Development of a modular Knowledge-Discovery Framework based on Machine Learning for the interdisciplinary analysis of complex phenomena in the context of GDI combustion processes

Zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)
angenommene

DISSERTATION

von

M. Eng. Massimiliano Botticelli

Tag der mündlichen Prüfung: 30.01.2023

Hauptreferentin: Prof. Dr. Dr.-Ing. Dr. h. c. Jivka Ovtcharova
Korreferent: Prof. Dr. sc. techn. Thomas Koch

When you know a thing,
to hold that you know it;
when you do not know a thing,
to allow that you do not know it;
this is knowledge

The Analects, Confucius

Acknowledgement

This work has been realized during my employment at Robert Bosch GmbH in the business unit Powertrain Solutions (PS) in Schwieberdingen in collaboration with the Institute for Information Management in Engineering (IMI) of the Karlsruhe Institute of Technology (KIT).

First of all, I would like to express my gratitude to Prof. Dr. Dr.-Ing. Dr. h. c. Jivka Ovtcharova not only for supervising but also for truly believing and supporting my thesis. The several suggestions and comments during our discussions have been without exception remarkably valuable and constructive. I would like to extend my thanks to Prof. Dr. sc. techn. Thomas Koch for accepting the co-referee role of the examination and for reviewing this work.

I could not have undertaken this journey without my two industrial supervisors Dr.-Ing. Paul Jochmann and Dr.-Ing. Robin Hellmann. I am extremely grateful for their professional and personal support at every stage of the thesis and for the sincere, open and reliable collaboration developed and strengthened during the years. I would like to deeply thank Dr.-Ing. Karl Georg Stapf and Dr.-Ing. Erik Schünemann for believing in me and letting me join their team. Furthermore, I would like to thank the colleagues of the departments PS/EPN and PS/EPS for the multiple discussions and pleasant working environment. Next, I would like to thank the Ph.D. community at Bosch and the colleagues at the IMI for the moral and technical support during the years.

Finally, I am extremely grateful to my family for their endless support in letting me achieve my dreams by profoundly believing in me as well as motivating me constantly.

Abstract

The physical and chemical phenomena occurring before, during and after the combustion in Gasoline Direct Injection (GDI) engines are complex and include multiple interactions between liquids, gases and the surrounding materials. In the past years, several simulation tools and measurement techniques have been developed in order to understand and optimize the components involved in the engine combustion processes. However, the possibility to explore the whole design space is limited by the significant efforts required to generate and to evaluate the non-linear and multidimensional results. The aim of this work is to develop and validate a knowledge discovery framework able to analyze the data produced in the GDI context through machine learning methods. These procedures are able to explore and exploit the investigated design spaces based on a limited number of observations, discovering connections and correlations in complex phenomena. Furthermore, costly and time consuming evaluations can be substituted by fast and accurate predictions.

After the introduction of the main data characteristics available in this context, the knowledge discovery framework is presented highlighting its modular and interdisciplinary nature. The core of the framework is a parameter-free, fast and dynamic data-driven model selection, which is tailored for the GDI heterogeneous datasets. Its potential is demonstrated on the analysis of numerical and experimental investigations regarding nozzles and engines. In particular, the non-linear influences of the design parameters on inflow and spray characteristics as well as on emissions are extracted from the data. Furthermore, new designs able to achieve predefined objectives and performance are identified based on machine learning predictions. The extracted knowledge is finally validated with the domain expertise, revealing the potential and the limitations of this novel approach.

Zusammenfassung

Die physikalischen und chemischen Phänomene vor, während und nach der Verbrennung in Motoren mit Benzindirekteinspritzung (BDE) sind komplex und umfassen unterschiedliche Wechselwirkungen zwischen Flüssigkeiten, Gasen und der umgebenden Brennraumwand. In den letzten Jahren wurden verschiedene Simulationstools und Messtechniken entwickelt, um die an den Verbrennungsprozessen beteiligten Komponenten zu bewerten und zu optimieren. Die Möglichkeit, den gesamten Gestaltungsraum zu erkunden, ist jedoch durch den hohen Aufwand zur Generierung und zur Analyse der nichtlinearen und multidimensionalen Ergebnisse begrenzt. Das Ziel dieser Arbeit ist die Entwicklung und Validierung eines Datenanalysewerkzeugs zur Erkenntnisgewinnung. Im Rahmen dieser Arbeit wird der gesamte Prozess als auch das Werkzeug als "Knowledge-Discovery Framework" bezeichnet. Dieses Werkzeug soll in der Lage sein, die im BDE-Kontext erzeugten Daten durch Methoden des maschinellen Lernens zu analysieren. Anhand einer begrenzten Anzahl von Beobachtungen wird damit ermöglicht, die untersuchten Gestaltungsräume zu erkunden sowie Zusammenhänge in den Beobachtungen der komplexen Phänomene schneller zu entdecken. Damit können teure und zeitaufwendige Auswertungen durch schnelle und genaue Vorhersagen ersetzt werden. Nach der Einführung der wichtigsten Datenmerkmale im Bereich der BDE Anwendungen wird das Framework vorgestellt und seine modularen und interdisziplinären Eigenschaften dargestellt. Kern des Frameworks ist eine parameterfreie, schnelle und dynamische datenbasierte Modellauswahl für die BDE-typischen, heterogenen Datensätze. Das Potenzial dieses Ansatzes wird in der Analyse numerischer und experimenteller Untersuchungen an Düsen und Motoren gezeigt. Insbesondere werden die nichtlinearen Einflüsse der Auslegungsparameter auf Einström- und Sprayverhalten sowie auf Emissionen aus den Daten extrahiert. Darüber hinaus werden neue Designs, basierend auf Vorhersagen des maschinellen Lernens identifiziert, welche vordefinierte Ziele und Leistungen erfüllen können. Das extrahierte Wissen wird schließlich mit der Domänenexpertise validiert, wodurch das Potenzial und die Grenzen dieses neuartigen Ansatzes aufgezeigt werden.

Contents

Abbreviations and Symbols

Abbreviations

ABM	Adaptive Basis-function Model
ADE	Automatic Data Extraction
ADT	Automatic Data Transformation
AI	Artificial Inteligence
ANN	Artificial Neural Networks
API	Application Programming Interface
BDC	Bottom Dead Center
CA	Crank Angle
CART	Classification And Regression Trees
CFD	Computational Fluid Dynamics
CLI	Command Line Interface
CPU	Central Processing Unit
CV	Cross Validation
DEAP	Distributed Evolutionary Algorithms in Python
DOE	Design Of Experiments
ETL	Extraction, Transformation, Load
EU	European Union
GA	Genetic Algorithm
GB	GigaByte
GBM	Gradient Boosting Machine
GDI	Gasoline Direct Injection
GUI	Graphical User Interface
HDF5	Hierarchical Data Format 5
HPC	High Performance Computer
HPO	Hyperparameter Optimization
KD	Knowledge Discovery

KDD	Knowledge Discovery in Databases
LOO	Leave-One-Out
MAE	Mean Absolute Error
MSE	Mean Square Error
NSGA	Non-dominated Sorting Genetic Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm II
ODV	Outlier Detection Value
PCA	Principal Component Analysis
RAM	Random-Access Memory
RMSE	Root Mean Square Error
TDC	Top Dead Center
TKE	Turbulent Kinetic Energy
UML	Unified Modeling Language
XGBoost	eXtreme Gradient Boosting

Notations

$1(\cdot)$	Binary Fucntion
$\boldsymbol{a}$	Set of parametric learning coefficients
α	L_1 regularization term
α	Parametric learning coefficient
B	Number of bagging iterations
$\boldsymbol{B}_m$	Partial estimation of parametric learning coefficients
b	Average response in a decision tree region
$\boldsymbol{\beta}$	Set of parametric learning coefficients
β	Parametric learning coefficient
C	Constraint
c	Constant
δ_M	Maximum tree depth for base learners
e_{Err}	Error computed on the test set
ϵ	Learning rate
ε	Random error term
$F_m(\cdot)$	Partial estimation of a function
$f(\cdot)$	General function relating an input to an output
Fr	Friedman dataset
F	Front of solutions in an evolutionary algorithm
$\mathbf{g}_m$	Steepest-descent direction at iteration m

γ	Weight associated to a decision tree region
γ	Parametric learning coefficient
γ	Minimum loss reduction for a leaf
$h(\cdot)$	Basis function in boosting modeling
H	Number of hyperparameter sets
i_t	Improvement at the node t of a decision tree
I_j	Relative influence of the j-variable
J	Decision tree size
K	Fold of a K-fold CV
λ	L_2 regularization term
$L(\cdot)$	Loss function
M	Number of boosting iterations
M	Mode
μ_{Err}	Average error from K-fold CV
μ	Mean
μ	Population size of an evolutionary algorithm
N	Number of hyperparameters
n	Number of observations
ω_{mc}	Minimum weights sum in a leaf
$\mathbf{p}$	Set of hyperparameters
p	Sensitivity for Chebyshev outlier detection
p	Number of explanatory variables
q	Number of response variables
R	Region of a decision tree
R^2	Coefficient of determination
ρ_m	Steepest-descent step length at iteration m
s_d	Observations subsampling
s_{cl}	Feature subsampling at each level
s_{cn}	Feature subsampling at each node
s_{ct}	Feature subsampling at each tree
s	Anomaly score for Isolation Forest
σ	Standard deviation
σ_{Err}	Standard deviation error from K-fold CV
t	Node of a decision tree
T	Decision tree
v_t	Splitting variable at the node t of a decision tree
$\boldsymbol{x_j}$	Observations of the j-th explanatory variable

x_{ij}	Observation i of the explanatory variable j
x_i	Observations i of all the p explanatory variables
$\mathbf{X}$	Collection of explanatory observations
X	General explanatory variable(s)
$\boldsymbol{y_j}$	Observations of the j-th response variable
y_{ij}	Observation i of the response variable j
$\hat{Y}$	Prediction of the response variable(s) Y
$\tilde{y}_i$	Pseudo-responses of a boosting learning method
y_i	Observations i of all the q response variables
$\mathbf{Y}$	Collection of response observations
Y	General reponse variable(s)

Physical Quantities

Most of the physical quantities are reported below without measurement unit because they have been anonymized for the current work.

α	Hole inclination angle	(-)
β	Hole circumferential angle	(-)
Ψ	Spray hole conicity	(-)
γ	Spray angle	(-)
HC	Hydrocarbons	(-)
$IMEP$	Indicated mean effective pressure	(bar)
i_f	Fuel flow rate	(-)
k	Fuel turbulent kinetic energy	(-)
λ	Air-fuel ratio	(-)
λ_{90}	Homogeneity index 90%	(-)
l/d	Spray hole length over diameter	(-)
l_f	Engine throttle valve	(-)
$\dot{m}$	Fuel massflow	(-)
m_l	Liner impinged mass	(-)
m_p	Piston impinged mass	(-)
m_s	Impinged spray mass	(-)
m_{split}	Injection fuel mass	(%)
m_{split_1}	First injection fuel mass	(%)
m_{split_2}	Second injection fuel mass	(%)
n	Engine speed	(rpm)
nl_h	Needle lift height	(-)

NO_x	Nitrogen oxides	(-)
p_p	Fuel plume penetration	(-)
p_s	Fuel spray penetration	(-)
ph_d	Pre-hole diameter	(-)
ph_l	Pre-hole length	(-)
PN	Particulate number	(-)
P_c	Combustion chamber pressure	$(mbar)$
P_f	Fuel pressure	(bar)
r_t	Spray targeting radius	(-)
sh_d	Spray hole diameter	(-)
$sh_{d,o}$	Spray hole outlet diameter	(-)
sh_l	Spray hole length	(-)
SOI	Single start of injection	$(°CA\ b.\ TDC)$
SOI_1	First start of injection	$(°CA\ a.\ TDC)$
SOI_2	Second start of injection	$(°CA\ a.\ TDC)$
st_h	Circumferential step height	(-)
s_w	Spray width	(-)
τ	Spray plume angle	(-)
T_f	Fuel temperature	$(°C)$
T_w	Cooling water temperature	$(°C)$
t_i	Injection time	(ms)
wt_l	Wall thickness	(-)
(x_t, y_t)	Spray targeting coordinates	(-)

Subscripts and Superscripts

$_0$	Constant value
$_C$	Complementary
$_i$	Observation i
$_j$	Region j of a decision tree
$_j$	Variable j
$_m$	Iteration m in boosting modeling
$_S$	Subset
^{f}	Fictive dataset
$^{(l)}$	Lower limit
$^{(u)}$	Upper limit

$prep$	Preprocessed dataset
s	Sampled DOE
$\bar{}$	Mean value
$\hat{}$	Estimation/approximation
tr	Training set

Measurement Units

$°C$	Celsius degree
$°CA\ a.\ TDC$	Crank angle after the TDC
$°CA\ b.\ TDC$	Crank angle before the TDC
rpm	Revolutions per minute

1 Introduction

The desire of reducing mobility climate impact and the demand of high efficiency powertrain solutions is leading the automotive industry to new challenges. The road transport in Europe is responsible for more than 25% of the whole greenhouse gas emissions, which makes it one of the major contributor to the climate change [1]. Since the 1990s, the European Union (EU) has started introducing new regulations aiming the reduction of concentration of pollutants. Fuel quality and emissions standards as well as air quality management plans ensured lower emissions from road transport, despite the higher activities in this sector [2]. To be specific, carbon monoxides CO decreased by 88%, nitrogen oxides NO_x by 60% and solphur oxides SO_x by 99% since the introduction of these regulations [2]. As next target, the EU aims to net-zero greenhouse gas emissions by 2050 [3].

In order to meet these expectations, many resources are invested in the research of alternative fuels and electric solutions. The spread of fully electric and plug-in hybrid cars has increased since their first appearance in the market [4]. Their production as well as their use-costs have become cheaper [4], thus, more competitive with respect to conventional vehicles. Nevertheless, several barriers are still present, e.g. reduced drive range, long recharging time as well as lower availability of infrastructures [5]. In particular, if the energy is not completely retrieved from renewables, the emissions of an electric vehicle throughout its whole life-cycle are not neglectable [4]. In addition, particulate matter coming from wear of tyres, brakes and roads are present as well and the production of electric vehicles is typically more energy-intensive than the conventional ones [4].

The volume shares of Gasoline Direct Injection (GDI) engines are expected to increase with respect to other conventional systems [6] since they are considered a valuable internal combustion engine solution for hybrid applications [7]. For these reasons, many activities are further involved in the research and development of GDI systems.

1.1 Direct Injection System for Gasoline Engines

The GDI system is characterized by the injection of high-pressurized gasoline directly into the combustion chamber, as represented in Figure 1.1. Based on the Otto cycle, the GDI engine converts chemical energy into kinetic energy by burning an air-fuel mixture [8]. This process is achieved with four phases, better known as *four-stroke principle*: *intake*, *compression*, *combustion* and *exhaust*, as represented in Figure 1.2. Following, the four-stroke principle is introduced referring to the components enumerated in Figure 1.1 and in Figure 1.2. During the intake phase, the *intake valve* (1) is opened and the downwards movement of the *piston* (2) allows fresh air to enter the *combustion chamber* (3). Once the piston reached its lower limit called Bottom Dead Center (BDC), the compression phase initiates. The intake valve closes and the piston moves upward going back to its upper limit called Top Dead Center (TDC), compressing the air volume. The *injector* (4) injects the high-pressurized fuel inside the combustion chamber during the intake and compression phases. According to the operating strategy, the injection time and the number of injections may vary. Just before the piston reaches the TDC, the combustion phase starts. The *spark plug* (5) ignites the air-fuel mixture such that the heat and the pressure increase, pushing the piston downward. Shortly before the

Figure 1.1: GDI System [9].

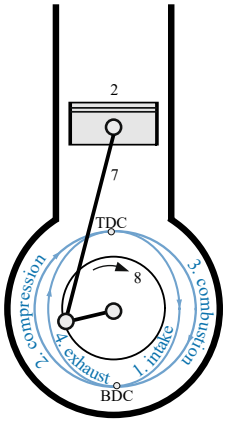


Figure 1.2: Four-Stroke Principle, based on [8].

piston reaches the BDC, the *exhaust valve* (6) opens, letting the exhaust gas leave the combustion chamber. The movement of the piston is converted by the *conrod* (7) into a rotational movement of the *crankshaft* (8). The sectional view of an injector is represented in Figure 1.3.

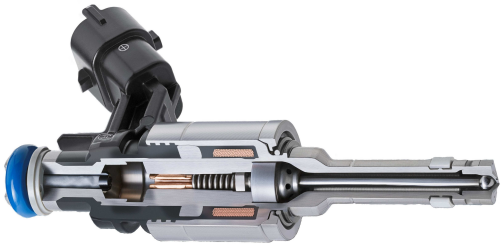


Figure 1.3: Sectional view of a multi-hole high pressure gasoline injector (Bosch HDEV 5.2) [10].

According to [11], the key element for a proper combustion is a good air-fuel mixture formation, which influences emissions as well as efficiency in terms of consumption and performance. Several direct and indirect phenomena are involved in this process, as summarized in Figure 1.4. Beside the ambition to reach the perfect air-fuel homogenization, the achievement of an inflammable mixture at the ignition time independently from engine speed or load is fundamental as well. During the development of a combustion system, the parameters that can be adjusted are typically the geometry of the cylinder and the injector nozzle together with the engine operating points and the fuel characteristics. However, these parameters do not have a direct influence to the the mixture formation. Principally, they can be used to affect spray *penetration* and *impingement*, which have indeed a direct influence to the mixture characteristics. The spray penetration corresponds to the distance between the injector valve seat and the spray plume tip. The impingement represents the amount of fuel droplets impinged on the combustion chamber surfaces. A large penetration may cause high impingement if the fuel does not fully evaporate before reaching a specific surface, as instance the piston head or the cylinder wall. The latter is also known as cylinder *liner*. The impingement may lead to unburned fuel, thus, to a lower air-fuel homogenization and higher emissions. Due to the complexity and high-dimensionality of the interrelations among the

design parameters, a large effort is required to properly define them in order to satisfy given engine and emission specifications.

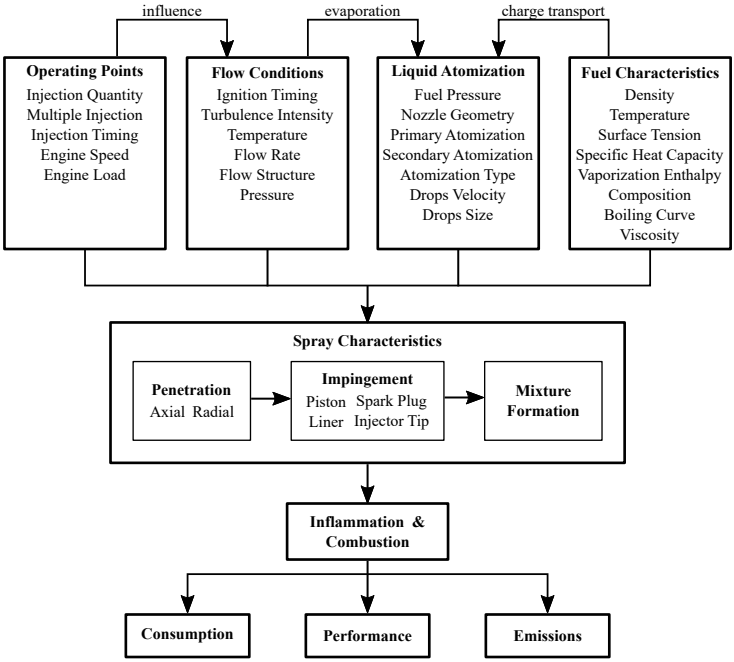


Figure 1.4: GDI parameters affecting engine emissions, performance and consumption.

The desire of better and deeper understanding of the complex GDI system led to a continuous development of tools and techniques in this field. Nowadays, its analysis is performed through complementary studies between physical experiments and numerical investigations [12]. In particular, the analysis can be divided into four parts: injector, spray, combustion chamber and vehicle [13], as depicted in Figure 1.5. Measurements are able to provide a global overview of the entire system. This is achieved in terms of vehicle performance, spray and emission analysis as well as of combustion chamber endoscopy. As the potential of computer aided engineering spread in many fields, it found a valuable application in the combustion development too. Specifically, 3D Computational Fluid Dynamics (CFD) simulations are able to model the complex

fluid-mechanical and thermodynamic processes of the combustion, including the spray system with the injector as a key component. The increase of computational performance enabled meaningful analysis by means of virtual product development. However, the data resulting from GDI investigations are expensive in terms of time and costs. Experiments require costly test benches and prototypes to be built and calibrated. Simulations are run on High Performance Computer (HPC), where a single evaluation may need days to be completed on a large number of parallel Central Processing Units (CPUs) [14]. Although the availability of advanced simulation and experimental tools, the analysis of GDI system is still a challenging task in terms of understanding its several complex, non-linear and highly multidimensional relations.

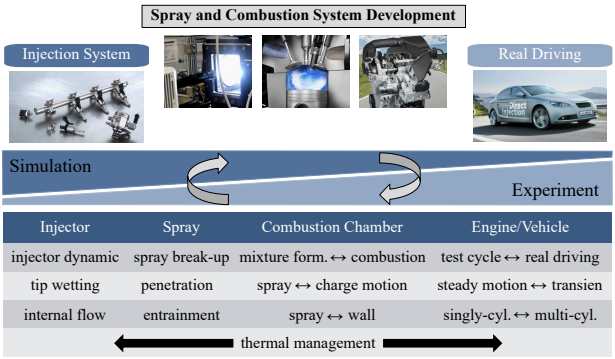


Figure 1.5: Complementary investigations in engine combustion systems [13].

1.2 Motivation and Goals of the Work

The classical engineering development in the GDI context consists in starting from a base geometric layout or engine operating point and through expert knowledge adjust the involved features iteratively until the required objectives are achieved. This procedure corresponds most of the times to a trade-off among several contradictive requirements. Statistical tools are generally adopted in order to analyze data or substitute costly investigations by means of estimated models [15], especially in the GDI development [14, 16]. In particular, input-output relationships are suggested by the domain expertise

in order to model physical phenomena. The resulting estimations are highly interpretable, but the exposure to bias determined by the choice of the model structure may be significant. Therefore, without proper tools, the possibility to explore the whole design space is limited to the large amount of resources requested, including the risk of neglecting some essential underlying relations.

Machine learning techniques based on advanced statistics and data analysis became more popular in the recent years due to the progress of the computational resources [17] and the larger production of data [18]. The application of machine learning supporting the product development is largely spread, as long as the data quality and the data quantity are adequate. In the context of combustion systems, several applications of machine learning can be found. In [19], machine learning is adopted to model emissions and performance of dual fuel high pressure direct injection based on engine operating points. In [20], GDI dynamics in terms of injector opening delay and closing time are modeled in order to improve combustion efficiency. In [21], in-cylinder GDI combustion characteristics are analyzed to predict particulate matter.

