcoming the limits of original topological representation. The new representation decreased the amount of geometric variables needed to define the surface and improved the genetic operator effectiveness reducing the generation of infeasible shapes. Finally, aiming to generate new profiles with alternative routes to CFD-based optimization, two meta-models have been developed and used to generate five new profiles to overcame the previous best efficiency datum and to control the secondary flow keeping high efficiency value.

Acknowledgements

After more than three years my PhD is coming to its conclusion. This experience has been very relevant for me in terms of both professional and personal growth. I want to thank the people who supported me during this period.

Firstly, I would like to thank my supervisor Prof. Alessandro Corsini for being a guide. He gave me valuable advices and supported me throughout all my research period.

I would also like to tank Prof. Sergio Lavagnoli who hosted me at Von Karman Institute for Fluid Dynamics in Bruxelles as visiting PhD. I keep a very pleasant memory of that experience.

I thank my colleagues Giovanni De Libra, Lorenzo Tieghi, Tommaso Bonanni, David Volponi and Valerio Barnabei. In addition to working together, we shared pleasant and fun moments.

Finally, I thank my family who have always supported me. I dedicate this important goal to them.

Dopo oltre tre anni il mio percorso di dottorato sta volgendo al termine. Per me questa esperienza ha significato moltissimo in termini di crescita professionale e personale. Desidero ringraziare le persone che mi hanno accompagnato e supportato durante questo periodo.
Prima di tutto vorrei ringraziare il mio supervisor Prof. Alessandro Corsini per essere stato una guida, consigliandomi ed anche incoraggiandomi lungo tutto il mio percorso di ricerca.
Vorrei inoltre ringraziare il Prof. Sergio Lavagnoli che mi ha ospitato presso il Von Karman Institute di Bruxelles presso il quale ho svolto 6 mesi di ricerca come visiting Phd. Periodo del quale conservo un piacevolissimo ricordo.
Ringrazio i miei colleghi Giovanni De Libra, Lorenzo Tieghi, Tommaso Bonanni, David Volponi e Valerio Barnabei con i quali oltre a lavorare insieme ho condiviso molti momenti divertenti.
Ringrazio infine la mia famiglia, mio Padre, mia Madre e mia Sorella, che da sempre mi supportano giorno dopo giorno; Elisa con la quale stiamo condividendo momenti importanti. A loro dedico questo importante traguardo.

Bibliography

[1] W. Hasperué, "The master algorithm: How the quest for the ultimate learning machine will remake our world", *Journal of Computer Science and Technology*, vol. 15, no. 02, pp. 157–158, 2015.

[2] N. Ketkar *et al.*, *Deep Learning with Python.* Springer, 2017.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[4] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database", *Scientific data*, vol. 3, p. 160 035, 2016.

[5] M. Banko and E. Brill, "Scaling to very very large corpora for natural language disambiguation", in *Proceedings of the 39th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2001, pp. 26–33.

[6] J. Rowley, "The wisdom hierarchy: Representations of the dikw hierarchy", *Journal of information science*, vol. 33, no. 2, pp. 163–180, 2007.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets", in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[8] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics", *Annual Review of Fluid Mechanics*, vol. 52, 2019.

[9] C. Teo, K. Lim, G. Hong, and M. Yeo, "A neural net approach in analyzing photograph in piv", in *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, 1991, pp. 1535–1538.

[10] I. Grant and X Pan, "An investigation of the performance of multi layer, neural networks applied to the analysis of piv images", *Experiments in Fluids*, vol. 19, no. 3, pp. 159–166, 1995.

[11] C. M. Bishop and G. D. James, "Analysis of multiphase flows using dual-energy gamma densitometry and neural networks", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 327, no. 2-3, pp. 580–593, 1993.

[12] S. L. Brunton and B. R. Noack, "Closed-loop turbulence control: Progress and challenges", *Applied Mechanics Reviews*, vol. 67, no. 5, p. 050 801, 2015.

[13] C. W. Rowley and S. T. Dawson, "Model reduction for flow analysis and control", *Annual Review of Fluid Mechanics*, vol. 49, pp. 387–417, 2017.

[14] K. Taira, S. L. Brunton, S. T. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley, "Modal analysis of fluid flows: An overview", *Aiaa Journal*, pp. 4013–4041, 2017.