The machine learning methods differently from the classical statistical approaches do not require assumptions about the relationships to model. The latter are identified based on the underlying data, leading towards a complete *data-driven development*. Due to the large activities carried out in the GDI development, many data are already available and many more are upcoming with new investigations. These data can be explored and exploited in order to gain a deeper understanding of complex and non-linear combustion phenomena. Furthermore, the machine learning models can support or even substitute challenging numerical and experimental studies, without any additional costs. The possibility to identify how design parameters and operating points influence injection spray shape, fuel consumption or emission is a powerful tool in order to improve the GDI investigations.

In this work, a *knowledge discovery (KD) framework* based on machine learning is developed and presented as the new frontier for a data-driven GDI development. The fundamental characteristic of the framework is its ability to analyze the data independently from their source (numerical or experimental) and from the represented component or system (injector, spray or engine). The proposed framework is not only limited to prediction tasks, but it enables the interpretability of its decisions from a human point-of-view, stepping beyond the concept of black-box *Artificial Inteligence (AI)*. The already strong and well-grounded expertise established in the industry and in the academy can now be deepened and reinforced with additional knowledge explained by machine

learning algorithms. Moreover, with this work it is intended to collect a series of best practices and lessons learned in order to exploit as most the application of machine learning on GDI data.

The structure of the work is presented as follows. In Chapter 2, the required fundamentals are provided. After the introduction of the KD concept, the main steps necessary to enable a data-driven development are explained: from the data preprocessing and the data modeling until the knowledge interpretation. In Chapter 3, the developed KD framework for the GDI context is explained together with its features and structure. Afterwards, in Chapter 4, a novel machine learning model selection algorithm for heterogeneous datasets is proposed. The application of the KD framework on GDI data is reported in Chapter 5 and Chapter 6. Chapter 5 focuses on the injector nozzle: data from numerical and experimental analysis are investigated in order to extract meaningful information regarding nozzle inflow and nozzle spray. Similarly, Chapter 6 analyzes engine data, including spray mixture and emissions based on numerical and experimental data. In Chapter 7, the main limitations and the risks of the machine learning applications are summarized. Finally, in Chapter 8 conclusions are drawn together with the outlook of the work.

2 Fundamentals

In this chapter, the fundamentals of knowledge discovery and machine learning necessary for the comprehension of the presented work are introduced.

2.1 Knowledge Discovery

Since the early 1990s, researchers and practitioners have started addressing the necessity of extracting knowledge from the rapidly growing volumes of data [22]. Traditional techniques like data visualization are limited to the large dimensionality of the datasets and impractical due to human limitations in absorbing details [23]. These approaches are based on manual analysis and interpretation, which are slow, expensive, and highly subjective. The extracted knowledge depends on the human ability to identify useful information based on statistical analysis [22]. Furthermore, the gap between the human processing level and the data availability is increasing exponentially [24]. The current data analysis capabilities are not yet compatible with the actual enormous data production and collection, preventing the access to the whole available knowledge [23–25]. In this context, famous has become the saying "data rich – but information poor" [26]. For these reasons, the interest of scientific communities in the data area has grown in the past years.

2.1.1 Definition

The problem of searching useful information in the data is multidisciplinary and it is approached by several research communities, e.g. statistics or machine learning as part of AI [23, 27]. The whole process starting from the raw data until the knowledge extraction was defined in [22] as knowledge discovery in databases (KDD), reported as the nontrivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data. Nevertheless, in the literature the whole KDD process is often addressed simply as *data mining* or *knowledge discovery (KD)*, especially when the data are not stored in databases. Independently by its naming, the KD process

focuses on data storage, data preparation, human-machine interfaces, results interpretation and visualization as well as how the extracted knowledge would affect the application domain [22]. In addition, the whole process or parts of it are automatized when possible. KD borrows ideas from different technologies, including machine learning, statistics, mathematical optimization, high-performance computing, databases and others [23], as summarized in Fig. 2.1. At the very beginning, the knowledge extraction was mainly focused on business applications. As the technology in sensors, storage and high-performance computing progressed, scientists and engineers started collecting large amount of data from simulations and experiments, demanding novel data analysis approaches as well [23, 28].

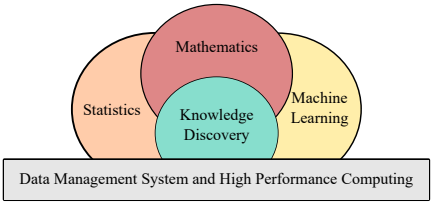


Figure 2.1: Multidisciplinary knowledge discovery.

2.1.2 Taxonomy

A taxonomy of knowledge extraction activities is introduced as follows and it is summarized in Fig. 2.2. Generally, it is possible to distinguish two different types of knowledge extractions: *verification-oriented* and *discovery-oriented* [24]. The knowledge verification consists in the evaluation of hypothesis delivered by an external source, e.g. by an expert. This activity is related to classical statistics approaches including, for instance, the analysis of variance or the goodness of fit test. The knowledge discovery counterpart aims to automatize the identification of new information in the data, sharing most of its methodologies with machine learning. The knowledge discovery can be further split into *descriptive-oriented* and *predictive-oriented* [22, 24]. The descriptive methods focus on the interpretation and the understanding of the processes hidden behind the data. A descriptive model is taken as the reflection of the reality and the knowledge can be extracted directly from it. Predictive approaches focus on the construction of accurate behavioral models, which

are able to predict responses of unseen input data. A predictive model does not have to be necessarily understandable or to reflect the reality, as long as it delivers accurate predictions. Nevertheless, predictive models can support knowledge understanding as well. In this case, the descriptive task is referred as *inference* [29].

Usually, complex predictive approaches tend to fit better the data at the costs of the interpretability, whereas simple models are robuster and more interpretable [22]. A practical example of predictive and predictive-descriptive approaches are the Artificial Neural Networks (ANN) and decision trees respectively. Generally, ANN outperforms decision trees in terms of prediction accuracy, while the latter are able to provide more understandable models [24]. According to the investigated problem, predictive modeling can be divided into *regression* and *classification*. Generally, if the investigated phenomenon is numeric, the predictive learning is known as regression, otherwise if it corresponds to a set of categories, it is called classification.

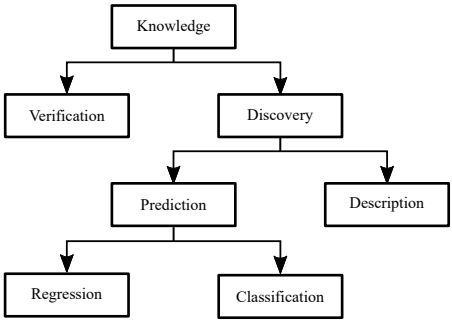


Figure 2.2: Knowledge extraction taxonomy.

Notation

In this section the notation adopted to define a knowledge discovery problem is introduced, which is based on [29, 30].

For instance, consider a set of measurements required to analyze the emissions level in a specific combustion process: the measurements are called *observations* and the known information about the process are the *explanatory*

variables, e.g. temperature and pressure of the combustion chamber. Explanatory variables are referred with different names, such as *predictors*, *independent variables*, *input variables*, *features*, *attributes* or just *variables*. In contrast, the variables indicating the behavior of the phenomena to be analyzed are called *dependent variables*, *output variables*, *response variables* or *ground truths*. In the engine example, the latter represent the emissions level measured at each observation.

The collection of observations of the explanatory and the response variables are indicated with the matrices $\mathbf{X}$ and $\mathbf{Y}$ respectively. The i-th observation of j-th variable is expressed as x_{ij} or y_{ij}. In case of n observations and p variables, the matrix corresponding to the explanatory variables can be written as:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}. \tag{2.1}$$

The rows of the matrices $\mathbf{X}$ and $\mathbf{Y}$ are indicated with x_i and y_i, which represent the i-th observation for all the variables. Considering the example on the explanatory variables in (2.1):

$$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix} \tag{2.2}$$

The columns are indicated with $\boldsymbol{x_j}$ and $\boldsymbol{y_j}$ for explanatory and response variables respectively, corresponding to the set of observations of the j-th variable. Considering the example on the explanatory variables in (2.1):

$$\boldsymbol{x_j} = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{pmatrix} \tag{2.3}$$

Without referring to a specific set of observations, the p independent variables are expressed as $X = (X_1, X_2, \ldots, X_p)$ and q dependent variables as $Y = (Y_1, Y_2, \ldots, Y_q)$. For simplicity, in case of a dataset with a single variable it can be written $X_1 = X$ or $Y_1 = Y$.

2.1.3 Predictive Learning and Inference

The goal of the knowledge discovery in terms of prediction and inference is to extract a function from the observed data, such that the relation between input and output variables can be reproduced. As instance, it can be used to predict new observations or to analyze the original relation. Consider a quantitative response Y and p different predictors $X = (X_1, X_2, \ldots, X_p)$. It is possible to assume that the relationship between X and Y can be written in the following general form [29]:

$$Y = f(X) + \varepsilon, \tag{2.4}$$

where $f(\cdot)$ is an unknown function of $(X_1, X_2, \ldots, X_p)$ representing the systematic information that X provides of Y. The term ε represents a random error with mean equal to zero and independent of X. The process able to estimate $f(\cdot)$ is referred as *statistical learning* or *machine learning* and it can be divided into *supervised* and *unsupervised learning*. Supervised learning requires that to each predictor observation a response observation is associated. In particular, $f(\cdot)$ can be estimated by generalizing the relationships between X and Y from the data. In contrast, unsupervised learning aims to learn specific patterns or behaviors only from the explanatory observations, e.g. clustering the data based on similar characteristics. Typically, predictive tasks are based on supervised learning, while descriptive tasks on unsupervised learning. The current work focuses on inference based on supervised learning.

For most real life phenomena, the set of inputs X is available unlike the output Y, which is difficult to obtain, especially in large quantities. Therefore, supervised modeling aims to predict Y, such that [29]:

$$\hat{Y} = \hat{f}(X) \approx Y \tag{2.5}$$

where $\hat{f}(\cdot)$ is the estimate for $f(\cdot)$ and $\hat{Y}$ the resulting prediction. The observations employed to estimate the function are called *training points*, while the process of function estimation is referred as *model training* or *model fitting*. The accuracy of the estimated function depends on two errors: the *reducible*

error and the *irreducible error* [29]. Consider an estimate $\hat{f}(\cdot)$ and a set of predictors X yielding $\hat{Y} = \hat{f}(X)$. The expected value of the squared difference between the predicted $\hat{Y}$ and the ground truth Y can be demonstrated to be [29]:

$$E(Y - \hat{Y})^2 = \underbrace{[f(X) - \hat{f}(X)]^2}_{\text{Reducible}} + \underbrace{Var(\varepsilon)}_{\text{Irreducible}} . \tag{2.6}$$

The reducible error can be improved by proper modeling techniques. The irreducible error corresponds to the variance associated with the random error ε in (2.4), which may contain information not included in X but still required to predict Y, e.g. unmeasured variables or unmeasurable variations. The focus of data modeling is to minimize the reducible error, while the irreducible error provides an unknown upper bound on the accuracy.

In case of inference, the estimation of the function $f(\cdot)$ in (2.4) is not necessarily adopted to deliver accurate predictions for the response Y: the aim of the inference task is to reveal the relationship between the response variable and each predictor [29]. Specific techniques are able to combine predictive and inference tasks (see Section 2.3 and Section 2.4).

2.1.4 Knowledge Discovery Process

The KD process is step-wise, iterative and interactive in terms of including feedbacks from experts of the analyzed domain. Based on the results of each step, some previous operations may have to be repeated or next steps may not be required anymore. The single steps can be found with different names or order in the literature, but the main structure remains the same [22–24, 27, 31]. The process starts with the definition of the discovery goals and terminates with the implementation of the discovered knowledge. The KD process considered for this work is reported in Fig. 2.3 and its steps are briefly explained as follows. The dashed arrows in Fig. 2.3 represent the interactivity of the process, which allows to return to previous steps or skip next ones.

Problem Understanding. A solid domain expertise is fundamental in order to validate the extracted information and to lead the investigation in the correct direction. The goals of the analysis are defined in this step based on relevant prior knowledge.

Data Selection and Exploration. The extracted knowledge is based on the underlying observations. Therefore, if important attributes or datapoints are

missing, the final results may not reflect the real process behind the data. This step consists in selecting and screening already available data or obtaining new observations, in order to fulfill the intended analysis.

Data Preprocessing. The reliability of the raw data is enhanced and these are transformed in a proper format for the modeling algorithm. According to the data, the preprocessing methods can be completely ignored or be the main activity of the process. Typically, this is a domain-specific step.

Data Modeling. First, an appropriate family of algorithms for the required task is chosen, i.e. predictive or descriptive. Second, specific modeling meta-parameters, the so-called *hyperparameters*, are selected and validated. Finally, the model is trained.

Knowledge Interpretation and Integration. The extracted knowledge in terms of predictive models, rules or descriptions is analyzed, validated and documented for further usage. The discovered information can be reported to experts and integrated with previous knowledge.

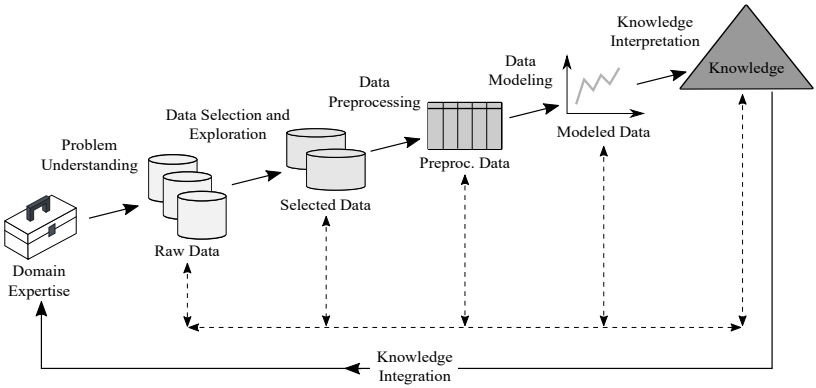


Figure 2.3: Knowledge discovery process.

In the next sections, more details regarding the state-of-the-art for data preprocessing and data modeling are given.

2.2 Data Preprocessing

The data preprocessing includes all the stages to prepare the raw data for the modeling algorithms. Real world data are typically heterogeneous due to the complexity of the sources and of the processes that generated them. Therefore, every dataset requires a tailored procedure able to enhance the reliability of the knowledge extraction and ensure the quality of the data. The latter can be described in terms of accuracy, currentness, completeness and consistency [32]. These requirements may not be fulfilled considering real-world data, which are often incomplete and inconsistent [31]. Some examples are: the integration of data coming from different sources may generate missing or redundant information; measurements may be affected by factors not tracked in the data; errors during the collection of the data may also contribute to the overall data quality.

Generally, preprocessing algorithms are divided into four categories: *data integration*, *data cleaning*, *data transformation* and *data reduction*. The application and the order of these processes are not strict and immutable. In addition, the knowledge extraction starts with the data preprocessing: some properties of the process generating the data may be highlighted during the data preparation. For instance, how the independent variables are related to each others or in which conditions invalid or redundant information are present. As follows, the main data preprocessing approaches are introduced.

2.2.1 Data Integration

The data integration operators allow to map heterogeneous datasets into a uniform one. Raw data may be collected from different sources, e.g. different sensors, tools or procedures, which may be stored in different formats, including inconsistent names or measurement units. Similar data may be collected from several sources, generating redundant information. Additionally, in case the structure of the datasets to be integrated is not equal, missing values or attributes may also be induced. Generally, the data integration is a manual step and it is driven by the domain expertise. Nevertheless, once the set of operations for a specific integration problem is fixed, software routines can be defined and executed whenever new observations are available. This would increase the efficienty and the reproducibility of the data integration.

2.2.2 Data Cleaning

Inaccuracies in the data may affect the reliability of the discovered knowledge. In the literature, datasets including missing data, noise and inconsistencies are denominated *dirty data* [33]. The main reasons for these issues are often associated to the source and the collection of the data [24, 27]. However, the enhancement of the data generation and collection processes may be costly or even impossible. Therefore, data cleaning tools support the preparation of the data in order to achieve a reasonable quality to enable the knowledge extraction.

Generally, the domain knowledge has a key role in the data cleaning: being able to define error boundaries or understanding the reason of missing data can support the identification and the correction of the issues. Often, data cleaning tools are such domain specific that are addressed as a "black art" [24]. However, when the complexity and high dimensionality of the data cannot be handled anymore by the domain knowledge, *data-centric* methods based on statistical behaviors are applied [24, 27]. Nevertheless, any data inconsistency has to be analyzed manually before discarding any information: neglecting correct data because they were identified as dirty data means losing information. Procedures to handle noisy and missing data are introduced as follows.

Outlier detection

Noise in the data is typically associated to the presence of *outliers*. A formal definition of outlier was given in [34] as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism". The different mechanism can be intended as an error or as a novelty, i.e. something correct from the perspective of the data generation, but not expected and different from the rest of the collected observations. For this reason, outliers are often referred as *anomalies* or *exceptions*. In Figure 2.4 an example of outlier is reported.

Anomalies, to be considered as such, have to cover a small percentage of the whole portion of the data. Therefore, outlier detection methods are mostly unsupervised algorithms (see Section 2.1.3) and they cannot be validated, unless domain knowledge is integrated.

Outliers can be removed, corrected or ignored [31]. Since outliers validation is not always possible, removing potential anomalies may cause the lost of valuable information. Similarly, if the outliers are indeed correct exceptions of

the investigated phenomenon, their correction based on the remaining observations would probably generate new unreliable instances. In case removal and correction are not possible, noise-robust modeling algorithms may be adopted. These methods are less sensitive to noisy data and their performance are similar in presence of clean and dirty data (see Section 2.3).

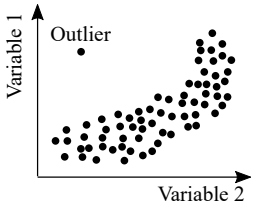


Figure 2.4: Example of an outlier in a two-dimensional dataset.

Two common outlier detection approaches are the *Chebyshev outlier detection* [35], a simple univariate statistical detection, and the *Isolation Forest* [36], a more complex ensemble-based detection for high-dimensional data.

Chebyshev Outlier Detection. The Chebyshev outlier detection, based on the homonym's theorem, computes the outlier boundaries for univariate data empirically and independently from their distribution [35]. This approach assumes that the observations are independent measurements and that the outliers are a relatively small portion.

Assuming p to be an arbitrary value larger than the overall probability of detecting an outlier in the data, i.e. it indicates the detection severity. The upper and lower outlier boundaries, also referred as Outlier Detection Values (ODVs), are defined as:

$$(ODV_l, ODV_u) = (\mu - k * \sigma, \mu + k * \sigma), \tag{2.7}$$

where μ and σ are the mean and the standard deviation of a given variable and $k = 1/\sqrt{p}$. In case of unimodal distribution, an extension of the Chebyshev outlier detection is able to achieve more accurate boundaries estimations [35]. The latter considers the mode M of the data instead of the mean μ, an adjusted standard deviation equal to $\sqrt{\sigma^2 + (M - \mu)^2}$ and $k = 2/(3\sqrt{p})$.

The Chebyshev outlier detection suggests an iterative two-step process to compute the final ODVs. First, a large p_1 is chosen to roughly exclude possible

outliers and to compute a first guess of the ODVs with all the available observations. Second, a smaller p_2 is used to define the final ODVs based on the data within the first interval. The computation of the temporal ODVs avoids that possible outliers inflate the detection interval, masking other anomalies.

Isolation Forest. The Isolation Forest explicitly isolates anomalies instead of profiling normal instances [36]. The main keys of this algorithm are the linear time complexity, the low computational resources demanded and the robustness against high-dimensional datasets as well as against data not containing outliers. It exploits the characteristics of the anomalies to be "few and different": outliers tends to be easier to be isolated with respect to normal observations.

The Isolation Forest is based on *binary trees* in computer science. A binary tree is a data-structure containing a finite number of *nodes* [37]. Every single node has at most two *children nodes*, the *left child* and the *right child*. The first node is called *root node* and a node without successors is referred as *leaf*. An example of binary tree is depicted in Figure 2.5.

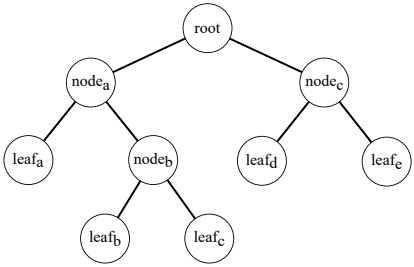


Figure 2.5: Example of a binary tree with three nodes and five leaves.

For a given dataset, the Isolation Forest builds an ensemble of *Isolation Trees* on stochastic sub-samplings of the domain space in terms of observations and features. Every Isolation Tree is a proper binary tree that partitions the datapoints recursively, until every single observation is isolated, i.e. until they all occupy a leaf node. The number of divisions required to isolate the observations is then averaged over the whole ensemble of Isolation Trees. The anomalies are identified as the instances requiring on average less divisions to be isolated, i.e. having on average a short path length from the root to the leaf

node. The Isolation Forest provides for each observation an anomaly score s based on its averaged path length. This can be used as a threshold in order to label an observation as anomaly. Specifically, s can be interpreted as follows:

- $s \to 1$, the instance is definitely an anomaly;
- $s \to 0$, the instance is a normal point;
- $s \to 0.5$ for all the instances, then no distinct anomalies are present.

The operation of stochastic sub-sampling during the building of Isolation Trees allows to soften the effects of two common problems in anomaly detection: *swamping* and *masking* [38, 39]. Swamping refers to the incorrect labeling of normal instances as anomalies, e.g. when normal instances are too close to anomalies. Masking indicates the presence of clusters of anomalies, which increase the difficulty to distinguish them from normal instances. Detecting anomalies on sub-domains allows to highlight the difference between them and the normal points. In addition, each Isolation Tree would be specialized on different observations and features, which makes it suitable to work with high-dimensions. For an efficient Isolation Forest, the authors of the algorithm suggest a small sub-sampling size, e.g. 28 to 256, combined with an ensemble of maximum 100 Isolation Trees [36].

Missing Data

Missing entries in datasets are common when the data collection process is not perfect, for example in case of incorrect measurements to be removed or manual data collection [27, 31]. Missing data can be handled in three different ways [24, 27]: eliminate the records containing at least a missing entry, replace them or adopt data modeling methods able to deal with this issue. The operation of replacing the data is also indicated as *imputation*.

In case many entries are missing for different records, the elimination of the latter would drastically reduce the dataset size. Furthermore, deleting any record in the dataset may lead to information loss. Similarly, the imputation of missing values may introduce additional uncertainty in the dataset. Specific imputation methods are available for some domain applications [40, 41]. Typically, the awareness of the mechanisms which generated the missing data is required. A simple approach consists in substituting the missing entries with the average value of the corresponding attribute [24, 31].

The handling of missing values is a delicate topic depending on the application domain and the available data. This means that out-of-the-box solutions

are not present yet; domain knowledge has to be integrated in order to evaluate possible solutions. For these reasons, the choice of data modeling methods able to work with missing data is typically preferred [27].

2.2.3 Data Transformation

Generally, raw data cannot be processed directly from the modeling algorithms. A proper data structure and format allow and improve the knowledge extraction. The data transformation process contains different approaches according to the available data and the problem to solve. For this reason, a human supervision is often necessary in order to define the required transformation methods [24,31]. The *data normalization*, *data type portability* and *data aggregation* are reported as follows.

Data Normalization

Datasets from different sources or expressed in different scales may not be directly comparable with each other [27]. In addition, some modeling algorithms are sensible to handle attributes on different scales, resulting in emphasizing the data with larger magnitude. Therefore, data normalization is required, especially if it is not possible to convert all the variables to the same unit of measurement. Moreover, in case of sensible data, it may be necessary to anonymize the data in order to share the results publicly.

The most common normalization method is the *Z-score* normalization, also known as *standardization* [27, 31]. Consider x_{ij} the i-th observation of the j-th variable of a dataset and $\boldsymbol{x_j}$ the set of observations of that variable. If μ_j and σ_j are the mean and the standard deviation of $\boldsymbol{x_j}$ respectively, each i-th datapoint x_{ij} can be normalized into $x_{ij,n}$ as:

$$x_{ij,n} = \frac{x_{ij} - \mu_j}{\sigma_j}. \tag{2.8}$$

The resulting dataset would be centered at zero, i.e. the mean corresponds to zero and the standard deviation to one.

A second method is the *Min-Max scaling* [27,31], which maps all the variables in the interval [0, 1]. Considering the datapoint x_{ij} introduced with the previous transformation, the Min-Max scaling can be written as:

$$x_{ij,n} = \frac{x_{ij} - \min_j}{\max_j - \min_j}, \tag{2.9}$$

where $\min_j$ and $\max_j$ are the minimum and the maximum values in $\boldsymbol{x_j}$. This method is not always applicable because the minimum and maximum values of the attributes may not be known or may be influenced by outliers, generating a biased normalization.

Data Type Portability

The presence of heterogeneous data types in a dataset may limit the choice of the modeling algorithm, since not all approaches can manage mixed data. Additionally, it may be necessary to enhance some characteristics of the phenomena by converting the data into a proper format. For example, categorical data can be converted to numerical data and vice versa. Another common conversion is from continuous series to discrete sequences or scalar values [27]. Even though the data conversion is in some cases essential, the representation accuracy and expressiveness of the data may be compromised [27]. For instance, spacial or time series may be averaged in order to represent a global behavior. In this case, the small local variations are lost but the dataset results more adequate for the analysis.

Data Aggregation

The data aggregation consists in enhancing the information contained in the dataset by applying mathematical formulas on its attributes.

Let $X_S = (X_1, X_2, \ldots, X_s)$ be a subset of the variables of a dataset. A new variable X' can be estimated with a transformation function $f_t(\cdot)$ of the variables in X_S as:

$$X' = f_t(X_1, X_2, \ldots, X_s). \tag{2.10}$$

The function $f_t(\cdot)$ can be of any kind and it is typically defined according to specific domain knowledge. For example, consider (X_1, X_3) and (X_2, X_4) to be

the coordinates representing the position of two objects. The distance between the two objects can be computed based on the Pythagorean theorem, such as:

$$X' = \sqrt{(X_1 - X_2)^2 + (X_3 - X_4)^2} \tag{2.11}$$

In case the domain knowledge required to select the proper transformation is not available, it is possible to estimate the transformation function $f_t(\cdot)$ using a brute force approach [42].

2.2.4 Data Reduction

Data reduction methods are divided into *feature* and *instance reduction*. These aim to decrease the dataset size in terms of attributes and observations without information loss. Differently, data aggregation creates a new representation of the data to enhance them.

Data reduction improves accuracy, computational performance, interpretability and visualization [24, 27, 31]. Irrelevant and redundant information may generate noise for the modeling algorithms, affecting the robustness and the performance of the knowledge extraction. Therefore, it is preferred to work with compact datasets including only essential information. Moreover, in order to estimate a function with a given level of accuracy, the number of required observations grows exponentially with the number of significant features in the dataset. This phenomenon is known as the *curse of dimensionality* [43]: an extensive set of features corresponds to a large search space where a proper model can be identified.

In case of new available observations for an already preprocessed dataset, the information to be neglected or combined have to be reevaluated [23]. The available data are usually a partial snapshot of the analyzed phenomena under predefined constraints and requirements. Therefore, specific features or data clusters may be uncritical only for the considered observations. A correct dimension reduction is essential to ensure the integrity of the final extracted knowledge.

Feature Reduction

The feature reduction can be defined as the process of choosing an optimal subset of features according to a certain criterion [44, 45]. The criterion depends on the scope of the application domain [31], but generally it corresponds

to the minimal subset of attributes achieving the best predictive accuracy or requiring lower computational resources.

Let X be the original set of features with cardinality p and $J(\cdot)$ the feature selection criterion, for instance to be maximized. The feature reduction problem can be written as:

$$X' = \arg\max_{Z} J(Z), \text{ with } Z \subseteq X, \tag{2.12}$$

where X' is the optimal set of features. Beside a brute force search, which would require a huge amount of resources for a large p, basic feature reductions start from an empty set and sequentially add new features monitoring their performance [31]. Other common approaches are based on the correlation analysis, neglecting redundant information [31]. Additionally, advanced methods like decision trees can be applied for data reduction as well, indicating the effects of the features on the output [46, 47].

In contrast, considering the response variables, it is common to collect as much information as possible at once, especially if the phenomena generating the data cannot be easily reproduced due to costs or complexity. All the response variables may not be useful for the current investigation, but some information may be required for later examinations. Furthermore, the response variables necessary to analyze a particular phenomenon are in some cases not clear prior to the data generation, which may be indeed the knowledge to be extracted [23].

As following the correlation analysis as feature reduction approach is presented.

Correlation Analysis and Multicollinearity. Redundancy corresponds to the presence of different features explaining completely or partly the same information. This characteristics is also known as *multicollinearity* and can be detected when more predictor variables are correlated [48]. A common correlation index is the *Pearson's coefficient* [49, 50], which measures the amount of association between two variables in terms of linear relationship. The Pearson's correlation of two variables X_1 and X_2 is given by:

$$r_{X_1,X_2} = \frac{cov(X_1, X_2)}{\sigma_{X_1}\sigma_{X_2}} \tag{2.13}$$

where $cov(X_1, X_2)$ is the estimated *covariance* between the two variables, representing their linear relationship. At the denominator, σ_{X_1} and σ_{X_2} correspond to the standard deviations of X_1 and X_2 respectively, necessary to normalize the covariance [51]. The correlation r_{X_1,X_2} assumes values in the range $[-1, 1]$:

- $r_{X_1,X_2} > 0$, the two attributes are positively correlated; i.e. they increase and decrease together;
- $r_{X_1,X_2} < 0$, the two attributes are negatively correlated; i.e. when one increases, the other one decreases;
- $r_{X_1,X_2} = 0$, the two attributes are independent.

The modulus of r_{X_1,X_2} indicates the strength of the relationship. However, the correlation index does not provide any indication regarding the causality of the relationship, i.e. it is not possible to estimate if X_1 causes X_2 or vice versa. In particular, it may be an intermediate variable X_3 that indirectly influences the behavior of both X_1 and X_2 [51, 52]. Even though the correlation index is enough to determine the presence of redundant information [31], it may still lead to *spurious* and *nonsense correlations* [52]. Spurious correlations correspond to apparent relationships between variables, which are indeed due to artifacts in the data, e.g. bias due to outliers or sampling procedures. Nonsense correlations are those relationships which are statistically correct, but accidental [52]. In case of spurious and nonsense correlations, no features can be completely removed without losing information. Methods like the Principal Component Analysis (PCA) can be applied in order to represent the data in a lower multidimensional space without information loss, where all the dimensions are independent from each other [53].

The correlation analysis is typically performed with a so-called *correlation matrix*: on the two axis are reported all the input variables; the content of every cell corresponds to the Pearson's correlation coefficient computed on the two variables identifying the cell.

Multicollinearity does not influence the predictive accuracy of the models; however, their interpretability may be affected [48]. Consider two input features X_1 and X_2 and the response variable Y. A simple estimation $\hat{Y}$ of Y can be written as:

$$\hat{Y} = \alpha X_1 + \beta X_2, \tag{2.14}$$

where α and β are the coefficients to be estimated given a set of observations. If the two variables X_1 and X_2 depend on each other, e.g. $X_2 = \gamma X_1$, the relation

in (2.14) does not properly represent the reality. A correcter estimation of Y corresponds to:

$$\hat{Y} = (\alpha + \beta\gamma)X_1 = \frac{\alpha + \beta\gamma}{\gamma}X_2. \tag{2.15}$$

In general, the correlation analysis gives an indication about the relationships in the data. These have to be validated with the domain expertise before proceeding with the feature reduction.

Instance Reduction

The reduction of the instances is typically necessary to analyze large datasets [54] or to remove redundant and inconsistent observations [31]. Large datasets can be sub-sampled in order to reduce their size. The fraction of data must however reflects the diversity of the entire observations. The presence of unbalanced datasets has to be taken into account [27, 31]: certain information may not be properly represented in the data because of their rarity, despite their importance for the learning task. In this case, *over-sampling* (sampling more rare observations) or *under-sampling* (sampling less common observations) may help balancing the dataset. Another approach is the *stratified sampling*. Here, the observations are divided into mutually disjointed portions according to predefined criteria able to represent all the diversities in the dataset. The disjointed portions are called *strata*. The new dataset is generated by equally sampling from the different strata [55].

In case the data contain clearly observable and independent trends represented by heterogeneous portions of datapoints, these are modeled separately.

Redundant information have to be handled also in the instances. In general, similar or duplicate observations do not provide any additional information or even worse, they may cause inconsistencies, e.g. if the same set of input parameters generate different output characteristics [31]. In some specific applications, redundant information are intentionally collected in order to evaluate the consistency of the process generating the data. In this case, plausibility checks can be applied to select the correct instances. Otherwise, the duplicated observations can be averaged.

2.3 Data Modeling

Once the raw data have been properly preprocessed, they can modeled by the learning algorithms. As already anticipated in Section 2.1.3, the data modeling aims to determine the estimate function $\hat{f}(\cdot)$ of the real function $f(\cdot)$, which relates the explanatory variables X to the response variable Y. This can be done by minimizing the reducible error in (2.6) based on the available observations. The reducible error is represented by a *loss function* indicated with $L(\cdot)$. The function estimation can be defined as an optimization problem, which search the best estimate function $\hat{f}(\cdot)$, able to minimize the loss function $L(\cdot)$ [47]:

$$\hat{f}(\cdot) = \arg\min_{f(\cdot)} L(Y, f(X)), \tag{2.16}$$

A typical loss function is the *least squares*, which aims to minimize the squared error between the ground truth Y and the function $f(X)$, such that [47]:

$$L(Y, f(X)) = \frac{(Y - f(X))^2}{2}. \tag{2.17}$$

The function estimation can be divided into two categories: *parametric* and *non-parametric* [29, 56].

Parametric Learning

The parametric learning estimates the function $f(\cdot)$ starting from a general assumption of its functional form and then adjusting a set of coefficients to better fit the available observations. Typically, the function is shaped according to the real physical meaning of the investigated relations or by brute force. A simple assumption may be a parametric linear relationship of the p explanatory variables in X, such that:

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p. \tag{2.18}$$

The real function can be estimated by determining the $p + 1$ coefficients indicated as $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)$, such that:

$$\hat{f}(X) = f(X, \hat{\boldsymbol{\beta}}) \approx f(X), \tag{2.19}$$

where $\hat{\boldsymbol{\beta}}$ corresponds to the estimation of the unknown real coefficients $\boldsymbol{\beta}$.

Therefore, the function estimation in (2.16) can be simplified as the estimation of the coefficients minimizing the reducible error:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} L(Y, f(X, \boldsymbol{\beta})). \tag{2.20}$$

The main disadvantage of the parametric learning is that if the form of $f(\cdot)$ is not correctly chosen to match its true shape, the model would not be able to represent the real function. The wrong choice of the parametric form may be done in terms of neglecting important explanatory variables or considering the wrong relationship between the predictors, e.g. simple terms instead of a quadratic ones. The relation described in (2.18) is known as *simple linear regression*, while including exponentiation of the explanatory variables is referred as *polynomial regression*.

Non-parametric Learning

Non-parametric learning extracts the shape of the real $f(\cdot)$ directly from the data. This is helpful in case no assumptions regarding the underlying process that generated the data can be done [57]. Although this is a more flexible approach, a larger number of observations are required to obtain an accurate estimate. Common non-parametric methods are [56]: Support Vector Machine, Gaussian Processes or Boosting Models. In this work the boosting method called eXtreme Gradient Boosting (XGBoost) [58] is adopted, which is derived from the Gradient Boosting Machine (GBM) [47, 59].

2.3.1 Gradient Boosting Machine

In the following section, the Gradient Boosting Machine (GBM) is introduced. First, the concept of boosting is explained. Then, the main mechanisms of the algorithm are reported in terms of gradient descent optimization in the function space. Afterwards, the concepts of regularization, regression trees and bagging are short presented together with the hyperparameters. Finally, the XGBoost is introduced as an improved variation of the original GBM.

Boosting

The boosting approach is based on the Adaptive Basis-function Model (ABM) [56]. Common ABM methods are the Classification And Regression Trees

(CART) [60], the Random Forest [61] and the generalized additive models [62]. For these approaches, the shape of the real function $f(\cdot)$ is learned from the data in terms of aggregation of several *basis functions* $h(\cdot)$, such as:

$$f(X) = \sum_{m=0}^{M} h_m(X) = \sum_{m=0}^{M} \beta_m h(X, \boldsymbol{a}_m), \tag{2.21}$$

where M is an arbitrary number and $(\beta_m, \boldsymbol{a}_m)$ are the parameters of the m-th basis function $h_m(\cdot)$. The basis function is also known as *weak* or *base learner*. The term weak emphasizes its performance mediocrity and the term base indicates a building block of the algorithm [63]. The basis function can be any kind of modeling approach, as long as it performs slightly better than a random guess [63]. Even though this is a non-parametric approach, the basis functions $h(\cdot)$ are indeed parametric with parameters $(\beta_m, \boldsymbol{a}_m)$.

With the term *boosting* is typically indicated a greedy algorithm able to fit iteratively the basis functions $h(\cdot)$ on the given set of observations. To be more specific, the coefficients of the m-th basis function are estimated with a weighted version of the available datapoints: at each iteration m, the algorithm associates a weight to each observation according to how the model is able to predict it correctly using the previous $m - 1$ basis functions [56].

Gradient Descent in Function Space

The boosting method was applied for the first time as a form of gradient descent in the function space in [64], which was then expanded to different loss functions as GBM [47]. The latter estimates the original function $f(\cdot)$ combining different approaches: numerical optimization in the function space, stage-wise additive expansion and steepest descent minimization [47]. As following, the steepest descent is introduced for the estimation of a parametric function. After that, the optimization in the function space is explained in order to connect the stage-wise additive expansion with the steepest descent for the estimation of a non-parametric function.

Steepest Descent. A general approach to solve the optimization problem in (2.20) in order to estimate the coefficients $\boldsymbol{\beta}$ of a parametric function $f(X, \boldsymbol{\beta})$

is using a numerical optimization [47]. The estimated parameters $\hat{\boldsymbol{\beta}}$ can be expressed as [47]:

$$\hat{\boldsymbol{\beta}} = \sum_{m=0}^{M} \boldsymbol{\beta}_m, \tag{2.22}$$

where $\boldsymbol{\beta}_0$ is an initial guess and $\{\boldsymbol{\beta}_m\}_1^M$ are M successive increments based on the estimations achieved with the previous iterations. The increments are also called *steps*, *boosts* or *updates*. The partial estimation of the coefficients achieved at any iteration m is indicated as $\boldsymbol{B}_m$ and it can be expressed as:

$$\boldsymbol{B}_m = \boldsymbol{B}_{m-1} + \boldsymbol{\beta}_m. \tag{2.23}$$

In particular, for $m = M$, then $\boldsymbol{B}_m = \hat{\boldsymbol{\beta}}$.

Each increment can be computed using the *steepest-descent minimization* method [65]. This technique defines the increments $\{\boldsymbol{\beta}_m\}_1^M$ as:

$$\boldsymbol{\beta}_m = -\rho_m \mathbf{g}_m, \tag{2.24}$$

where $-\mathbf{g}_m$ indicates the *steepest-descent direction* corresponding to the gradient of the function to minimize and ρ_m is the step length along the steepest-descent direction such that the objective function decreases. Considering the *loss function* $L(\cdot)$ in (2.20) to be minimized, the steepest-descent direction is defined as [47]:

$$\mathbf{g}_m = \{g_{jm}\} = \left\{ \left[\frac{\partial L(Y, f(X, \boldsymbol{B}))}{\partial \beta_j} \right]_{\boldsymbol{B}=\boldsymbol{B}_{m-1}} \right\}, \tag{2.25}$$

which corresponds to a vector containing the partial derivatives over the j parameters of $\boldsymbol{\beta}$ computed at the partial estimation $\boldsymbol{B}_{m-1}$. The step length ρ_m is computed through the *line search method* as [47]:

$$\rho_m = \arg\min_{\rho} L(Y, f(X, \boldsymbol{B}_{m-1} - \rho \mathbf{g}_m), \tag{2.26}$$

which indicates the length of the step size from the partial estimation $\boldsymbol{B}_{m-1}$ required to minimize $L(\cdot)$.

Optimization in the Function Space. The evaluation of $f(\cdot)$ at each observation can be considered a parameter of the function to model, i.e. in a finite

dataset of n observations there would be $\{f(x_i)\}_1^n$ parameters. In this way, the optimization problem in (2.16) can be solved in the function space with the steepest-descent approach. Considering the empirical loss minimized over the observations $\{x_i\}_1^n$, the function estimation can be written as:

$$\hat{f}(\cdot) = \arg\min_{f(\cdot)} \sum_{i=0}^{n} L(y_i, f(x_i)). \tag{2.27}$$

Expressing the function $f(\cdot)$ as the additive expansion defined in (2.21), the equation (2.27) becomes:

$$\{\beta_m, \mathbf{a}_m\}_1^M = \arg\min_{\{\beta'_m, \mathbf{a}'_m\}_1^M} \sum_{i=1}^{n} L\Big(y_i, \sum_{m=1}^{M} \beta'_m h(x_i, \mathbf{a}'_m)\Big). \tag{2.28}$$

Considering the boosting approach, the parameters $(\beta_m, \mathbf{a}_m)$ for a single increment m can be indicated as:

$$(\beta_m, \mathbf{a}_m) = \arg\min_{\beta, \mathbf{a}} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \beta h(x_i, \mathbf{a})), \tag{2.29}$$

and

$$F_m(X) = F_{m-1}(X) + \beta_m h(X, \mathbf{a}_m) \tag{2.30}$$

indicates the partial estimation of $\hat{f}(X)$ at the boosting step m.

The combination of the steepest-descent approach and the optimization in the function space is explained as follows. Some similarities can be noticed between the expansion in (2.30) and the partial parameter estimation of a parametric-function in (2.23): given any approximation of $F_{m-1}(X)$, the function $h(X, \mathbf{a}_m)$ can be seen as the best greedy step towards $\hat{f}(X)$. By construction from (2.25), the negative gradient based on the observations $\{x_i, y_i\}_1^n$ can be written as:

$$-g_m(x_i) = -\Big[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\Big]_{F(X)=F_{m-1}(X)}, \tag{2.31}$$

which indicates the best steepest-descent direction $-\mathbf{g}_m = \{-g_m(x_i)\}_1^n$ in the n-dimensional data space evaluated at $F_{m-1}(X)$. However, the gradient reported in (2.31) is defined only at the given $\{x_i\}_1^n$ observations. In order to generalize the solution to other datapoints, the parametric class of the weak learner $h(X, \mathbf{a}_m)$ is chosen such that $\mathbf{h}_m = \{h(x_i, \mathbf{a}_m)\}_1^n$ is parallel to $-\mathbf{g}_m \in R^n$. This

can be obtained selecting the parameters of the basis function such that the latter is most correlated to the gradient over the data distribution, such as:

$$\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{n} [-g_m(x_i) - \beta h(x_i, \mathbf{a})]^2. \tag{2.32}$$

The $h(X, \mathbf{a}_m)$ can be substituted to $-g_m(X)$ for the step size search in (2.26):

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \rho h(x_i, \mathbf{a})), \tag{2.33}$$

which can be used to compute the approximation update as:

$$F_m(X) = F_{m-1}(X) + \rho_m h(X, \mathbf{a}_m). \tag{2.34}$$

This procedure allows to estimate the parameters of (2.28) in two steps:

- solve the least squares minimization in (2.32), corresponding to fit the weak learners $h(X, \mathbf{a}_m)$ to the steepest-descent directions $\{-g_m(x_i)\}_{i=1}^n$, which are called *pseudo-responses* $\{\tilde{y}_i\}_{i=1}^n$;
- solve a single parameter optimization to find the step size in (2.33).

Any differentiable loss $L(\cdot)$ can be minimized based on forward stage-wise modeling using any weak learner, for which a feasible least squares algorithm exists. This procedure is called *Gradient Boosting Machine (GBM)*.

Considering the least squares loss function in (2.17), the GBM can be significantly simplified. This is reported in Algorithm 1. The initial guess $F_0(X)$ is typically set to the average of the response observations $\bar{y}$. The pseudo-responses from (2.31) are simply the residuals of the current approximation

Algorithm 1: Gradient boosting for least squares loss [47].

Routine
  $F_0(\mathbf{x}) = \bar{y}$;
  for $m = 1, M$ **do**
    $\tilde{y}_i = y_i - F_{m-1}(x_i), i = 1, n$;
    $(\rho_m, \mathbf{a}_m) = \arg\min_{\mathbf{a},\rho} \sum_{i=1}^{n} [\tilde{y}_i - \rho h(x_i, \mathbf{a})]^2$;
    $F_m(X) = F_{m-1}(X) + \rho_m h(X, \mathbf{a}_m)$;

with respect to the available observations: $\tilde{y}_i = y_i - F_{m-1}(x_i)$. This means that the parameters of the weak learners are chosen in order to fit the residual of the current iteration. In other words, the gradient boosting iteratively updates the function estimation by fitting the current residuals.