[15] J.-L. Bourguignon, J. Tropp, A. Sharma, and B. McKeon, "Compact representation of wall-bounded turbulence using compressive sampling", *Physics of Fluids*, vol. 26, no. 1, p. 015 109, 2014.

[16] E. Kaiser, B. R. Noack, L. Cordier, A. Spohn, M. Segond, M. Abel, G. Daviller, J. Östh, S. Krajnović, and R. K. Niven, "Cluster-based reduced-order modelling of a mixing layer", *Journal of Fluid Mechanics*, vol. 754, pp. 365–414, 2014.

[17] B. Colvert, M. Alsalman, and E. Kanso, "Classifying vortex wakes using neural networks", *Bioinspiration & biomimetics*, vol. 13, no. 2, p. 025 003, 2018.

[18] J.-X. Wang, J.-L. Wu, and H. Xiao, "Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data", *Physical Review Fluids*, vol. 2, no. 3, p. 034 603, 2017.

[19] W. Hou, D. Darakananda, and J. Eldredge, "Machine learning based detection of flow disturbances using surface pressure measurements", in *AIAA Scitech 2019 Forum*, 2019, p. 1148.

[20] K. Giannakoglou, D. Papadimitriou, and I. Kampolis, "Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels", *Computer methods in applied mechanics and engineering*, vol. 195, no. 44-47, pp. 6312–6329, 2006.

[21] M. Gazzola, W. M. Van Rees, and P. Koumoutsakos, "C-start: Optimal start of larval fish", *Journal of Fluid Mechanics*, vol. 698, pp. 5–18, 2012.

[22] B. Strom, S. L. Brunton, and B. Polagye, "Intracycle angular velocity control of cross-flow turbines", *Nature Energy*, vol. 2, no. 8, p. 17 103, 2017.

[23] C. Lee, J. Kim, D. Babcock, and R. Goodman, "Application of neural networks to turbulence control for drag reduction", *Physics of Fluids*, vol. 9, no. 6, pp. 1740–1747, 1997.

[24] S. L. Dixon and C. Hall, *Fluid mechanics and thermodynamics of turbomachinery.* Butterworth-Heinemann, 2013.

[25] M. Drela, "Xfoil: An analysis and design system for low reynolds number airfoils", in *Low Reynolds number aerodynamics*, Springer, 1989, pp. 1–12.

[26] R. I. Lewis, *Turbomachinery performance analysis.* Butterworth-Heinemann, 1996.

[27] T. Bonanni, "Metamodel based design optimization in industrial turbomachinery", PhD thesis, Sapienza Università DI Roma, 2018.

[28] S. Smith, "A simple correlation of turbine efficiency", *The Aeronautical Journal*, vol. 69, no. 655, pp. 467–470, 1965.

[29] O. Balje, *Turbomachines—a guide to design selection and theory*, 1981.

[30] I. H. Abbott and A. E. Von Doenhoff, *Theory of wing sections: including a summary of airfoil data.* Courier Corporation, 2012.

[31] G. L. Mellor, "The aerodynamic performance of axial compressor cascades with application to machine design", PhD thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, 1957.

[32] R. A. Wallis, *Axial Flow Fans: design and practice.* Academic Press, 2014.

[33] A. Howell, "The present basis of axial flow compressor design", *Part I: Cascade Theory and performance. R & M*, vol. 2095, 1942.

[34] S. Lieblein, F. C. Schwenk, and R. L. Broderick, "Diffusion factor for estimating losses and limiting blade loadings in axial-flow-compressor blade elements", NACA Cleveland, Ohio, LEWIS FLIGHT, Tech. Rep., 1953.

[35] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering", *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[36] A. I. Forrester and A. J. Keane, “Recent advances in surrogate-based optimization”, *Progress in aerospace sciences*, vol. 45, no. 1-3, pp. 50–79, 2009.

[37] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, “Surrogate-based analysis and optimization”, *Progress in aerospace sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii”, *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[39] A. Arias-Montano, C. A. C. Coello, and E. Mezura-Montes, “Multi-objective airfoil shape optimization using a multiple-surrogate approach”, in *2012 IEEE Congress on Evolutionary Computation*, IEEE, 2012, pp. 1–8.