Regularization

Due to their structure, boosting methods may be able to fit perfectly the given observations at the costs of the generalization on new datapoints [47]. This phenomenon is called *overfitting* and *regularization methods* are applied to reduce its effects. Overfitting is described in detail in Section 2.3.2.

A natural regularization parameter for boosting is the number of iterations M: a large number of updates fitting the residuals computed on the given observations would reduce the validity of the estimate function on new data [47]. Another regularization approach is achieved through the shrinkage of the additive terms [66]. For the GBM it corresponds to the reduction of the updates impact by a *learning rate* ϵ. The approximation update in (2.34) becomes:

$$F_m(X) = F_{m-1}(X) + \epsilon \cdot \rho_m h(X, \mathbf{a}_m), \ 0 < \epsilon \leq 1, \tag{2.35}$$

such that the estimate function is generated with softened steps.

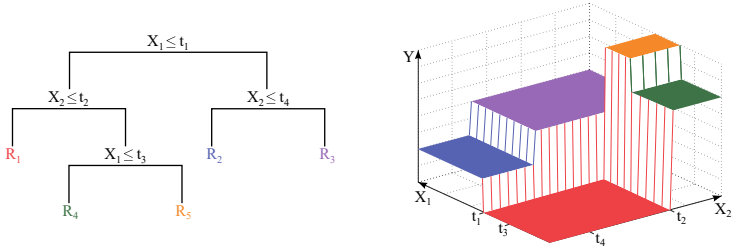
The number of iterations M and the learning rate ϵ have opposite effects [47]: for a smaller ϵ, a larger number of iterations M is required to minimize the loss. Additionally, more iterations increase the computational effort. The best regularization parameters can be identified through model selection methods (see Section 2.3.3).

Regression trees

A typical base learner family adopted for the GBM is the regression tree from CART [60], also known as decision regression tree [30, 56]. This algorithm applies recursive binary partitions on the input space defining local models in each of the resulting region. Consider a regression problem with two inputs (X_1, X_2) and a continuous response Y. Starting from the entire input space, the decision tree divides it into two regions. The split point is chosen such that the average of the responses in the regions represents them with the smallest loss. Afterwards, each region is split again into two more regions; this procedure is recursed until a stopping rule is applied, e.g. until a predefined

number of iterations is achieved. The regression model results into a piecewise constant surface, where the responses of the observations within each region are associated to the average of the responses available in that region.

Although applied for different purposes, the structure of a decision tree is similar to the one of the binary search tree reported in Figure 2.5; the main difference is that the leaves of a decision tree correspond to the regions. In Figure 2.6a an example of decision tree is represented. In this case, the input space (X_1, X_2) is divided into five regions $(R_1, \ldots, R_5)$ based on four split points $(t_1, \ldots, t_4)$. The resulting piecewise constant surface is depicted in Figure 2.6b. The major advantage of the recursive binary tree is its interpretability: the feature space can be fully described by a single tree.



(a) (b)

Figure 2.6: Simple regression tree on two inputs, from [56].

A regression tree can be written in the following additive form [47, 56]:

$$h(X, \{b_j, R_j\}_1^J) = \sum_{j=1}^{J} b_j 1(X \in R_j), \tag{2.36}$$

where $\{R_j\}_1^J$ are the disjoint regions that collectively cover the input space of the explanatory variables X, while b_j indicates the average response in the j-th region. Since the regions are disjoint, the function $1(\cdot)$ is equivalent to the prediction rule: if $X \in R_j$ then $h(X) = b_j$. The parameters of the regression tree in (2.36) are the coefficient $\{b_j\}_1^J$ and the boundaries of the regions $\{R_j\}_1^J$. Consider the parametric regression tree as base learner for the GBM with least squares loss reported in Algorithm 1. At each iteration, the regression tree is built to best predict the current residuals $\tilde{y}$, which means that the partial

estimated function $F_{m-1}(X)$ is updated by simply adding the averages of the residuals computed on the new regions. The generated piecewise-constant approximation results into a robust approach, able to attenuate the effect of wide tails distribution and outliers, both in the input and in the output variables [47].

The number of nodes of a decision tree depends on the highest order of dominant interactions within the input variables [47]. Considering p features, a single decision tree with J terminal nodes is able to represent an interaction order of at most $\min(J-1, p)$. Empirically, a low order of interactions is already able to approximate the target function [47, 67]. Therefore, small trees are preferred. The best maximum tree size for the available observations can be identified through model selection methods (see Section 2.3.3).

Bagging

The performance of the GBM can be improved injecting some randomness into the function estimation procedure [59]. Decision trees suffer from high variance [29], which means that estimating a function on different portions of data may produce different results. The *bootstrap data and aggregating models* approach [68], or shorter *bagging*, was introduced to solve this issue. It consists in bootstrapping [29], i.e. random sampling with replacement, the available observations generating B different datasets. In this way, B functions $\hat{f}(\cdot)$ are estimated separately on the bootstrapped sets. The predictions are finally averaged along the B different results:

$$\hat{f}_{bag}(X) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(X). \tag{2.37}$$

The bagging approach is included in the gradient boosting procedure with the stochastic GBM [59]. Differently from the original bagging, the functions are not averaged, but at each iteration the base learners are fitted with a bootstrapped set of the training data. This variant is called *boosting bagging hybrid* [59]. The fraction of data $s_d \in [0, 1]$ adopted for the bootstrap affects directly the amount of injected variance. For large datasets, a smaller set of observations at each iteration would reduce the computation by a factor equivalent to s_d [59].

Hyperparameters

The number of iterations M (also known as *number of estimators*), the learning rate ϵ and the fraction of subsampling s_d are the meta-parameters of the learning algorithm, also called *hyperparameters*. In addition, the hyperparameters associated to the base learner have to be set as well. In the case of regression trees, some hyperparameters are [69]:

- maximum depth of a base regression tree;
- minimum number of observations per split;
- minimum number of observations in a terminal node;
- minimum loss decrease required to split a node.

The choice of the hyperparameters may be a challenging task. More information about the hyperparameters selection is given in Section 2.3.3.

Gradient Boosting Machine Variations

In the last two decades, several variants of the original GBM have been developed. Most of them focus on improving computing and accuracy performance as well as on solving specific tasks. One of the first and most successful variant is the XGBoost [58], introducing a distributed and scalable solution for the original GBM. Afterwards, LightGBM [70] proposes a novel technique called gradient-based one-side sampling to find the split values. Finally, CatBoost [71] focuses on categorical data. In the following section, a more detailed presentation of XGBoost is given, which has been used as modeling approach in the frame of this work.

Extreme Gradient Boosting Machine Due to the increasing production of data, higher performance solutions are demanded for data modeling. The XGBoost improves the original decision tree based GBM in terms of [58]:

- a weighted procedure for efficient split proposal;
- a sparsity-aware algorithm for parallel tree learning;
- a novel highly scalable tree boosting system;
- an effective cache-aware structure for tree learning;
- additional regularization approaches.

The more extensive process in a regression tree is the search of the best split points. The GBM iterates over all the possible splits on all the features, demanding high computational resources especially in continuous domains. The XGBoost proposes an approximation of the original split search called *weighted quantile sketch*. This is based on the statistical characteristics of the data introducing a novel structure supporting *merge* and *prune* operations. The XGBoost split achieves the same accuracy of the original procedure requiring dramatically less computational resources [58]. Furthermore, the XGBoost is developed to be sparsity-aware. The term sparsity in a dataset implicates the presence of missing values or low variation. The XGBoost includes a default direction for sparse datapoints in each tree node, which is learned directly from the data. Moreover, since the GBM is an iterative procedure, the construction of the estimate function cannot be completely parallelized. Nevertheless, some procedures like the split point research are run in parallel in XGBoost and optimized through proper data structures, called blocks. Combining the block structure with management of cache and memory, high computational performance are achieved.

XGBoost introduces additional regularization forms to the already implemented in GBM. One of them is the *regularized learning objective* that penalizes the loss function $L(\cdot)$ according to the complexity of the trees. Consider M boosting iterations and that each m-th regression tree has size J_m and weights γ_m associated to its leaves. The regularized loss function for XGBoost becomes:

$$\mathcal{L}(Y, f(X)) = L(Y, f(X)) + \sum_{m=1}^{M} \Omega(J_m, \gamma_m), \tag{2.38}$$

where:

$$\Omega(J, \gamma) = \alpha J + \frac{1}{2}\lambda ||\gamma||^2. \tag{2.39}$$

The two terms α and λ are known as L_1 and L_2 regularization respectively [30]. The parameter α penalizes complex and large trees in order to reduce overfitting [58]. Differently, λ favors models able to achieve accurate predictions by penalizing trees producing large residuals. The regularized objective tends to select a model employing simpler and more predictive functions.

Typically, if a dominant predictor is present in the dataset, this would be chosen as split variable for all the decision trees, producing highly correlated trees and losing precious information regarding the other features. For this reason, XGBoost selects a random subset of predictors at each iteration, which

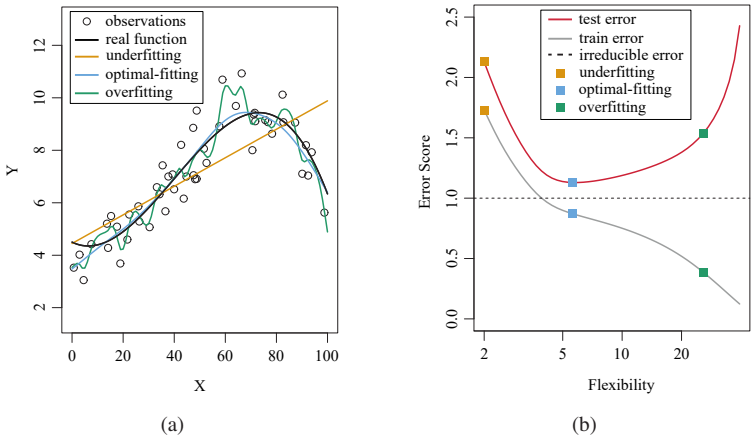
are then further sub-sampled at each level and at each split [58]. In this way, additional variance is brought at each update in order to deeper explore the feature space.

2.3.2 Modeling Performance

An essential step of the data modeling is the assessment of the accuracy [29, 30, 56]. Several evaluation metrics, also called *accuracy* or *error metrics*, are available to quantify the prediction capability of a learning method. These metrics measure the deviation between the predicted responses $\{\hat{f}(x_i)\}_1^n$ and the ground truth values $\{y_i\}_1^n$, where $\{x_i, y_i\}_1^n$ are in this case a general set of observations, not necessarily those used to fit the learning algorithm. Indeed, in supervised learning two types of errors can be computed: the *training error* and the *test error*. The former is computed on the data used to train the model, also called *training data*, while the latter is computed on new data, which are also known as *test* or *validation data*. The prediction capability of a learning method on the test data is referred as *generalization* [30].

The division between test and training error is required since the learning procedure may identify a relation in the training data caused by random chance rather than by the real proprieties of the investigated phenomenon. This problem is called *overfitting* and it occurs when the training error is much lower than the test error. Nevertheless, the training error is expected to be generally lower [29]. The opposite effect of overfitting is called *underfitting*. This case cannot be identified directly from the errors, but it indicates the inadequacy of the chosen learning method to capture the real relationship in the data. An example of overfitting and underfitting is reported in Figure 2.7 from [29]. In Figure 2.7a, general observations are depicted together with the real function and different estimates. In particular, the latter are defined adjusting the modeling hyperparameters in order to influence the flexibility of the learning method. The optimal-fitting is the one close to the real function. The underfitting situation is represented by a linear estimation, which cannot model the non-linearity of the underlying phenomena. Finally, in the overfitting case the model learns the noise in the data as well, missing the real relationship of the observations. In Figure 2.7b, the test and training errors are depicted in function of the learning algorithm flexibility. In the underfitting case, both test and training errors are large due to the inability of the chosen model to estimate the real behavior. In contrast, a very flexible method may learn perfectly the training data, while missing the generalization on new observations.

The horizontal dashed line indicates the irreducible error induced by external factors, as reported in (2.6). The optimal-fitting corresponds to the model able to produce the minimum reducible error achievable by the chosen applied learning technique. The *U-shape* assumed by the test error over the model flexibility in Figure 2.7b is independent of the dataset or the applied learning method [29,56]. A proper trade-off between modeling flexibility and error has to be always found. In particular, the "*no-free lunch theorem*" [72] indicates that there is no universal method able to outperform other approaches for all possible datasets. Therefore, the proper learning algorithm has to be selected and calibrated in order to produce the best results on a given dataset.



(a) (b)

Figure 2.7: Examples of overfitting and underfitting, from [29].

Another factor influencing the fitting performance is the correct choice of the training and test sets, especially when a proper test set is not available due to the high costs of the data generation. Some *training-test splitting* approaches are introduced later in this section.

Evaluation Metrics

In this section three evaluation metrics adopted to compute the training and test scores in regression tasks are introduced: the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE) and the coefficient of determination R^2.

The MAE [55], also called averaged L_1-norm, indicates the averaged absolute deviation between the ground truth and the prediction, such as:

$$MAE(Y, \hat{f}(X)) = \frac{1}{n} \sum_{i=1}^{n} \left| y_i - \hat{f}(x_i) \right|. \tag{2.40}$$

The RMSE [55] indicates the performance of the model as the root square of the averaged squared distance between the ground truth Y and the prediction $\hat{f}(X)$, such as:

$$RMSE(Y, \hat{f}(X)) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}(x_i) \right)^2}. \tag{2.41}$$

This metric corresponds to the L_2-norm between the real y_i and the predicted $\hat{f}(x_i)$, averaged by $\sqrt{n}$, where n is the number of datapoints. Due to the squared difference, the RMSE is very sensitive to large outliers: in case of a wrong prediction, the whole score is largely affected.

Differently from the previous two, the coefficient of determination R^2 [48] is a unit-independent metric and it is generally adopted to quantify the linear relationship between two variables. The linearity of the relation $Y = \hat{f}(X)$ is used to evaluate the goodness-of-fit of a model. The R^2 is defined as:

$$R^2(Y, \hat{f}(X)) = 1 - \frac{MSE(Y, \hat{f}(X))}{Var(Y)} = 1 - \frac{\sum_i^n (y_i - \hat{f}(x_i))^2}{\sum_i^n (y_i - \bar{y})^2}, \tag{2.42}$$

where the Mean Square Error (MSE) corresponds to the square of the RMSE in (2.41) normalized by the variance of the ground truth Y. If the MSE is zero, then the prediction matches the ground truth, resulting in $R^2 = 1$. In contrast, if the predictive capability of the model is equal to a baseline able to only predict the average of the observed responses $\bar{y}$, i.e. $\hat{f}(X) = \bar{y}$, then $R^2 = 0$. In case that the model is worse then the baseline, R^2 assumes negative values.

It is important to distinguish the introduced evaluation metrics from the loss function considered as objective during the function estimation in (2.16). Considering the training set, both evaluation metric and loss function indicate the reducible error in (2.6); however, they are applied for different scopes. While the loss function is part of the learning algorithm, the error evaluates the general model performance a posteriori.

Training-Test Split

Different approaches are available to obtain a proper test dataset from the available observations [29, 56]. Generally, two essential requirements have to be fulfilled. First, the datapoints in the test set do not have to be included in the training set, otherwise the generalization capabilities of the model cannot be properly evaluated [73]. This may be the case of duplicated observation in the dataset. Second, both sets should represent the whole input space.

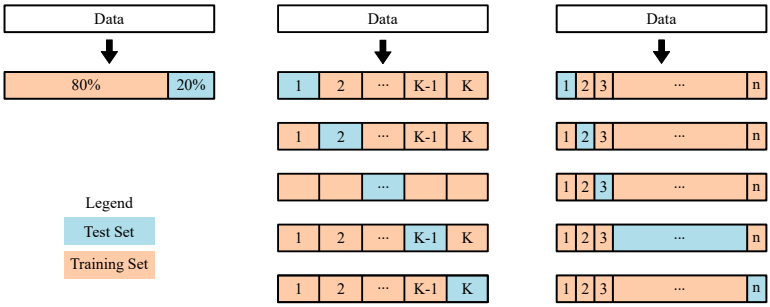
In Figure 2.8, the most common training-test splitting approaches are depicted. The blue boxes indicate the test data and the orange ones the training data. The simplest split is represented in Figure 2.8a. It consists in randomly splitting the dataset into two portions: typically, 80% of the data are used for the training and the rest 20% for the test. However, reducing small datasets further may affect the representation of the input space. To deeper evaluate the modeling performance based on the whole available observations, the *cross validation (CV)* is applied. The K-fold CV procedure is summarized in Figure 2.8b. This method consists in randomly dividing the observations into K different folds of approximately equal sizes. Afterwards, $K-1$ folds are used as training set and the remaining fold as test set. This operation is repeated K times, using each time a different fold as test set and consequently different $K-1$ folds for the training set. In this way, K validation errors are generated $(Err_1, \ldots, Err_k)$. Finally, the K validation errors can be evaluated through their average μ_{Err} and standard deviation σ_{Err}, such as:

$$\mu_{Err} = \frac{1}{K}\sum_{i=1}^{K} Err_i, \tag{2.43}$$

$$\sigma_{Err} = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(Err_i - \mu_{Err})^2}. \tag{2.44}$$

As instance, in case the selected evaluation metric is the MAE, those are referred as μ_{MAE} and σ_{MAE}. The value of K is chosen according to the quantity of data available in order to maintain a certain input space representation. Commonly, K is set equal to 5 or 10 to ensure 20% or 10% of the data as test set [29]. A variant of the K-fold CV is the Leave-One-Out (LOO) CV reported in Figure 2.8c, where K is set to be equal to the number of available observations n. In this case, the error is evaluated on every single observation

with a model trained on the remaining $n - 1$ points. The LOO CV estimates more precisely the generalization error for future predictions; however, it is not ideal for computational expensive function estimations or in case of large datasets [29]. After the assertion of the model quality, all the available data are typically used to train the final model.



(a) (80-20)% Split (b) K-fold CV (c) Leave-One-Out CV

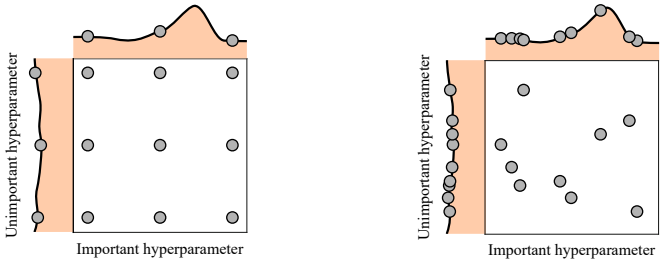
Figure 2.8: Modeling validation types.

2.3.3 Hyperparameter Optimization

As already introduced for the GBM in Section 2.3.1, the hyperparameters are the meta-parameters of the learning algorithm. The selection of the hyperparameters is necessary in order to adapt the learning method to the given data [74]. In particular, the right choice of hyperparameters can avoid overfitting and ensure generalization on new data, as mentioned in Section 2.3.2. The selection process of the optimal hyperparameters of a learning method for a given set of observations is referred as *Hyperparameter Optimization (HPO)*.

Beside manual search, optimization methods can be applied to solve an HPO problem [74, 75]. In this case, the learning algorithm is treated as a black-box and different combination of hyperparameters are tested in order to find the optimal model on the given data. Typically, potential hyperparameters are determined through sampling techniques on the whole hyperparameter space. One of the simplest black-box HPO is the *Grid Search*, also known as *full factorial design* or *Grid Sampling* [76]: for each hyperparameter a finite set of values is manually chosen and their Cartesian product is evaluated, as shown

in Figure 2.9a. Finally, the best hyperparameter set is selected. However, the number of function estimations grows exponentially with the considered hyperparameter space size and the chosen discretization [74]. An alternative to the Grid Search is the *Random Search* or *Random Sampling* [75]. Here, the modeling configurations are sampled at random on the hyperparameter space, as represented in Figure 2.9b. Random Search is considered to outperform Grid Search, in particular when the hyperparameters do not have the same effectiveness on the model [75]. In Figure 2.9, Grid Search is able to achieve a higher coverage of the two dimensional hyperparameter space; however, the projections onto the hyperparameters axis result much more efficient for the Random Search considering the importance of the hyperparameters. To be specific, considering a budget of B possible function evaluations and N hyperparameters, Random Search is able to explore B different solutions of each hyperparameter, while Grid Search only $B^{1/N}$ [74]. Nevertheless, both approaches can be well parallelized.



(a) Grid Search (b) Random Search

Figure 2.9: Comparison between Grid Search and Random Search on a two dimension hyperparameter space, considering their effect on the learning algorithm, based on [75].

A more complex black-box HPO is the *Bayesian Optimization* [74], which is typically adopted for computational expensive learning algorithms. This is an iterative approach composed of a probabilistic surrogate model and an acquisition function, which suggest the points to evaluate next. However, the Bayesian Optimization is still limited to some specific applications, especially for low hyperparameter space, e.g. for the tuning of deep neural networks [77]. Another family of black-box optimizations applied for HPO are the *evolution-*

ary algorithms [74,78], which are deeper described later in this section due to their relevance to this work.

In contrast to black-box optimizations, *multi-fidelity* HPOs include some knowledge of the learning algorithm in the optimization process. One of these is the *early stopping* [79], which avoids overfitting for iterative learning algorithms. The difference between the training and the test errors is monitored along the iterations and the learning algorithm is stopped as soon as no further improvements are achieved. A more recently proposed multi-fidelity HPO is the *Hyperband* [80]. In this case, a predefined type of budget is allocated to randomly sampled hyperparameters. The Hyperband searches the best hyperparameters able to consume less possible budget, while achieving the best accuracy. The budget may be, for instance, the number of iterations of the learning process.

Intensive HPOs may provide biased hyperparameters valid only for the considered observations. Therefore, an additional evaluation with data not included during the HPO is required. In particular, the test and validation sets introduced in Section 2.3.2 assume here different meanings. The validation set indicates the datapoints adopted to evaluate the goodness of the model during the HPO, while the test dataset is used to evaluate the final hyperparameter set. The typical procedure is described as follows. First, the whole data is split into training and test data. The training data are then further divided by a K-fold CV to choose the best hyperparameters during the HPO. Finally, the chosen hyperparameters are evaluated on the test data to ensure their generalization. This procedure is shown in Figure 2.10 with a $(80-20)\%$ training-test split. The score computed on the final test set is typically indicated as e_{Err}.

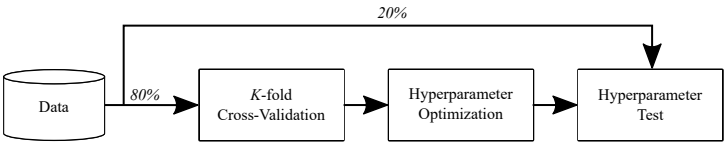


Figure 2.10: Generic HPO validation with a $(80-20)\%$ training-test split and a K-fold CV.

Evolutionary Algorithms

Evolutionary algorithms are optimization techniques based on biological evolution and natural selection theory [78], whose main mechanism is reported in Figure 2.11. These are iterative procedures, where each iteration is called *generation*. During each generation, the search space is explored and new solutions are evaluated through a *fitness function*. In case of HPO, the search space corresponds to the hyperparameter space and the fitness function to the evaluation of the hyperparameters on the training data. Every proposed solution is called *individual* and a set of individuals composes a *population*. Starting with an initial, generally random, population of size μ, the first step in each generation corresponds to select individuals in order to create new solutions in the search space. The selected individuals are called *parents* and the selection procedure is referred as *selection for reproduction*. Typically, potentially good solutions have higher chance to be selected as part of the parents set. Afterwards, specific stochastic operators are applied in order to introduce some variations into the parents set. These operators can be divided into *mutation* and *crossover*. Mutation refers to the variations of a single parent in order to create a new individual, while crossover, also known as combination, involves two parents for the creation of at least a new solution. The just generated

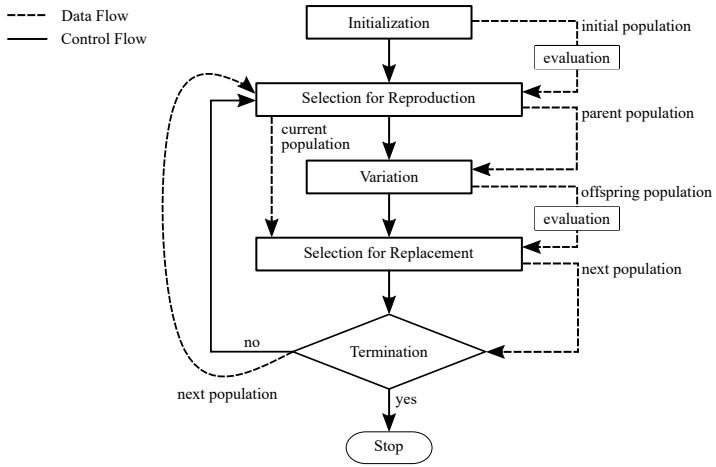
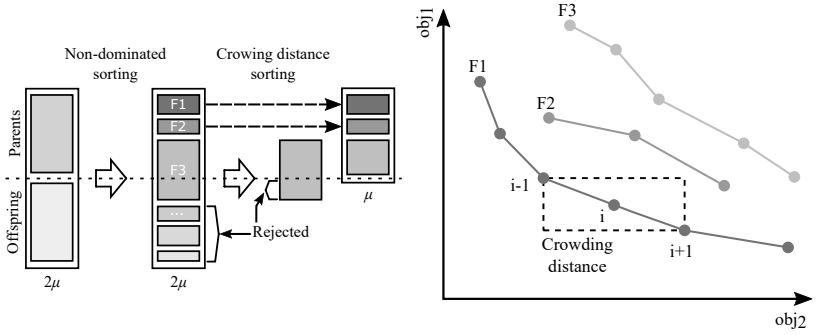


Figure 2.11: Generic evolutionary algorithm flowchart, based on [78].

individuals are called *offspring*. The final step corresponds to the *selection for replacement*, where individuals from parents and offspring are chosen to generate a new population of the same size μ. This selection can be interpreted as selection for survival: only the best individuals are allowed to be member of the next population. The evolutionary cycle is iterated until a *termination criterion* is meet, e.g. a maximum number of iterations.

In the last decades many variations of the general evolutionary algorithm have been developed. In particular, the research focused on different strategies for selections and stochastic operators as well as on different individual representations. One of these variations is the Genetic Algorithm (GA) [81, 82], where the individuals assume a genetic representation composed of chromosomes and genotypes, which can be mutated and altered. A particular GA is the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [83], which is an improvement of the NSGA [84]. The NSGA-II is one of the most adopted multi-objective optimization able to achieve fast and robust solutions. In this case, the selection for replacement is based on the *non-dominated sorting* and *crowding distance* estimation, which are able to generate better spread solutions on multi-objective domain spaces. In a multi-objective optimization, a solution x_1 is said to dominate another solution x_2, if x_1 is no worse than x_2 in all objectives and if x_1 is strictly better than x_2 in at least one objective [85]. The non-dominated solutions represent a non-domination front, also called *Pareto front* [85]. In Figure 2.12 the selection for replacement of the NSGA-II is represented. The non-dominated sorting classifies the population composed from parents and offspring into different non-domination classes. First, considering the whole population, the non-dominated solutions are associated to the class $F1$. Afterwards, the non-dominated solutions of the remaining individuals are clustered into the class $F2$, and so on until all the individuals are associated to a class. The different F classes correspond to the Pareto fronts, as represented in Figure 2.12b. The first μ individuals in the sorted F classes are selected to represent the population of the next generation, while the rest are discarded. In this way, the dominating solutions are kept along the generations, such that the NSGA-II is considered an elitist algorithm. In case that a front has to be split in order to fit the new population size, its individuals are sorted according to the so-called crowding distance. The crowding distance of an individual i indicates the space around it not occupied by any other solution, as represented in Figure 2.12b. The crowded-sorting ensure the persistence of diversity in the population. Typically, the first front $F1$ is referred as the Pareto front of the

considered generation. The final solution is then chosen from the final Pareto front.



(a) NSGA-II selection for replacement.

(b) Pareto fronts for objectives minimization.

Figure 2.12: NSGA-II characteristics, based on [83, 86].

2.4 Knowledge Interpretation

The main drawback of complex function estimations is their black-box characteristic. Robust and accurate models can be built, but no explanations regarding the reasons of specific decisions are provided [87]. The necessity of explainable AI is growing in many fields, principally where legal and ethical norms are essential [88]. Moreover, the interpretability of the models may be used for acquisition of new knowledge as well [87]. In particular, a large interest concerns how variations in the input space influence the investigated phenomena.

2.4.1 Feature Importance

Based on their structure, decision trees (see Regression trees in Section 2.3.1) can be easily interpreted: the influence of every single feature on the considered response can be extracted from the data. Consider i_t as the improvement of the estimation achieved with the split at the node t. For a given decision tree

T, the relative influence I_j of the variable j on the estimate function can be defined as [60]:

$$I_j^2(T) = \sum_{t=1}^{J-1} i_t^2 1(v_t = j), \tag{2.45}$$

which indicates the summation over the squared improvements i_t^2 of the internal $(J-1)$ nodes t, when the splitting variable v_t coincides with the variable j. The relative influence of a variable corresponds to the number of times it was selected for a split during the construction of the tree, which is then weighted with the improvements i_t^2 attributed to those splits. For boosting trees, the influence of a variable j on the single trees is averaged over all the M basis learners [47], such as:

$$I_j^2(T) = \frac{1}{M} \sum_{m=1}^{M} I_j^2(T_m). \tag{2.46}$$

Generally, the influences of all the input variables on the output are normalized such that their sum is one. The feature importance can be considered as knowledge extracted from the data in the KD process: in this way, a data-driven development can focus only on the most important features. Additionally, the importance can be used as feature reduction in the data preprocessing stage, as introduced in Feature Reduction in Section 2.2.4.

2.4.2 Partial Dependencies

The feature importance analysis does not provide any information about the nature of the relationship between the explanatory and the response variables. This kind of information can be depicted as partial dependence of the output on few input variables [30,47]. Consider the input features X of size p and X_S a subset of it. Let X_C be the complement set of X_S, such that $X_S \cup X_C = X$. In particular, the estimate function $\hat{f}(X)$ depends on all the input variables: $\hat{f}(X) = \hat{f}(X_S, X_C)$. Given n observations, the partial dependence of $\hat{f}(X)$ on X_S can be estimated as [30]:

$$\hat{f}_S(X_S) = \frac{1}{n} \sum_{i=1}^{n} \hat{f}(X_S, x_{iC}), \tag{2.47}$$

where $\{x_{iC}\}_1^n$ are the observations of X_C. The partial dependency corresponds to the effect of X_S on $\hat{f}(X)$ taking into account the average effects of X_C. If X_S and X_C are independent, i.e. they do not have strong interactions, the partial dependence can be interpreted as the effect of X_S on $\hat{f}(X)$, ignoring X_C [30]. In case the two subsets are dependent, possible effects of X_C on $\hat{f}_S(X_S)$ have to be taken into account. The number of dependent variables can be reduced with the multicollinearity analysis (see Section 2.2.4). Partial dependence functions can be applied to interpret the results of any black box learning method [30, 56].

2.5 Summary

In this chapter the fundamental notions of the KD process are introduced: starting from its definition and the nomenclature adopted in this work, the main stages to extract the knowledge from raw data are described. A particular attention is addressed to the data preprocessing and data modeling, which are the two core steps in order to ensure reliability and quality of the extracted information. Even though most of the preprocessing procedures are based on domain expertise, several tools and operators are available to support this step, e.g. data inconsistency and redundant information handling. In the data modeling section, beside the basic concepts of learning algorithms, the boosting approach is introduced through GBM and XGBoost as a robust and efficient function estimation. Furthermore, the essential techniques employed to select a model and evaluate its performance on the available data are reported as well, e.g. validation procedures and proper choice of data and modeling tuning. Finally, the extraction of the knowledge from the models is reported in terms of feature importance and partial dependencies.

3 Knowledge Discovery Framework

In this chapter, the KD framework specifically designed and developed for the multidisciplinary analysis of complex and multidimensional phenomena in the product development is presented. The overview of the KD framework is depicted in Figure 3.1. This can be considered as a bridge between the product development and the AI. As shown in the left side, the product development includes measurements and simulations of components, sub-systems and complete systems. On the other side, the AI incorporates data preprocessing, computer science, statistics and machine learning, as introduced in Section 2.1. The framework is responsible to collect the data from the product development and provide them to the AI, where advanced data analysis tools are adopted in order to derive complex information from the data. Afterwards, the extracted knowledge is transferred back to the product development and it is integrated with the already available know-how. In this way, it is possible to focus future investigations on yet to be explored research areas and design spaces. Such upcoming studies will generate new data leading to an iterative and continuous product improvement.

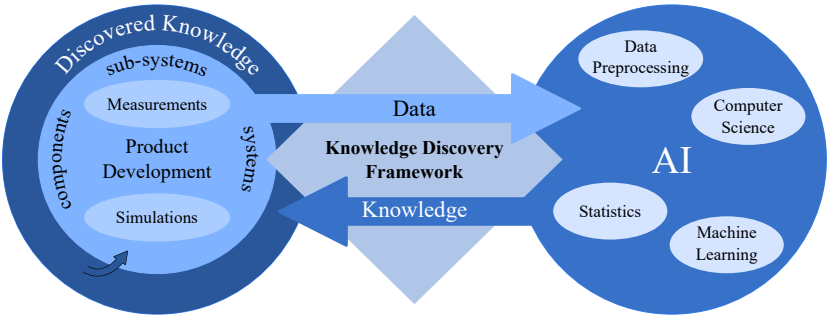


Figure 3.1: Overview of the KD framework, showing the interaction between product development and AI.

For the scope of the current work, the general structure of the KD process depicted in Figure 2.3 is adapted to and encapsulated in a specific workflow, which is reported in Figure 3.2. The latter includes algorithms and methodologies able to deal with the main demands for an interdisciplinary analysis of complex phenomena. Starting from the raw data, three main steps are required to access the knowledge:

1. **Data Preparation**: includes data preprocessing and data exploration in order to get familiar and prepare the data for the analysis (see Section 2.2);
2. **Data Modeling**: selects and validates the best modeling configuration in terms of HPO and train-test-validation datasets (see Section 2.3);
3. **Knowledge Extraction**: extracts the knowledge in terms of *Model Exploration* and *Model Exploitation*. The term exploration indicates the extraction of feature importance and partial dependencies from the models (see Section 2.4). Differently, exploitation consists of exploiting the trained models in order to substitute costly and time consuming evaluations. These two parts represent respectively the descriptive and the predictive tasks, introduced in modeling taxonomy in Section 2.1.2.

The direction of the arrows in Figure 3.2 indicates the main flow of the process; the single steps can still be reviewed based on intermediate results. The KD framework is named pyMICE, which stands for ***py**thon **M**ining **I**nternal **C**ombustion **E**ngines*. As the name suggests, the framework is written in the cross-platform programming language Python [89], which includes libraries specialized for machine learning and advanced data analysis.

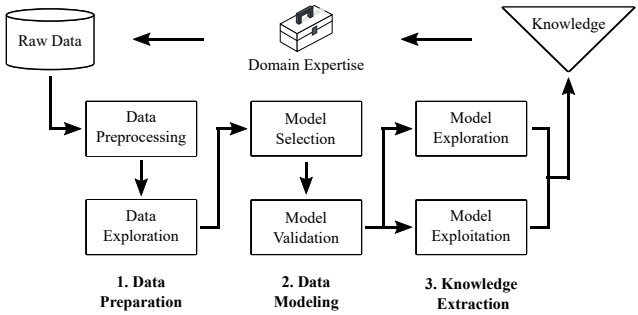


Figure 3.2: Workflow of the KD framework, based on Figure 2.3.

In the next sections, the main characteristics of the KD framework are described.

3.1 Modularity

The modularity is an essential feature of the framework: a modular and multidisciplinary framework is able to collect and analyze data independently from their source and from the investigated component or system. The modularity is generally defined with the terms abstraction, information hiding and interfaces [90]. A complex problem can be divided into smaller steps represented by modules, which are indeed abstraction levels. The modules interact with each others by means of well defined interfaces in order to complete a specific task. According to the complexity of the single operations, a module can be divided into several sub-modules, i.e. sub-steps.

The KD framework is developed such that every stage of the workflow presented in Figure 3.2 is a module. Each of them requires only a dataset and specific settings regarding the operations to be performed, independently from the content and the source of the data. Typically, default settings are available such that only the data to analyze are necessary. The modularity enables four fundamental characteristics of the KD:

1. the analysis is based on the dataflow and decoupled by the workflow direction. Every single procedure offered by the framework can be applied at any time of the analysis by just providing the data. For instance, it is possible to re-run any specific preprocessing operators based on the final extracted knowledge;
2. the generalization of the analysis is enhanced since the modules are independent from the physical meaning of the data. The interpretation of the results and the direction of the analysis are left to the domain expert;
3. the application of complex and sophisticated algorithms can be abstracted to the user. The latter is required to provide few intuitive settings to the framework, which then applies the proper algorithms according to the desired task. An example may be the data modeling: the framework runs the proper modules in order to process the provided data, let them compatible with the modeling algorithm and select the best hyperparameters in order to ensure maximum accuracy and generalization;

4. as long as the interfaces are satisfied, every module can be singularly substituted without affecting the whole framework; for example, an improved learning algorithm or an additional preprocessing procedure can be easily implemented in this way.

3.2 Data Management

In this section, the main aspects of the data in the product development are exposed, focusing on the GDI context. First, the main characteristics and the requirements to ensure high modeling quality are presented. Afterwards, the storage structure developed within the KD framework in order to manage those data is introduced.

3.2.1 Data Characteristics and Requirements

Due to the nature of the investigations in the GDI context, the produced data are highly heterogeneous. The main contribution to this characteristic is given by the several components, sub-systems and systems analyzed with different measurement and simulation procedures. Each of them is able to answer specific research questions with different abstraction levels and precision. Therefore, most of the data are diverse and not directly compatible with each other. For instance, the design space of the injector valve seat is continuous from a simulation point-of-view, while it is limited by manufacturing constraints for experimental investigations. Furthermore, the several simulation and measurement procedures provide different data representations, contrasting naming and measurement units as well as diffferent data structures and file formats. This produces inconsistencies in the data.

Another main challenge is the complexity and the high dimensionality of the problems. In order to limit the large design spaces in this context, constraints and specifications based on domain expertise or on previously extracted knowledge have to be included in the analysis. For example, the influence of the injector geometries on spray characteristics is typically analyzed within fixed engine operating points or for a single engine, which are based on customer requirements or research indications. However, separated investigations based on different constraints would often produce datasets incompatible with each others. Here, the domain expertise is essential: the available data and the

extracted knowledge have to be contextualized within the considered investigation.

Preprocessing Demand

The quality of simulations or measurements data in the GDI context is considered high. Robust and automated simulation workflows include procedures able to detect numerical anomalies during the simulations. Measurements are partially manually collected, therefore, missing values and outliers are rare considering the human supervision. Missing values are generally atypical for simulations as well: these are either completely solvable or numerically impossible.

Independently from the source of the data, the common preprocessing tasks for GDI data are reduction and aggregation. The first includes removing redundant or useless information to avoid multicollinearity and the second aims to reinforce the collected information. During the data reduction, the domain knowledge is fundamental in order to validate the plausibility and the causality of the correlations. For example, to recognize spurious or nonsense correlations (see Section 2.2.4). The latter may be induced by specific sampling constraints, by manufacturing limitations or by given design specifications. Different are the correlations due to variables totally or partially representing the same quantity, which can be neglected without consequences. Furthermore, time and space observations are often averaged during the collection of the results in order to focus the investigation on the most attractive surfaces and instants.

Another critical circumstance is the presence of duplicated or irrelevant observations. Typically, every single experiment is repeated a certain number of times in order to mitigate the measurement errors. Moreover, calibration or reference points are often included in the datasets, which do not include any additional information for the analysis. This issue is not common for simulation data, since these are more deterministic then experiments.

Data Sampling Approaches

The design space represented by the available observations has a strong influence on the quality and the generalization of the models. In particular, it is not possible to guarantee the correctness of the models outside the training boundaries. In contrast, within the training boundaries, the quality of the

models can be ensured through proper sampling approaches. So-called *Design Of Experiments (DOE)* procedures are applied to cover the high-dimensional domain space of the input variables as best [91, 92].

A common DOE approach corresponds to the Grid Sampling, already introduced in Section 2.3.3 for the HPO. In this case, potentially good design values are combined together in the Cartesian product. More complex methods belong to the *pseudo-random* and the *quasi-random sequences* [93]. Pseudo-random samplings aim to mimic random natural processes to generate a sequence of designs [93]; an example is the Random Sampling previously introduced in Section 2.3.3. Quasi-random samplings generate new designs taking into account the already sampled ones, so that a low-discrepancy sequence is generated [93]. A widely used quasi-random sampling method for high-dimensional design spaces is the *Sobol-sequence* [94, 95]. Quasi-random sampling approaches require a well defined sampling plan, including the number of dimensions and the sampling boundaries, limiting the possibility to expand it at a later time. Sobol has been applied to analyze the effect of injector valve seat geometries on spray characteristics based on 3D-CFD simulations in [14, 16]. In contrast, the coupling of two Grid Samplings has been applied in [96] to investigate emissions and spray characteristics through experimental measurements.

3.2.2 Storage and Accessibility

Prerequisites for a modular and dataflow-oriented framework are the standardization of the data formats and a proper data storage system. Since the modular framework is supposed to handle a large amount of different datasets, it should be aware of their formats. Due to the several measurement procedures and simulation tools, the raw data are available in different formats, e.g. as Comma-Separated Values (CSV), as simple text files or in some cases even as proprietary formats. Another main complication is the location of the raw data. The accessibility to the data may be limited by the lacking of a central data storage. For these reasons, a system able to store the data in a single format and to interface with the framework is essential. Furthermore, intermediate results have to be stored as well, enabling the possibility of moving backwards to any previous version of the data in order to skip or re-run preprocessing steps. This system is generally referred as data warehouse [97]. In particular, the interface regarding the extraction, the transformation and finally the load of raw data into a data warehouse is typically referred as Extraction, Transformation, Load (ETL). The data storage system adopted in this work is the

Hierarchical Data Format 5 (HDF5) [98], which is a portable scientific format allowing the storage of data in a hierarchical structure, including interfaces for writing, reading and organizing data and metadata.

The data management structure of the KD framework is reported in Figure 3.3. In this case, the ETL is characterized by an Automatic Data Extraction (ADE), an Automatic Data Transformation (ADT) and specific data format configurations. The latter include information about data location, formats, input/output variables, units and naming conventions. Based on the data format configurations, the ADE and ADT extract the raw data, convert names and measurement units and then load them into the data warehouse. The automatic procedures allow to speed up the raw data processing, hiding the main ETL mechanisms and requiring just few information about the format. In this way, new data, whose format configuration has already been defined, can be automatically loaded without any additional effort. The standardization of names and measurement units enables the integration of multiple datasets. In case of highly heterogeneous and incompatible datasets, their results can still be compared if standardized.

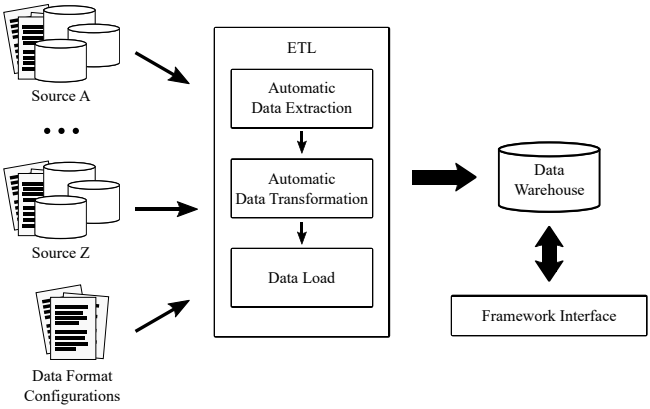


Figure 3.3: Data management structure of the KD framework.

The data warehouse ensures that all the required data are stored in a central place, including meta-information and intermediate result. A detailed view of the data warehouse of the KD framework is represented in Figure 3.4.

Considering the hierarchical structure of the HDF5, it is possible to organize the datasets into five different categories:

- *raw data* → original data extracted from the ADE process;
- *transformed data* → results of the ADT process;
- *selected data* → data selected for the analysis;
- *cleaned data* → results of the data preprocessing;
- *metadata* → measurement units and original names.

Each dataset is organized further into a dataset for the input variables and one for the output variables. With this structure, any intermediate data can be recalled at any point of the analysis.