[40] A. Samad and K.-Y. Kim, “Shape optimization of an axial compressor blade by multi-objective genetic algorithm”, *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, vol. 222, no. 6, pp. 599–611, 2008.

[41] Y. Tenne, “Initial sampling methods in metamodel-assisted optimization”, *Engineering with Computers*, vol. 31, no. 4, pp. 661–680, 2015.

[42] R. Jin, W. Chen, and A. Sudjianto, “An efficient algorithm for constructing optimal design of computer experiments”, in *ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers Digital Collection, 2008, pp. 545–554.

[43] A. Díaz-Manríquez, G. Toscano, J. H. Barron-Zambrano, and E. Tello-Leal, “A review of surrogate assisted multiobjective evolutionary algorithms”, *Computational intelligence and neuroscience*, vol. 2016, 2016.

[44] K. Bamberger and T. Carolus, “Performance prediction of axial fans by cfd-trained meta-models”, in *ASME Turbo Expo 2014: Turbine Technical Conference and Exposition*, American Society of Mechanical Engineers, 2014, V01AT10A028–V01AT10A028.

[45] L. Gonzalez, J. Periaux, K. Srinivas, and E. Whitney, “A generic framework for the design optimisation of multidisciplinary uav intelligent systems using evolutionary computing”, in *44th AIAA Aerospace Sciences Meeting and Exhibit*, 2006, p. 1475.

[46] J. Kim, J. Choi, A Husain, and K. Kim, “Multi-objective optimization of a centrifugal compressor impeller through evolutionary algorithms”, *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, vol. 224, no. 5, pp. 711–721, 2010.

[47] T. Goel, R. Vaidyanathan, R. T. Haftka, W. Shyy, N. V. Queipo, and K. Tucker, “Response surface approximation of pareto optimal front in multi-objective optimization”, *Computer methods in applied mechanics and engineering*, vol. 196, no. 4-6, pp. 879–893, 2007.

[48] S. F. Adra, I. Griffin, and P. J. Fleming, “An informed convergence accelerator for evolutionary multiobjective optimiser”, in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ACM, 2007, pp. 734–740.

[49] T. Goel, R. T. Haftka, W. Shyy, and N. V. Queipo, “Ensemble of surrogates”, *Structural and Multidisciplinary Optimization*, vol. 33, no. 3, pp. 199–216, 2007.

[50] S. J. Thorpe, R. J. Miller, S. Yoshino, R. W. Ainsworth, and N. W. Harvey, “The effect of work processes on the casing heat transfer of a transonic turbine”, *Journal of Turbomachinery*, vol. 129, no. 1, pp. 84–91, 2007.

[51] R. S. Bunker, “Axial turbine blade tips: Function, design, and durability”, *Journal of propulsion and power*, vol. 22, no. 2, pp. 271–285, 2006.

[52] V. Shyam and A. Ameri, “Comparison of various supersonic turbine tip designs to minimize aerodynamic loss and tip heating”, 2012.

[53] Q Zhang and L He, “Tip-shaping for hp turbine blade aerothermal performance management”, *Journal of Turbomachinery*, vol. 135, no. 5, p. 051 025, 2013.

[54] G. Angelini, T. Bonanni, A. Corsini, G. Delibra, L. Tieghi, and D. Volponi, “On surrogate-based optimization of truly reversible blade profiles for axial fans”, *Designs*, vol. 2, no. 2, p. 19, 2018.

[55] G. I. Rozvany, “A critical review of established methods of structural topology optimization”, *Structural and multidisciplinary optimization*, vol. 37, no. 3, pp. 217–237, 2009.

[56] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, “Active shape models-their training and application”, *Computer vision and image understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[57] T. Heimann and H.-P. Meinzer, “Statistical shape models for 3d medical image segmentation: A review”, *Medical image analysis*, vol. 13, no. 4, pp. 543–563, 2009.

[58] C De Maesschalck, S Lavagnoli, G Paniagua, T Verstraete, R Olive, and P Picot, “Heterogeneous optimization strategies for carved and squealer-like turbine blade tips”, *Journal of Turbomachinery*, vol. 138, no. 12, p. 121 011, 2016.

[59] J. H. Friedman, “Greedy function approximation: A gradient boosting machine”, *Annals of statistics*, pp. 1189–1232, 2001.