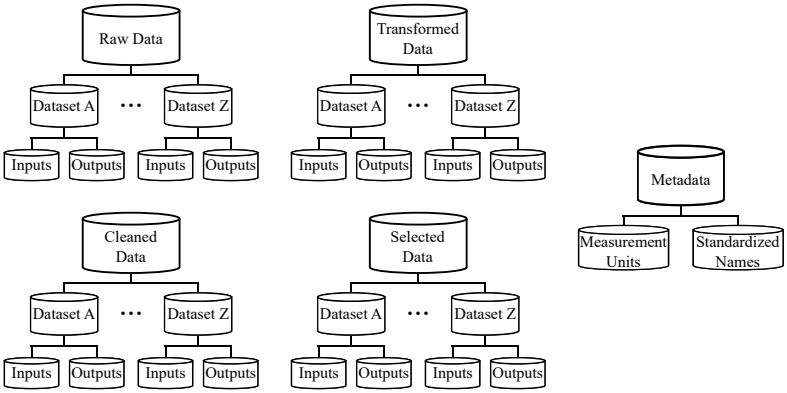


Figure 3.4: Data warehouse structure of the KD framework.

3.3 Framework Structure and Usage

The modular structure of the KD framework, i.e. of pyMICE, is reported in Figure 3.5 as Unified Modeling Language (UML). The framework can be considered as a single module containing in turn several sub-modules. The main modules of the framework are *ETL*, *Tools*, *Analysis* and *Optimization*, while the external interfaces are *Storage*, *HPC Manager* and *User Interface*. These are introduced as follows together with their sub-modules.

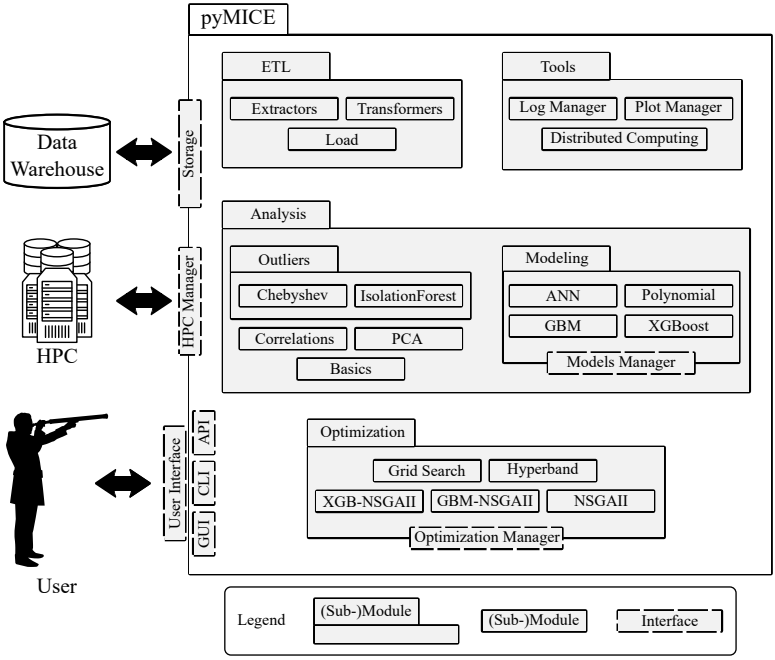


Figure 3.5: UML showing the modules and the interfaces included in the KD framework.

Resources Interfaces. The *Storage* interface allows the connection between the framework and the data warehouse, including all the procedures in order to store, to read and to manage the data. The *HPC Manager* interacts with the nodes of the HPC in order to correctly submit and manage remote jobs.

User Interfaces. The high level of abstraction due to the modularity supports the users to perform the data-driven analysis. The KD framework provides different interfaces according to the scope and the expertise of the user. Every single module of the framework can be implemented through Application Programming Interfaces (APIs) in other source codes. Automatized procedures like HPO, training and validation of models as well as predictions can be performed through the Command Line Interface (CLI). In this case, the main settings to complete the task are provided in form of configuration files, e.g. the amount of available resources in case of a distributed computation. Finally,

a Graphical User Interface (GUI) exposes all the information extracted from the data in form of an interactive web application. An example of the latter is reported in Appendix B.

ETL. The *ETL* module contains custom extractors and transformers based on the provided data format configurations. Through the *Storage* interface, extracted and transformed data as well as metadata are properly loaded into the data warehouse.

Tools. The *Tools* module contains general support functionalities. The *Distributed Computing* allows to set up a scheduler and different workers enabling the computation and the management of parallel processes. The *Distributed Computing* module is separated from the *HPC Manager* in order to increase the flexibility of the framework: this allows to choose whether to run the computations on the HPC or on a local machine. The *Log Manager* allows to keep track of the automatized procedures of the framework. This is useful for time-consuming operations or for debugging. Finally, the *Plot Manager* is another sub-module contributing to the overall modularity of the framework. Data visualization is an essential part of the KD process: many complex representations of the data are demanded, especially during the data exploration and preprocessing. The construction of such plots requires often data preparation and the configuration of specific plotting libraries. Furthermore, it is necessary to maintain certain standards between the analyses, such as plots structures, fonts, sizes and so on. In order to fulfill these requirements, the *Plot Manager* includes interfaces for several figures typologies, requiring as input only the data to visualize and few settings to plot them in a standardized way.

Analysis. The *Analysis* module contains all the required algorithms and procedures in order to analyze the data. Through the *Outliers* module it is possible to apply the Chebyshev outlier detection or the Isolation Forest, introduced in Section 2.2.2. Other modules allow to perform the correlation analysis or reduce the dimensionality of a dataset applying the PCA, as introduced in Section 2.2.4. Some simple operations such as listing the number of entries with missing values or querying the dataset are included in the *Basic* module. The *Modeling* module implements the learning algorithms together with the procedures to evaluate and validate the models. The available algorithms are ANN, polynomial regression, GBM and XGBoost. Moreover, the procedures to compute and to visualize feature importance and partial dependencies are available for the boosting-based algorithms. The *Models Manager* interface

connects the *Distributed Computing* to the *Modeling* in order to train the models in a distributed environment. In this way, the potential of the HPC can be fully exploited to test different hyperparameters, to run CVs or to compute partial dependencies in parallel. The *Models Manager* includes also operations to create documents reporting the modeling performance and the extracted knowledge.

Optimization. The *Optimization* module includes optimization algorithms. One of the algorithms is the NSGA-II, which is then implemented as XGB-NSGAII and as GBM-NSGAII in order to optimize the hyperparameters of the learning algorithms XGBoost and GBM respectively. These procedures are accessible through an *Optimization Manager*, which abstracts the complexity of the algorithms automatizing the whole HPO procedures. In this way, only the dataset to model is required to start the optimization. The progression and the results of the optimization are collected and summarized in reports and diagrams allowing a fast and easy assessment of the outcome. Hyperband and Grid Search are implemented as well. The novel XGB-NSGAII developed in this work is described in Chapter 4.

3.4 Summary

In this chapter, the KD framework is presented. After a general introduction about the main structure and scope of the framework, the necessity of modularity is highlighted. Afterwards, the management of the data within the framework is described. First, the main characteristics of the data in the GDI context are reported together with the main preprocessing operations and the required data characteristics in order to ensure a robust and reliable knowledge extraction. In particular, sampling methods to cover properly the investigated domain space are reported. Then, the storage and accessibility of the data in the framework are explained. Finally, the structure and the usability of the framework are reported, presenting the main modules required to enable the KD. The application of the KD framework in the GDI context is presented in Chapter 5 and Chapter 6.

4 Model Selection for Heterogeneous Datasets

As already introduced in Section 2.3.2, specific hyperparameters are required for any learning algorithm trained on a given dataset. Due to the several heterogeneous and multidisciplinary datasets in the GDI context, an automated, parameter-free, dynamic and data-driven model selection has been developed within this work. This algorithm is referred as XGB-NSGAII: it optimizes the hyperparameters of the XGBoost introduced in Section 2.3.1, applying the multi-objective optimization algorithm NSGA-II presented in Section 2.3.3. The objective of the optimization is the research of high precision modeling while ensuring generalization on new data.

Several HPOs have been already applied on boosting-based algorithms. For example, Grid Search in [99] and Random Search in [100]. More advanced techniques like Bayesian optimization have been also used for HPO of boosting machines in [101, 102]. However, the latter assumes independency among the hyperparameters, which does not hold for boosting machines (see Regularization in Section 2.3.1), limiting a complete parameter-free hyperparameter space exploration. In [103] the more recently proposed Hyperband has been applied as HPO for XGBoost. Finally, also GAs have been experimented in this context in [104, 105]. In particular, [104] applies the GA for GBM regression problems considering a population size of 16 individuals and 30 generations in order to minimize the modeling error. Furthermore, the hyperparameter space explored by this GA application is predefined within discrete values for some hyperparameters, precluding a parameter-free application. In [105], the NSGA-II has been applied to optimize XGBoost hyperparameters in a classification problem, including the multi-objective optimization of several accuracy criteria. In this case, neither information regarding the NSGA-II settings nor the considered hyperparameter search space are provided.

In this chapter, the proposed XGB-NSGAII is introduced. First, the workflow as well as the implementation of the optimization algorithm is presented. Second, the optimization problem defined to select the best models in the HPO is described. Finally, the analysis of the XGB-NSGAII performance and the

required resources for different *genetic settings*, i.e. the number of generations and the population size, are reported together with a comparison with other state-of-the-art HPOs.

4.1 Optimization Algorithm

As follows, the workflow and the implementation of the hyperparameter optimization algorithm XGB-NSGAII are presented.

4.1.1 Workflow

The XGB-NSGAII combines the hyperparameters validation reported in Figure 2.10 with the evolutionary algorithm workflow represented in Figure 2.11, applied with the XGBoost and with the NSGA-II respectively. The optimization starts with an initial random population of hyperparameter sets. At each generation of the GA, new sets are derived with the genetic operators and validated through the CV on the training data within a distributed system. Finally, the best hyperparameters are evaluated on the test data. The workflow of the developed XGB-NSGAII is divided into four steps: *initialization*, *distributed modeling*, *genetic operators* and *model selection*. The flowchart of this approach is reported in Figure 4.1 and the mentioned steps are described below.

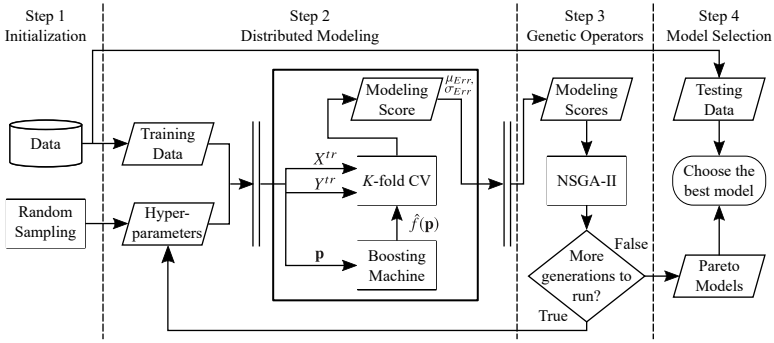


Figure 4.1: Workflow of the XGB-NSGAII divided into: initialization, distributed modeling, genetic operators and model selection.

Step 1 - Initialization. Beside preparatory procedures such as the loading of the preprocessed data and the start of the distributed system, in this step the available observations are split into training and test data. Moreover, the hyperparameter sets corresponding to the initial population are selected with a random uniform sampling. This corresponds to a Random Search (see Section 2.3.3) which is considered a useful method for initializing searching processes [74]. In particular, H hyperparameter sets $\{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_H}\}$ are generated within the predefined boundaries of the hyperparameter space introduced later in this section. The number of hyperparameter sets H corresponds to the population size. In addition, two specific hyperparameter sets are added to the randomly sampled ones in order to increase the efficiency of the HPO. These correspond to the default hyperparameters of the XGBoost and a variant of it, which includes a larger number of estimators, a deeper maximum tree size and a lower learning rate.

Step 2 - Distributed Modeling. In the distributed modeling the exploration of the hyperparameter space takes place. The prediction capability of each hyperparameter set $\mathbf{p}$ is validated through the K-fold CV on the training data, referred from now on as (X^{tr}, Y^{tr}). Totally $H \times K$ models are trained in this step. For each hyperparameter set the mean μ_{Err} and the standard deviation σ_{Err} are computed on the K-fold CV scores, see (2.43) and (2.44) respectively.

Step 3 - Genetic Operators. The genetic operators of the NSGA-II are applied in order to generate new hyperparameter sets according to the optimization objectives (see Evolutionary Algorithms in Section 2.3.3). First, selection for reproduction and variation are applied to produce new potentially good hyperparameters from the current parent population, known as offspring. The latter are then evaluated within the distributed modeling from Step 2. Finally, the selection for replacement is performed in order to compare the offsprings with the current parent population and select the individuals belonging to the next population. This step is iterated for the given number of generations.

Step 4 - Model Selection. In the final step, the hyperparameter sets included in the Pareto front of the final population (see Figure 2.12b) are used to evaluate the test error e_{Err} on the test data. The final model selection consists of two steps: first, only the hyperparameter sets of the final population for which the error e_{Err} lies within $2 * \sigma_{Err}$ from the μ_{Err} are kept, i.e. for which:

$$\mu_{Err} - 2 * \sigma_{Err} < e_{Err} < \mu_{Err} + 2 * \sigma_{Err}. \tag{4.1}$$

Afterwards, out of the filtered hyperparameter sets, the one with a lower μ_{Err} is selected to train the final model.

4.1.2 Implementation

The implementation of XGB-NSGAII is based on three main Python libraries: Dask [106], Distributed Evolutionary Algorithms in Python (DEAP) [107] and the XGBoost [108]. Dask includes APIs enabling advanced parallelism and resource scaling for analytics. It is able to run algorithms independently from the hardware composition by efficiently breaking up large computations onto distributed systems. The available hardware is abstracted by a *scheduler* and a set of *workers*. The running algorithms communicate with the scheduler, which ensures a balanced computing load on all the workers. This abstracted system is robust to any hardware fault because workers can be dynamically restored. In this way, the perfect balance between required performance and available computational power is ensured either on a HPC or on a local machine. The XGB-NSGAII is developed such that each worker can run on a single node of the HPC or on a thread of a local machine. The DEAP library includes the main genetic operators and tools in order to create customized optimization algorithms to fit any specific problem. Hence, it is possible to combine the individual genetic operations with the Dask distributed system and the XGBoost. According to the amount of data and the genetic settings, the XGB-NSGAII can be scaled based on the available computational power.

In Figure 4.2, the structure of the implementation of the XGB-NSGAII is represented in terms of framework modules, external modules and interfaces; this is a partial detailed view of the UML reported in Figure 3.5, including connections among the different modules and interfaces. The structure is introduced as follows. The *Optimization Manager* acts as a central hub connecting the required modules and resources. This ensures the correct initialization of the procedure and the central management of possible errors. Through the *User Interface*, the user is able to indicate its preferences for the optimization, e.g. the data to model, the genetic settings, but also the required resources in terms of number of workers and memory to allocate for each computing node. The *Optimization Manager* first loads the data to model from the data warehouse through the *Storage* interface. Afterwards, the scheduler and the workers are started. As soon as everything is properly set up, the *XGB-NSGAII* module is run. The latter combines the NSGA-II operations from DEAP with the *models manager*, which is in charge of training and validating the XGBoost

models. An instance of the scheduler is provided to the *Models Manager* from the *Optimization Manager*, such that the former is able to communicate directly with the distributed system in order to submit the procedures regarding the models training and validation.

As introduced in Section 3.1, all the modules involved in the framework are independent from each other such that each of them can be substituted with a variation of it. The only requirement is that the interfaces are satisfied, i.e. that the information flow between the modules remains unaltered. For instance, the *XGB-NSGAII* module can be substituted by another optimization module. In this way, the Grid Search and Hyperband are integrated in the framework. Similarly, also the learning method or the storage system can be substituted.

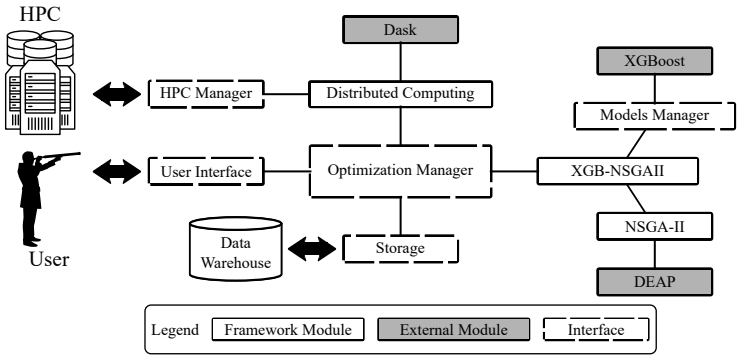


Figure 4.2: UML of the XGB-NSGAII optimization.

4.2 Optimization Problem

As already introduced in Section 2.3.3, the goal of a HPO consists in selecting the model able to achieve the best performance on the underlying data. In this section, the considered objectives and constraints as well as the hyperparameter space in order to achieve this goal are explained.

4.2.1 Objectives and Constraints

Single objective optimizations are able to focus only on the model performance in terms of either accuracy or resources. Adopting a multi-objective optimiza-

tion algorithm like the NSGA-II, it is possible to tune the hyperparameters in order to be robust and efficient, selecting the best model ensuring both accuracy and generalization on new data. This can be translated as the research of the minimum error, but at the same time be able to maintain this precision on data not included during the modeling phase. Thus, the error estimation does not depend on the portion of data where it is computed on. The considered evaluation metrics for the XGB-NSGAII are the MAE and the R^2, introduced in (2.40) and in (2.42) respectively. These scores are the results of the K-fold CV in the distributed modeling step. Consider μ_{MAE} the mean and σ_{MAE} the standard deviation computed on the MAE scores of the K folds; similarly, μ_{R^2} is the mean computed on the R^2 scores from the same procedure. The multi-objective optimization problem considered for the proposed HPO is based on three objectives:

1. minimize μ_{MAE} to achieve the best precision, i.e. to reduce the difference between predictions and ground truths;
2. minimize σ_{MAE} to increase the generalization on unknown data, i.e. the same MAE error has to be achieved on different portions of data;
3. maximize μ_{R^2} to strengthen the precision of the optimization with a global and unit independent score.

Furthermore, since a negative R^2 indicates an invalid model (see Section 2.3.2) μ_{R^2} is constrained during the genetic operations to be positive, i.e. solutions with negative R^2 would have a lower chance to be selected for the next generation. Each score is function of the training observations (X^{tr}, Y^{tr}) and of a hyperparameter set $\mathbf{p}$ sampled within a predefined hyperparameter space $[\mathbf{p}^{(l)}, \mathbf{p}^{(u)}]$. The optimization problem is summarized as follows:

$$\begin{aligned} \text{Minimize} \quad & \mu_{MAE}(X^{tr}, Y^{tr}, \mathbf{p}),\ \sigma_{MAE}(X^{tr}, Y^{tr}, \mathbf{p}) \\ \text{Maximize} \quad & \mu_{R^2}(X^{tr}, Y^{tr}, \mathbf{p}) \\ \text{subject to} \quad & \mu_{R^2}(X^{tr}, Y^{tr}, \mathbf{p}) > 0,\ \mathbf{p}^{(l)} \leq \mathbf{p} \leq \mathbf{p}^{(u)}. \end{aligned} \tag{4.2}$$

Although the MAE is chosen as metric for the HPO, the loss function considered to train the XGBoost models is the RMSE (see Section 2.3.2 for the difference between the evaluation metric and the loss function). Considering two different metrics for the internal and external optimization allows robust models and a fair comparison during the choice of the best hyperparameters. The RMSE is more sensitive to outliers, ensuring precise models. In contrast,

the MAE does not accentuate the difference of the error magnitude. Therefore, the latter is considered a valid inter-comparison metric with respect to other errors [109].

4.2.2 Hyperparameter Space

The hyperparameter space considered for the XGB-NSGAII is reported in Table 4.1. As introduced in Section 2.3.3, the hyperparameters are related mostly to the characteristics of the dataset and it is not possible to define them ahead. For instance, the hyperparameters representing the subsampling of features and observations depend on the number of significant variables in the dataset and on the distribution of the data. Similarly, the learning rate and the number of estimators depend on each other and on the complexity of the phenomena to model.

Table 4.1: Hyperparameters and their boundaries considered for the XGB-NSGAII.

Name	Description	Boundaries
Level Subsample s_{cl}	Feature subsampling at each level	[0.5, 1.0]
Node Subsample s_{cn}	Feature subsampling at each node	[0.5, 1.0]
Tree Subsample s_{ct}	Feature subsampling at each tree	[0.5, 1.0]
Data Subsample s_d	Observ. subsampling at each iteration	[0.5, 1.0]
Gamma γ	Minimum loss reduction for a leaf	[0.0, 1.0]
Regularization λ	L1 weights regularization	[0.0, 2.0]
Regularization α	L2 weights regularization	[0.0, 1.0]
Min. Child Weight ω_{mc}	Minimum weights sum in a leaf	[0.0, 10.0]
Learning Rate ϵ	Shrinkage contribution for each tree	[0.0, 1.0]
Number of Estimators M	Number of boosting iterations	[1, 1000]
Tree Max Depth δ_M	Maximum tree depth for base learners	[1, 15]

In order to apply the genetic operators in the HPO, some adaptations have to be performed on the hyperparameter space. The genetic operators work with binary or simulated-binary representations of the individuals, which are expected to be continuous. Therefore, the presence of mixed discrete and continuous hyperparameters requires additional steps before applying the genetic operators. To overcome this issue, the discrete hyperparameters are mapped as floating values to the intervals [0, 1] before applying the genetic operators. Afterwards, they are re-mapped to the original intervals for the modeling phase. The two discrete hyperparameters are the tree max depth and the number of estimators.

This operation allows a parameter-free HPO on the whole hyperparameter domain.

Genetic Regularization As introduced in Regularization in Section 2.3.1, a large number of estimators for the XGBoost may lead to overfitting. A common procedure to define the best number of required estimators is using the early stopping approach (see Section 2.3.3). In the proposed HPO the overfitting conditions are directly monitored by the genetic operators through the defined objectives. Overfitting situations are identified during the K-fold CV in case of large σ_{MAE}, which means that the hyperparameters are not able to generalize on new data. Overfitting solutions have a high chance to be neglected by the selection operator. Thus, only the best combination of hyperparameters survives the optimization and no early stopping is implemented in this work.

4.3 Results

In this section, the results concerning the application of the XGB-NSGAII as HPO are presented. First, the datasets considered for the experiments are introduced. Afterwards, the analysis of the XGB-NSGAII performance and the required resources for different genetic settings is reported. Finally, a comparison with other state-of-the-art HPOs is given.

4.3.1 Datasets

The four datasets chosen to investigate the potential of the XGB-NSGAII are reported in Table 4.2. Three of them correspond to Bosch proprietary datasets of different GDI characteristics and the fourth one to a synthetic public dataset. The selected datasets aim to represent different data typologies, e.g. measurement, simulation and synthetic, as well as different data sizes. The results of 1288 spray chamber experiments are considered to analyze the effect of 10 injector geometries and operating points on the spray width s_w. 3D-CFD simulations results of 500 variations of 8 spray targeting coordinates and injection strategies are analyzed in order to investigate the impinged spray mass m_s. Similarly, the effect of 9 injector geometries on fuel turbulent kinetic energy (TKE) k is analyzed with 591 simulation results. More details regarding these datasets are given in Sections 5 and 6. In order to show the universal capability of the XGB-NSGAII, the synthetic public Friedman dataset [68,110]

is considered, which is referred as *Fr*. The Bosch datasets are anonymized with the Z-score normalization due to confidentiality data policy (see Section 2.2.3).

Table 4.2: Datasets considered to investigate the XGB-NSGAII.

Variables	Source	Samples	Features
spray width s_w	Spray Experiments	1288	10
impinged spray mass m_s	Engine 3D-CFD	500	8
TKE k	Nozzle 3D-CFD	591	9
synthetic variable *Fr*	Friedman Dataset	500	10

4.3.2 Influence of the Genetic Settings

The most characterizing parameters of a GA are the size of the population and the number of generations. Typically, the choice of these genetic settings is limited by the available computing and storage resources, especially when the individuals evaluations are costly and time consuming. Based on the application context, the choice of the proper genetic settings may be different [111–113]. In this section, the influence of different population sizes and number of generations is analyzed on the proposed HPO in terms of required resources and optimization performance. For this work, populations of sizes 24, 48, 100, 200 combined with generations between 0 and 50 are investigated, as summarized in Table 4.3. The generation zero indicates the evaluation of the initial population.

Table 4.3: Investigated combinations of number of generations and population sizes.

Population Size	Number of Generations
24	[0-50]
48	[0-50]
100	[0-50]
200	[0-50]

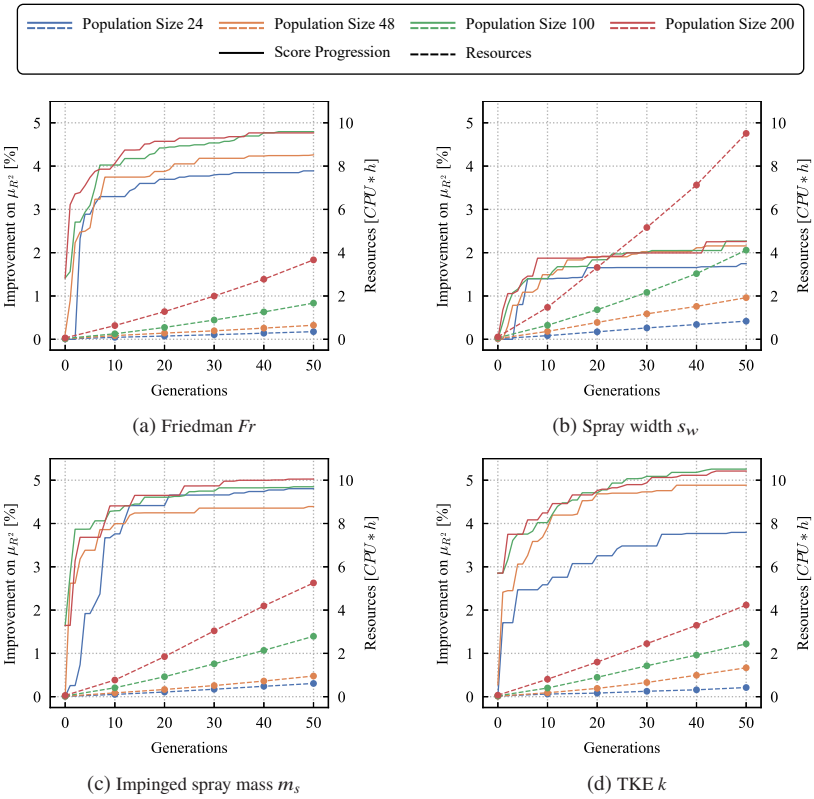
The modeling validation adopted for the investigation corresponds to a training-test split with 80% of the data for the GA and 20% for the test set, while the K of the K-fold CV is set equal to 5. All the results reported with this investigation are related to experiments run on the Bosch HPC through one Dask scheduler and 16 workers, each one running on a single CPU node.

Resource and Performance

The influence of the different genetic settings on the modeling performance and on the required resources for the datasets listed in Table 4.2 are depicted in Figure 4.3. The x-axes report the number of generations. The left y-axes indicate the percentage of improvement of μ_{R^2} at each generation of the considered variable. The latter corresponds to the improvement with respect to the smallest μ_{R^2} among the different population sizes found at the generation zero. The right y-axes represent the amount of resources required during the optimization process in terms of $(CPU * h)$ measured every 10 generations.

As expected from a typical optimization process, the scores tend to improve and the required computational resources to increase along the number of generations. In general, a larger population has higher chance to find better scores already at the generation zero. Within the first 10 generations, the scores show a collective strong improvement. After that, they converge at different speeds and levels, according to the population sizes and datasets. Some main patterns can be recognized: in most of the cases, a smaller population size converges to a poorer score. The only exception is for the impinged spray mass m_s, where the run with 24 individuals outperforms the case with a population size of 48. Generally, the population of 100 and 200 individuals converge to similar scores, except for the impinged spray mass m_s. In this case, a clear distinction between the two populations is present, i.e. the higher population size reaches the better score. For this reason, in order to ensure the best results for all the variables, a population of 200 individuals together with 50 generations is chosen in the frame of this work.

The number of models to train and to validate during the optimization increases with larger population sizes and number of generations. Thus, the required resources increase as well. Furthermore, particular hyperparameters of the XGBoost like the max depth δ_M, the learning rate ϵ and the number of estimators M influence the resources too. Therefore, the progress of the required $(CPU * h)$ may change its gradient along the generations according to the optimal hyperparameters found. In addition, the required resources depend also on the amount of data to model. Similar datasets sizes, such as the impinged spray mass m_s, the Friedman dataset Fr and the TKE k, require comparable resources. In contrast, the spray width s_w demands more resources considering its larger size (see Table 4.2).



(a) Friedman *Fr*

(b) Spray width s_w

(c) Impinged spray mass m_s

(d) TKE k

Figure 4.3: Progression of the improvement on R^2 and the required resources of the XGB-NSGAII on different datasets.

Optimized Hyperparameters

In the following analysis, the modeling of the TKE k dataset with 200 individuals and 50 generations is considered as illustration case. In Figure 4.4, the evolution of the scores μ_{MAE} and σ_{MAE} belonging to the Pareto fronts along different generations is depicted. In addition, part of the individuals generated during the optimization are reported as well. Since both R^2 and MAE follow the same behavior, the former is omitted for this analysis. *Utopia* indicates the desired target of the optimization, i.e. the point where all the objectives assume

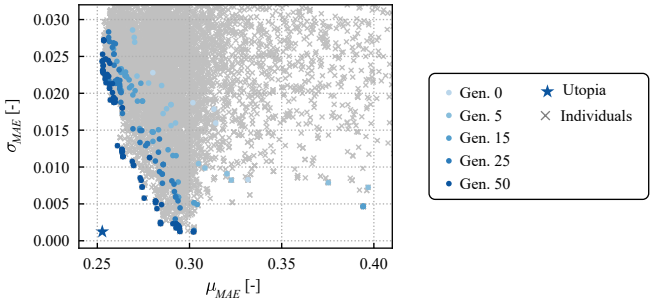
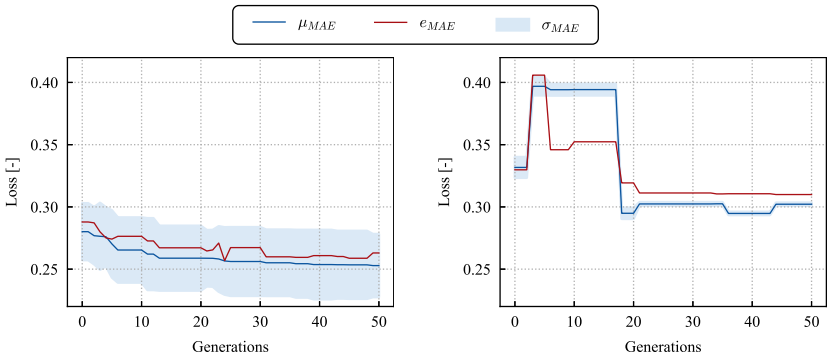


Figure 4.4: Evolution of the Pareto fronts of TKE k along the generations for the objectives μ_{MAE} and σ_{MAE}. Part of the generated individuals are also reported.

their optimum value. Along the generations, the genetic operators are able to discover new individuals producing scores closer to utopia.

In Figure 4.4, it is also possible to observe the contradictive nature of μ_{MAE} and σ_{MAE}. Considering the Pareto front of the final generation, the models able to achieve the minimum average error μ_{MAE} imply a bigger standard deviation σ_{MAE}. In contrast, the models reaching the minimum standard deviation σ_{MAE} have a higher average error μ_{MAE}. To describe this phenomena, the scores of the models achieving the smallest μ_{MAE} and σ_{MAE} in the Pareto front of each generation are depicted in Figure 4.5a and in Figure 4.5b respectively. On the one hand, a larger standard deviation corresponds to different results produced from each fold of the K-fold evaluation. This represents a more unstable model with a low average error, see Figure 4.5a. On the other hand, a smaller standard deviation indicates a stable model producing similar results on different portions of data. This may represent a more conservative model with a large average error, see Figure 4.5b. In Figure 4.5, the error e_{MAE} on the test data is reported as well. This follows the average error μ_{MAE} in both cases indicating the absence of overfitting for the optimized hyperparameters. The contrasting objectives and the application of the NSGA-II develop a broad final Pareto front, from which the best model can be selected (see Step 4 - Model Selection in Section 4.1.1).



(a) Minimum μ_{MAE} on each Pareto front.

(b) Minimum σ_{MAE} on each Pareto front.

Figure 4.5: Development of the best TKE k modeling score along the generations.

In Figure 4.6, the optimal hyperparameter combinations discovered during the XGB-NSGAII are represented for each dataset in Table 4.2. Additionally, the default values of the hyperparameters of the XGBoost[1] are included in the comparison. In the parallel plot, each vertical axis corresponds to a hyperparameter and the lines indicates the optimized hyperparameter values for the considered datasets as well as the default hyperparameter values. The identified hyperparameters are generally different for each modeled dataset and from the default hyperparameters. The feature subsampling s_{cl}, s_{cn}, s_{ct} tend to assume large values. Similarly, the number of estimators M, the learning rate ϵ and gamma γ do not show a large variation in the hyperparameter space. The optimized solutions present a large number of estimators M, which require low learning rates ϵ (see Regularization in Section 2.3.1). The rest of hyperparameters assumes values on wider intervals of the considered hyperparameter space.

These results confirm the necessity of adapting each model to the underlying data through the HPO. Therefore, the automated, parameter free, dynamic and data-driven model selection presented is the proper approach for the heterogeneous datasets available in the GDI context.

[1] Python Scikit-Learn Wrapper Interface for XGBoost v. 1.0.2 [108]

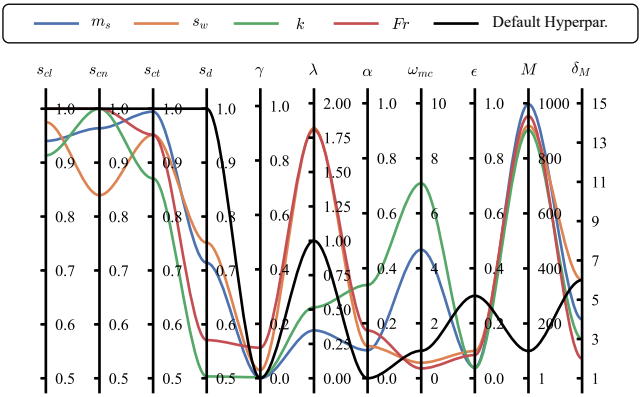


Figure 4.6: Parallel plot of the optimal hyperparameters for each investigated dataset.

4.3.3 Benchmark with the State-of-the-Art

The proposed XGB-NSGAII is compared with different state-of-the-art HPOs, which are: Random Search, Grid Search and Hyperband, all introduced in Section 2.3.3. In addition, the default hyperparameters of the XGBoost are included in the comparison in order to represent a no-HPO case. The considered hyperparameter space is the one defined in Table 4.1. The optimization settings adopted for each of these methods are reported as follows.

XGB-NSGAII. A population size of 200 and 50 generations are chosen. Although this combination of genetic settings requires a higher amount of resources, it has been shown that it is able to reach optimal results.

Random Search. Through a random uniform sampling, 200 hyperparameter sets are drawn from the hyperparameter space.

Grid Search. The combination of the first, the second and the third quantile of the hyperparameter space is considered to achieve a sparse selection.

Hyperband. The number of estimators M is set as the resource to be optimized. The other hyperparameters are sampled from a random uniform distribution of the hyperparameter space. The range of resources allowed, i.e. the maximum number of estimators, is conform to the hyperparameter space considered for the other HPOs.

The validation of the models is performed through a K-fold CV with K equals to 5 and a training-test split corresponding to $(80-20)\%$ for all the approaches. The selection of the best models is based on the lowest μ_{MAE} achieved, except for the XGB-NSGAII. For the latter the best model selection is described in Step 4 - Model Selection in Section 4.1.1.

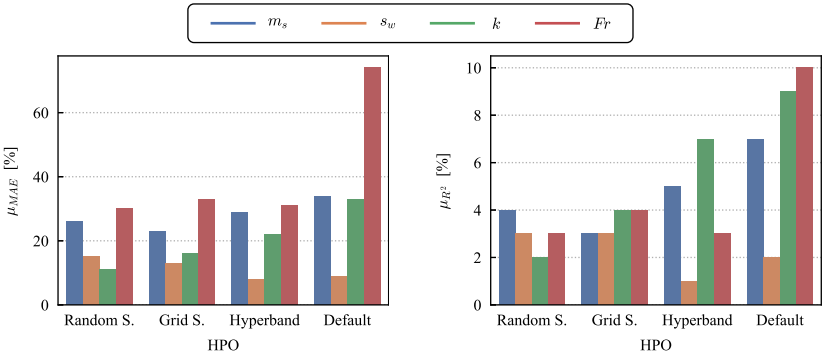
In Table 4.4, the scores and the performance of the considered HPOs as well as of the default hyperparameters are listed for the datasets in Table 4.2. The reported scores are the three objectives considered for the XGB-NSGAII, i.e. μ_{MAE}, σ_{MAE} and μ_{R^2}, as well as the test score e_{MAE}. Furthermore, the required computational resources ($CPU * h$), the consumed average RAM pro CPU in GigaByte (GB) and the total amount of evaluated models for each method are reported also in the table.

Table 4.4: Benchmark of XGB-NSGAII with other HPO methods.

Data	HPO	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	$CPU * h$	mem. [GB]	models
	XGB-NSGAII	**0.885**	**0.253**	0.024	0.274	4.23	6.71	10 202
	Random Search	0.864	0.280	0.023	0.288	0.07	3.54	200
k	Grid Search	0.849	0.294	0.027	0.286	34.49	28.00	177 147
	Hyperband	0.826	0.307	0.033	0.343	**0.04**	**2.40**	1663
	Default	0.801	0.336	0.026	0.351	-	-	1
	XGB-NSGAII	**0.876**	**0.242**	0.014	0.237	9.51	10.05	10 202
	Random Search	0.853	0.277	0.020	0.274	0.11	2.91	200
s_w	Grid Search	0.850	0.273	0.012	0.277	52.96	29.80	177 147
	Hyperband	0.866	0.261	0.016	0.251	**0.06**	**2.43**	1663
	Default	0.862	0.264	0.010	0.262	-	-	1
	XGB-NSGAII	**0.926**	**0.189**	0.019	0.175	5.25	6.90	10 202
	Random Search	0.886	0.237	0.029	0.238	0.06	2.78	200
m_s	Grid Search	0.898	0.233	0.025	0.221	41.27	27.34	177 147
	Hyperband	0.884	0.245	0.025	0.187	**0.05**	**2.41**	1663
	Default	0.866	0.253	0.025	0.220	-	-	1
	XGB-NSGAII	**0.957**	**0.803**	0.083	0.711	3.67	5.76	10 202
	Random Search	0.924	1.042	0.103	0.821	0.06	2.71	200
Fr	Grid Search	0.917	1.065	0.078	0.842	35.55	28.30	177 147
	Hyperband	0.927	1.049	0.075	0.947	**0.04**	**2.39**	1663
	Default	0.864	1.397	0.156	1.451	-	-	1

In the following analysis, the μ_{MAE} and the μ_{R^2} are considered first. Starting from the simplest method, the default hyperparameters are able to provide good models for all the variables. However, these are generally weaker with respect

to the other approaches. Grid Search demands a huge amount of resources due to the extremely large number of models to compute. Since Grid Search performs just similar to other state-of-the-art HPOs, the demanded resources do not justify the application of this approach. Random Search and Hyperband are comparable in terms of resources. The amount of explored models is larger for the Hyperband but the performance improvement is not substantially different with respect to the Random Search. Only for TKE k the latter largely outperforms the Hyperband. It is not possible to identify the best approach among Random Search, Grid Search and Hyperband, except for the required resources and the exploration of the hyperparameter space. In contrast, the proposed XGB-NSGAII provides the best scores in every case. In Figure 4.7, the percentage of the improvement achieved by the XGB-NSGAII on μ_{MAE} and μ_{R^2} with respect to the other HPOs is reported for every considered dataset. In general, the XGB-NSGAII ensures an improvement between 8% and 74% for the μ_{MAE} with an average of 25%. Similarly, it reaches an improvement up to 10% for the μ_{R^2} with an average of 4%. As expected, large improvements are achieved against the default hyperparameters, since these are part of the initial population of the XGB-NSGAII.



(a) Improvement of MAE.

(b) Improvement of R^2.

Figure 4.7: Improvement of the scores provided by XGB-NSGAII compared to other HPOs.

Due to the larger amount of computed models, the proposed HPO requires more resources with respect to Random Search and Hyperband. As previously demonstrated, the amount of resources demanded by the XGB-NSGAII is

sensible to the dataset size. This effect is less present for the Hyperband, even when compared with the Random Search.

Beside the high precision, also the generalization on unknown data is successfully achieved with the XGB-NSGAII. Although its values of the σ_{MAE} in Table 4.4 are not always the lowest compared to the other HPOs, contextualizing these with the other errors, the goal of the optimization problem is successfully achieved: the test error e_{MAE} is contained for every dataset within the interval $[\mu_{MAE} \pm 2\sigma_{MAE}]$. This indicates that the standard deviation provided by the XGB-NSGAII can be considered as a meaningful indicator for the expected error on new data and that the hyperparameter search did not overfit the training data.

4.4 Summary

In this chapter, an automated, parameter-free, dynamic and data-driven model selection for GDI data is presented. In particular, the potential of coupling XGBoost with the optimization algorithm NSGA-II is shown in order to choose the best set of hyperparameters ensuring good accuracy and generalization for regression modeling. Compared to the previously published studies, the XGB-NSGAII includes proper objectives to not only focus on the optimization of the XGBoost accuracy, but also on its ability to generalize on new data. In addition, the explored hyperparameter space is continuous and highly multidimensional, including more hyperparameters. GAs are heuristic global search methods requiring high computational resources to find optimal solutions. Therefore, a distributed computation is part of the proposed HPO. The optimization performance of the novel approach is compared against default hyperparameters and state-of-the-art methods, like Random Search, Grid Search and Hyperband. Totally, three datasets from the GDI development and one public are considered for the investigation. The optimization objectives chosen for the NSGA-II, combined with a large population size and number of generations showed a higher accuracy with respect to the other HPOs. However, the resources demanded from XGB-NSGAII depend on the dataset size and are higher with respect to Random Search and Hyperband. Nevertheless, in the context of GDI development, the additional computational resources are not significant compared to the expensive generation of the data. The presented approach is not restricted to combustion engine development, but it is shown that it can be extended to other data and applications.

5 Injector Nozzle Analysis

In this chapter, the KD framework is applied to analyze the effects of injector nozzle geometries and engine operating points on flow dynamics and spray characteristics. The analysis is performed through numerical investigations and experiments. In Figure 5.1, the schematic geometry of the injector valve seat is represented. The latter corresponds to the tip of the injector presented in Figure 1.3. The fuel enters the combustion chamber through the injector holes as the needle moves upwards. This movement is parameterized by the needle lift height nl_h. The spray hole axis is defined by the inclination angle α and the circumferential angle β. Along this axis, the spray hole and the pre-hole can be identified, characterized by the lengths sh_l and ph_l as well as by the diameters sh_d and ph_d respectively. The sum of the two holes lengths is equal to the wall thickness wt_l. The spray hole is further parameterized with the spray hole conicity Ψ and the spray hole outlet diameter $sh_{d,o}$, while the valve seat with the circumferential step height st_h.

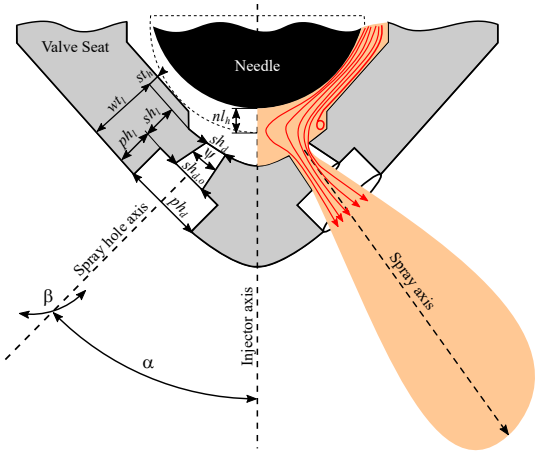


Figure 5.1: Injector nozzle geometry, from [114].

5.1 Numerical Analysis of Inner Flow

In order to investigate the nozzle inflow characteristics through the KD framework, the data produced to initialize the optimization of the GDI high-pressure injector valve seat in [114] are considered. For this work, the data are normalized with the Z-score method (see Section 2.2.3) due to confidentiality data policy. A general presentation of the data is given as follows, based on [114]. The dataset represents a DOE of 700 geometry variations sampled using the Sobol approach (see Section 3.2.1). Totally, nine of the design parameters introduced in Figure 5.1 are analyzed, which are summarized as follows:

$$X = \left(\; st_h \, , \; \beta \, , \; \alpha \, , \; nl_h \, , \; ph_d \, , \; \Psi \, , \; sh_d \, , \; sh_l \, , \; wt_l \; \right). \tag{5.1}$$

Furthermore, two additional parameters are taken into account to define constraints in order to ensure manufacturing feasibility. These are the pre-hole length ph_l, corresponding to the difference between the wall thickness wt_l and the spray hole length sh_l, and the spray hole outlet diameter $sh_{d,o}$. The constraints are listed in Definition 5.1.1.