[60] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, “Tensorflow: A system for large-scale machine learning”, in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[61] J Gauss, “Combinationis observationum erroribus minimis obnoxiae”, *Gottingen: University of Gottingen*, 1825.

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

[63] G. Louppe, “Understanding random forests: From theory to practice”, *arXiv preprint arXiv:1407.7502*, 2014.

[64] T. Dietterich, “Overfitting and undercomputing in machine learning”, *ACM computing surveys*, vol. 27, no. 3, pp. 326–327, 1995.

[65] R. Genuer, “Variance reduction in purely random forests”, *Journal of Nonparametric Statistics*, vol. 24, no. 3, pp. 543–562, 2012.

[66] G. Biau and E. Scornet, “A random forest guided tour”, *Test*, vol. 25, no. 2, pp. 197–227, 2016.

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

[68] R. Caruana, N. Karampatziakis, and A. Yessenalina, “An empirical evaluation of supervised learning in high dimensions”, in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 96–103.

[69] J. O. Ogutu, H.-P. Piepho, and T. Schulz-Streeck, “A comparison of random forests, boosting and support vector machines for genomic selection”, in *BMC proceedings*, BioMed Central, vol. 5, 2011, S11.

[70] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[71] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, *arXiv preprint arXiv:1412.6980*, 2014.

[72] A. Karpathy *et al.*, “Cs231n convolutional neural networks for visual recognition”, *Neural networks*, vol. 1, 2016.

[73] S. Ruder, “An overview of gradient descent optimization algorithms”, *arXiv preprint arXiv:1609.04747*, 2016.

[74] D. C. Liu and J. Nocedal, “On the limited memory bfgs method for large scale optimization”, *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[75] A. Maćkiewicz and W. Ratajczak, “Principal components analysis (pca)”, *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, 1993.

[76] H. Abdi and L. J. Williams, “Principal component analysis”, *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[77] W Wu, D. Massart, and S De Jong, “The kernel pca algorithms for wide data. part i: Theory and algorithms”, *Chemometrics and Intelligent Laboratory Systems*, vol. 36, no. 2, pp. 165–172, 1997.

[78] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis”, *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[79] H. Abdi, “Partial least squares regression and projection on latent structure regression (pls regression)”, *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 1, pp. 97–106, 2010.

[80] S. Wold, M. Sjöström, and L. Eriksson, “Partial least squares projections to latent structures (pls) in chemistry”, *Encyclopedia of computational chemistry*, vol. 3, 2002.

[81] R. Rosipal and N. 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.

[82] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm”, *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[83] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, “Scalable k-means++”, *Proceedings of the VLDB Endowment*, vol. 5, no. 7, pp. 622–633, 2012.

[84] D. Arthur and S. Vassilvitskii, “K-means++: The advantages of careful seeding”, in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[85] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise.”, in *Kdd*, vol. 96, 1996, pp. 226–231.

[86] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning.* Springer, 2013, vol. 112.

[87] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications.* Springer, 2008, vol. 207.

[88] H. Liu and H. Motoda, *Feature extraction, construction and selection: A data mining perspective.* Springer Science & Business Media, 1998, vol. 453.

[89] J. W. Osborne, A. B. Costello, and J. T. Kellow, “Best practices in exploratory factor analysis”, *Best practices in quantitative methods*, pp. 86–99, 2008.

[90] J. W. Osborne, “Improving your data transformations: Applying the box-cox transformation”, *Practical Assessment, Research & Evaluation*, vol. 15, no. 12, pp. 1–9, 2010.

[91] G. E. Box and D. R. Cox, “An analysis of transformations”, *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 26, no. 2, pp. 211–243, 1964.

[92] I.-K. Yeo and R. A. Johnson, “A new family of power transformations to improve normality or symmetry”, *Biometrika*, vol. 87, no. 4, pp. 954–959, 2000.

[93] D. C. Montgomery, “Design and analysis of experiments. ed”, *John Wiley & Sons*, vol. 52, pp. 218–286, 2001.

[94] D Wilkie, “The design of experiments for engineering studies with particular reference to thermofluids”, *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, vol. 205, no. 1, pp. 67–76, 1991.

[95] G. E. Box, J. S. Hunter, and W. G. Hunter, “Statistics for experimenters”, in *Wiley Series in Probability and Statistics*, Wiley Hoboken, NJ, USA, 2005.

[96] C. J. Greenshields, “Openfoam user guide”, *OpenFOAM Foundation Ltd, version*, vol. 3, no. 1, e2888, 2015.

[97] P. Spalart and S. Allmaras, “A one-equation turbulence model for aerodynamic flows”, in *30th aerospace sciences meeting and exhibit*, 1992, p. 439.

[98] G. Angelini, T. Bonanni, A. Corsini, G. Delibra, L. Tieghi, and D. Volponi, “A meta-model for aerodynamic properties of a reversible profile in cascade with variable stagger and solidity”, in *ASME Turbo Expo 2018: Turbomachinery Technical Conference and Exposition*, American Society of Mechanical Engineers Digital Collection, 2018.

[99] D. Spalding, “A single formula for the “law of the wall””, *Journal of Applied Mechanics*, vol. 28, no. 3, pp. 455–458, 1961.

[100] A. Sheard and K Daneshkhah, “The conceptual design of high pressure reversible axial tunnel ventilation fans”, *Advances in Acoustics and Vibration*, vol. 2012, 2012.

[101] A. Bolton, “Installation effects in fan systems”, *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, vol. 204, no. 3, pp. 201–215, 1990.

[102] W. Cory, *Fans and ventilation: a practical guide.* Elsevier, 2010.

[103] L. J. Herrig, J. C. Emery, and J. R. Erwin, “Systematic two-dimensional cascade tests of naca 65-series compressor blades at low speeds”, 1957.

[104] S. Lieblein and R. H. Ackley, “Secondary flows in annular cascades and effects on flow in inlet guide vanes”, 1951.

[105] T. E. Oliphant, “Python for scientific computing”, *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10–20, 2007.

[106] H.-Y. Wu, S. Yang, F. Liu, and H.-M. Tsai, “Comparisons of three geometric representations of airfoils for aerodynamic optimization”, in *16th AIAA computational fluid dynamics conference*, 2003, p. 4095.

[107] B. Kulfan and J. Bussoletti, “" fundamental" parameteric geometry representations for aircraft component shapes”, in *11th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2006, p. 6948.

[108] J. A. Samareh, “Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization”, *AIAA journal*, vol. 39, no. 5, pp. 877–884, 2001.

[109] H. Sobieczky, K Fujii, and G. Dulikravich, “Notes on numerical fluid mechanics, vol. 68, vieweg verlag”,

[110] P. J. Schneider, “An algorithm for automatically fitting digitized curves”, in *Graphics gems*, Academic Press Professional, Inc., 1990, pp. 612–626.

[111] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, “Convergence properties of the nelder–mead simplex method in low dimensions”, *SIAM Journal on optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[112] B. Daly, *Woods practical guide to fan engineering.* Woods of Colchester Limited, 1978.

[113] R. Jin, W. Chen, and T. W. Simpson, “Comparative studies of metamodelling techniques under multiple modelling criteria”, *Structural and multidisciplinary optimization*, vol. 23, no. 1, pp. 1–13, 2001.

[114] R. Jin, X. Du, and W. Chen, “The use of metamodeling techniques for optimization under uncertainty”, *Structural and Multidisciplinary Optimization*, vol. 25, no. 2, pp. 99–116, 2003.

[115] T. Okabe, Y. Jin, and B. Sendhoff, “A critical survey of performance indices for multi-objective optimisation”, in *The 2003 Congress on Evolutionary Computation, 2003. CEC’03.*, IEEE, vol. 2, 2003, pp. 878–885.

[116] H. S. Chung and J. Alonso, “Design of a low-boom supersonic business jet using cokriging approximation models”, in *9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization*, 2002, p. 5598.

[117] M. Rosenblatt, “Remarks on some nonparametric estimates of a density function”, *The Annals of Mathematical Statistics*, pp. 832–837, 1956.

[118] E. Parzen, “On estimation of a probability density function and mode”, *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[119] P. Bloomfield and W. L. Steiger, *Least absolute deviations: theory, applications, and algorithms.* Springer, 1983.

[120] T. M. Kodinariya and P. R. Makwana, “Review on determining number of cluster in k-means clustering”, *International Journal*, vol. 1, no. 6, pp. 90–95, 2013.