Definition 5.1.1. DOE constraints applied for the nozzle numerical analysis:

- C_1: minimum pre-hole length $\rightarrow ph_l = wt_l - sh_l \geq c_1$;
- C_2: pre-hole diameter larger than the sum of the spray hole outlet diameter and a tolerance $\rightarrow ph_d \geq sh_{d,o} + c_2$;
- C_3: minimum spray hole outlet diameter $\rightarrow sh_{d,o} \geq c_3$.

The 700 sampled designs are evaluated with an automated 3D-CFD workflow. The numerical analysis required ~520.000 $CPU * h$ corresponding to 75 computation days on a HPC. Due to geometrical constraints violation and simulation failures, out of the 700 geometry variations, 592 designs generated valid results, which are the datapoints considered for the analysis.

A single 3D-CFD simulation generates a large amount of flow dynamics and spray information. In this work, five spray and inflow characteristics are analyzed: the fuel massflow $\dot{m}$, the spray plume angle τ, the TKE k, the spray targeting radius r_t and the fuel plume penetration p_p, which are summarized as follows:

$$Y = \left(\; \dot{m} \, , \; \tau \, , \; k \, , \; r_t \, , \; p_p \; \right). \tag{5.2}$$

Except the TKE k, the investigated characteristics are depicted in Figure 5.2. The TKE k indicates the fuel atomization quality depending on the turbulence level of the flow. Both k and τ are calculated at the pre-hole outlet. The quantities are spatially and temporally averaged along the considered surface and within the simulated injection time, i.e. time and spacial series are ported to scalar values (see Data Type Portability in Section 2.2.3). The analyzed spray and inflow characteristics are critical for the atomization process; hence, they are strongly related to combustion efficiency and emissions. For instance, a small spray plume angle τ ensures a robust spray shape, while a large TKE k allows a good fuel atomization. A large fuel plume penetration p_p may increase the fuel impingement on the combustion chamber surfaces, which may cause a lower air-fuel homogenization level and lead to higher consumption and emissions. The fuel massflow $\dot{m}$ and the spray targeting radius r_t have to be instead adapted to the required engine and application specifications. Being able to control and to deeper understand these characteristics would positively contribute to a further improvement of the valve seat design.

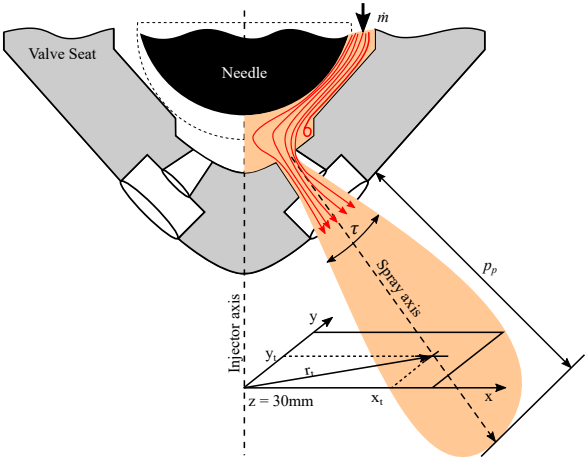


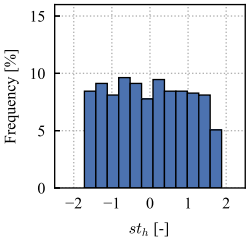
Figure 5.2: Investigated nozzle inflow and spray characteristics, from [114].

5.1.1 Data Exploration and Preprocessing

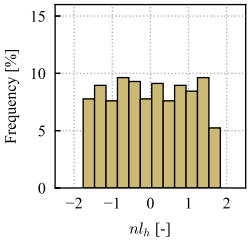
In this section, the exploration and the preprocessing steps of the data are reported. Part of the data analysis is based on [114].

Input Analysis

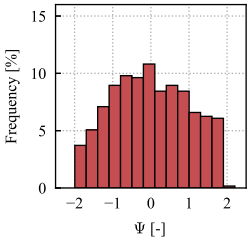
In Figure 5.3, the most characterizing data distributions of the sampled geometry parameters are represented. The rest of the distributions are reported in Appendix A.1.1. Since the Sobol-sequence has been adopted for the DOE, the data distributions are supposed to be uniform. This is the case of the circumferential step height st_h and the needle lift height nl_h. However, due to the constraints defined in Definition 5.1.1, the distribution of the spray hole conicity Ψ, the spray hole length sh_l, the pre-hole diameter ph_d and the wall thickness wt_l are not uniform. In order to analyze deeper the possible effects



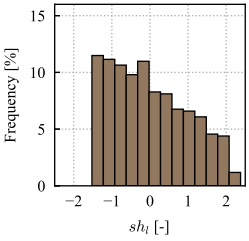
(a) Distribution of the circumferential step height st_h.



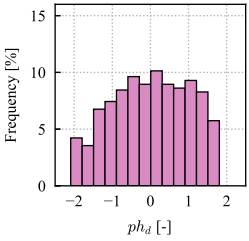
(b) Distribution of the needle lift height nl_h.



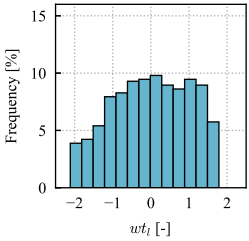
(c) Distribution of the spray hole conicity Ψ.



(d) Distribution of the spray hole length sh_l.



(e) Distribution of the pre-hole diameter ph_d.



(f) Distribution of the wall thickness wt_l.

Figure 5.3: Most characterizing data distributions of the input variables.

of these constraints on the modeling performance, it is necessary to investigate the presence of multicollinearity (see Section 2.2.4).

In Figure 5.4, the correlation matrix for the input variables is reported.

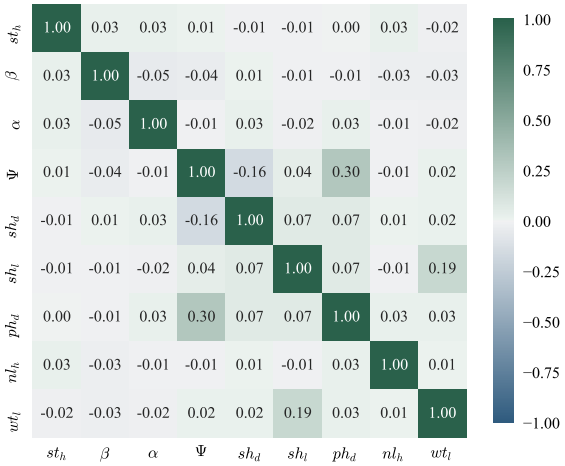


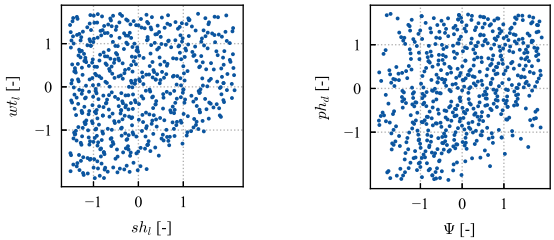
Figure 5.4: Correlation matrix of the explanatory variables.

The correlations are generally weak and can be attributed to spurious correlations due to the sampling constraints:

- $C1$ ensures a minimum difference between wall thickness wt_l and spray hole length sh_l, which results into a slightly positive correlation. The latter can be clearly seen in Figure 5.5a, where a projection on wt_l and sh_l of the multi-dimensional sampled domain is depicted;
- $C2$ and $C3$ are responsible for the correlations of the spray hole conicity Ψ with the spray hole diameter sh_d and the pre-hole diameter ph_d. In this case, the correlation is due to an intermediate variable: the spray hole outlet diameter $sh_{d,o}$. The latter is influenced by the spray hole conicity Ψ, which consequently limits the pre-hole diameter ph_d based on C_2, as represented in Figure 5.5b. The constraint C_3 ensures a minimum value for the spray hole outlet diameter $sh_{d,o}$, which reduces the amount of designs with small spray hole conicity Ψ and spray hole diameter sh_d, as shown in Figure 5.5c.

For completeness, a case of unconstrained sampling referred to the needle lift height nl_h and the circumferential step height st_h is reported in Figure 5.5d.

The correlations are moderate and they are caused by the imposed constraints. Therefore, all the sampled variables can be used for the modeling. If necessary, the correlations have to be taken into account during the models interpretation, since the constraints may influence the extracted knowledge.



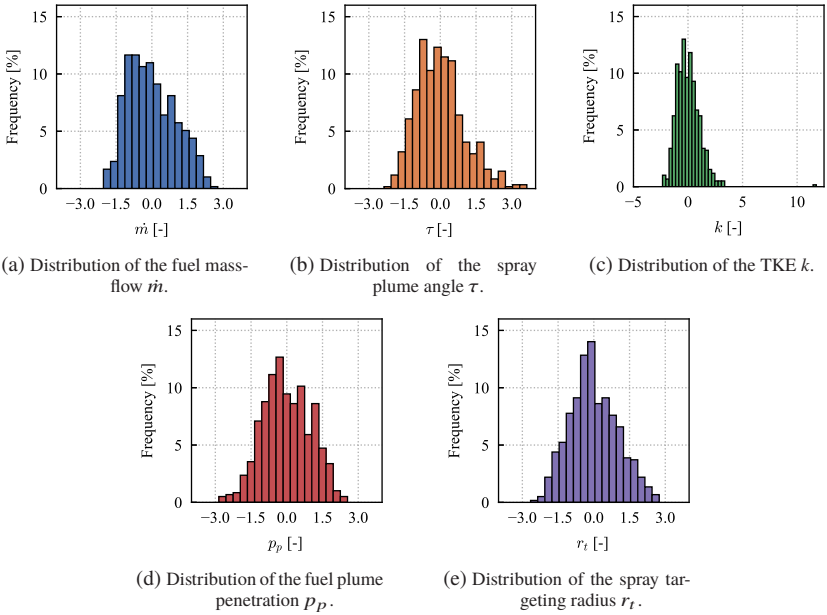
(a) Constraint C_1. (b) Constraint C_2.

(c) Constraint C_3. (d) Unconstrained sampling.

Figure 5.5: Examples of constrained and unconstrained samplings, following Definition 5.1.1.

Output Analysis

The distribution of the considered spray and inflow characteristics resulting from 3D-CFD simulations are reported in Figure 5.6. All the characteristics tend to assume a normal distribution, without highlighting any particular anomaly in the results, except for TKE k. In this case, a small percentage of data falls far away from the rest, which may corresponds to an anomaly. As follows, the outlier analysis is performed using the Chebyshev outlier detection and the Isolation Forest (see Outlier detection in Section 2.2.2). First, stricter



(a) Distribution of the fuel mass-flow $\dot{m}$.

(b) Distribution of the spray plume angle τ.

(c) Distribution of the TKE k.

(d) Distribution of the fuel plume penetration p_p.

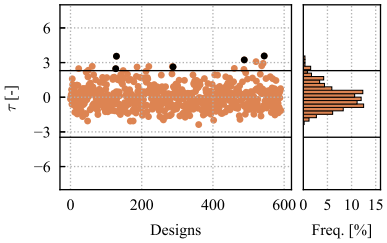
(e) Distribution of the spray targeting radius r_t.

Figure 5.6: Distribution of the investigated 3D-CFD nozzle characteristics.

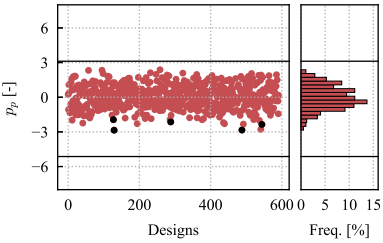
outlier detection settings are set and then, these are adapted to the underlying data. Only the variables with more potential outliers are reported.

Strict Outlier Detection. In Figure 5.7, the results concerning the Chebyshev algorithm applied with p_1 and p_2 chosen as 0.1 and 0.05 and the Isolation Forest with a score threshold of -0.6 are depicted. Each figure is composed of two parts. The y-axis indicates in both cases the considered output characteristic. On the right side, the x-axis represent the analyzed designs, while on the left side, the distribution of the output. Considering the boundaries of the Chebyshev outlier detection, anomalies are recognized only for TKE k and spray plume angle τ, including observations belonging to the very end of their distribution tails. In the case of Isolation Forest, it is not possible to direct track back the variables that caused the points to be considered outliers, due to the stochastic operations run during the outliers search. Although these points are far away from the rest, most of them are still plausible. For this

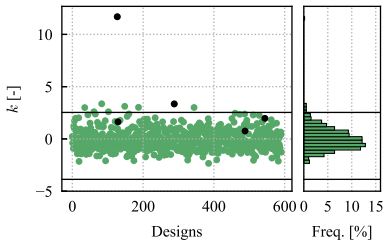
reason, adjusting the settings of the outlier detection methods may increase their cleaning efficiency.



(a) Outlier detection on spray plume angle τ



(b) Outlier detection on fuel plume penetration p_p



(c) Outlier detection on TKE k

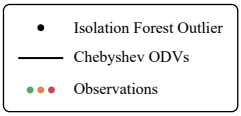
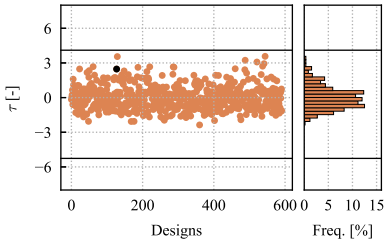


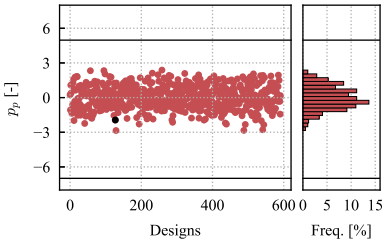
Figure 5.7: Outliers detection results adopting strict settings.

Optimal Outlier Detection. In Figure 5.8, the results regarding the outlier detection with adjusted settings in order to adapt the methods to the data are reported. In this case, the Chebyshev algorithm is applied with p_1 equals to 0.05 and p_2 to 0.025. The Isolation Forest score threshold is slightly decreased to −0.65. These small variations allow the algorithms to become less severe with normal observations, but they are still able to catch the single potential anomaly in TKE k.

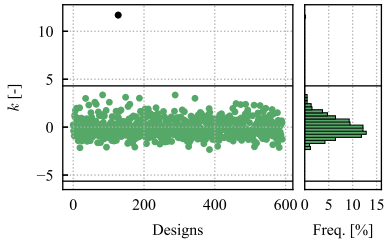
Considering the results of the optimal outlier detection, the amount of outliers is not significant with respect to the total dataset size. Therefore, the single point highlighted from the optimal outlier detection is neglected for the next KD steps and no further reconstruction procedures are performed. Although boosting algorithms are robust against outliers, the presence of wrong observations in the data has to be avoided in order to ensure a correct knowledge extraction.



(a) Outlier detection on spray plume angle τ



(b) Outlier detection on fuel plume penetration p_p



(c) Outlier detection on TKE k

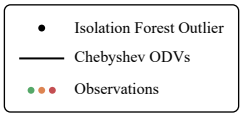


Figure 5.8: Outliers detection results adopting optimal settings.

5.1.2 Model Selection and Validation

The preprocessed dataset containing 591 observations and 9 features can be used for the modeling step. The XGB-NSGAII and the polynomial regression are considered in order to compare the parametric and the non-parametric learning. In both cases, a 5-fold CV and $(80-20)\%$ training-test split are applied to compute the scores. The XGB-NSGAII models are selected with 50 generations and a population of 200 individuals. The polynomials degrees are chosen as the degrees providing the best average MAE in the 5-fold CV.

In Table 5.1, the scores resulting from the modeling methods are reported in terms of μ_{R^2}, μ_{MAE}, σ_{MAE} and e_{MAE}. In most of the cases, the performance of the two learning methods are equivalent. For the fuel massflow $\dot{m}$, the spray plume angle τ and the spray targeting radius r_t, the polynomial modeling is able to provide slight higher μ_{R^2}, even though a better μ_{MAE} is achieved only for the fuel massflow $\dot{m}$. The decision-tree based modeling is able to largely outperform the polynomial for TKE k and fuel plume penetration p_p. In particular, the XGBoost performance on TKE k improves of 6% for the

μ_{R^2} and about 19% for the μ_{MAE} with respect to the polynomial. Similarly, μ_{R^2} improves of 2% and μ_{MAE} of 5% for the modeling of the fuel plume penetration p_p.

Table 5.1: Modeling scores of the 3D-CFD nozzle analysis.

	XGBoost				Polynomial				
Data	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	degree
$\dot{m}$	0.989	0.074	0.008	0.066	**0.990**	**0.069**	0.006	0.058	3
τ	0.786	**0.329**	0.040	0.339	**0.789**	0.339	0.025	0.325	2
k	**0.885**	**0.253**	0.024	0.274	0.834	0.311	0.024	0.322	3
p_p	**0.782**	**0.363**	0.020	0.378	0.767	0.380	0.023	0.373	2
r_t	0.908	**0.212**	0.024	0.236	**0.915**	0.215	0.007	0.206	2

The results obtained through the XGBoost ensure that the models are able to capture the physics behind the data. The lowest μ_{R^2} is 0.782 achieved by the fuel plume penetration p_p; right after it, there are the spray plume angle τ with 0.786 and the TKE k with 0.885. The rest of the variables are modeled with a μ_{R^2} larger than 0.9.

In Figure 5.9, the μ_{MAE}, the σ_{MAE} and the e_{MAE} are depicted for each model. The test error is always in the interval $[\mu_{MAE} \pm 2\sigma_{MAE}]$, which ensures the reliability of the models and excludes overfitting. This means that the XGBoost models are able to provide a good generalization on unknown datapoints.

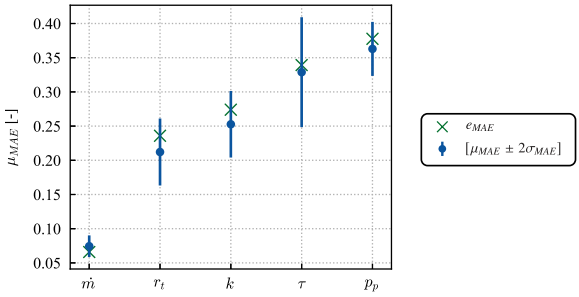


Figure 5.9: Evaluation of the XGBoost modeling quality in terms of the scores μ_{MAE}, σ_{MAE} and e_{MAE} from the 3D-CFD nozzle analysis.

5.1.3 Knowledge Extraction

In this section, the knowledge extraction for the analysis of the GDI high-pressure injector valve seat is performed in terms of models exploration and models exploitation.

Models Exploration

Once the robust and precise XGBoost models are available, the model exploration based on feature importance and partial dependencies can be performed (Section 2.4). In Figure 5.10, the relative feature importance retrieved during the modeling of the five inflow and spray characteristics are depicted. On the x-axis the investigated valve seat geometries are listed, while on the y-axis their relative feature importance on each modeled output is reported. According to the feature importance, it is possible to determine the geometries with a lower effect on the outputs. In particular, these are the circumferential step height st_h, the circumferential angle β and the pre-hole diameter ph_d, with an importance lower than 0.1. In contrast, one of the key geometries is the needle lift height nl_h, which influences most of the variables. The spray targeting radius r_t is characterized principally by the inclination angle α. The fuel plume penetration p_p is dominated by the needle lift height nl_h with the highest influence achieved with respect to the other geometries and characteristics. The fuel massflow $\dot{m}$ is influenced mostly by the spray hole diameter sh_d. Compared to

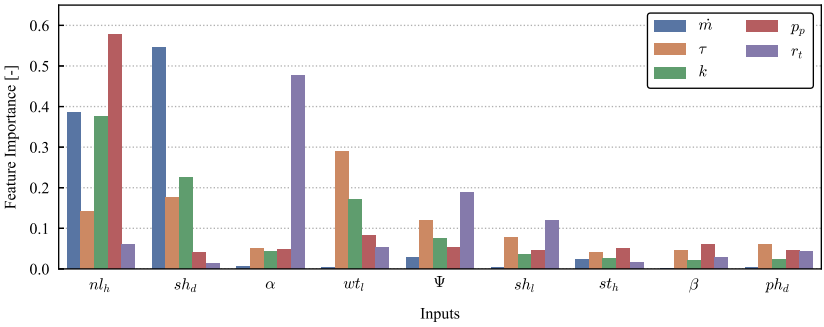


Figure 5.10: Feature importance of the 3D-CFD nozzle analysis.

the other outputs, the spray plume angle τ and the TKE k do not have a single characterizing geometry.

A connection between the variables magnitude of influence and the modeling performance reported in Table 5.1 can be established. In particular, the less complex phenomena, i.e. the ones that can be described with less input variables, are able to achieve higher modeling precision. For instance, the fuel massflow $\dot{m}$ can be modeled with almost 99% of accuracy and it is identified by the main interaction of two variables, i.e. nl_h and sh_d. In contrast, the variables influenced by many more geometries achieve lower performance. This behavior can be associated to the curse of dimensionality (Section 2.2.4): a higher number of relevant dimensions requires a larger amount of observations to correctly describe the output. Nevertheless, the irreducible error introduced in (2.6) may still affect the accuracy upper limit, independently by the amount of collected observations.

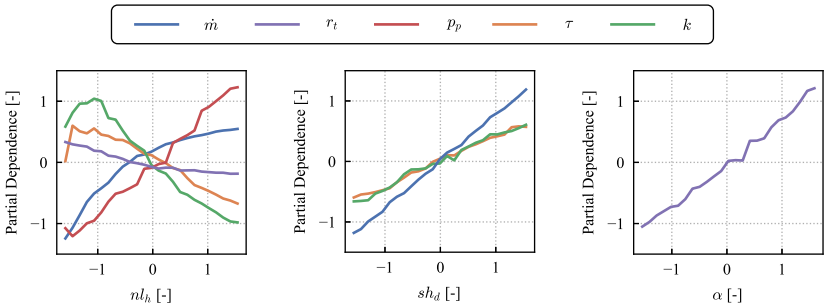
The single effects of the most influencing nozzle geometries nl_h, sh_d and α are represented through the partial dependencies in Figure 5.11 for the corresponding most affected outputs. The complete partial dependencies are reported in Appendix A.1.2. On the x-axes are reported the geometry variations and on the y-axes their effect on the investigated characteristics. It is important to keep in mind that all the data are normalized, therefore the y-axes indicates only the variations of the estimate function. Furthermore, the partial dependencies indicate the general behavior of the outputs and not the punctual response. The partial dependencies are explained and validated through the domain knowledge as follows.

Needle lift height nl_h. The influence of nl_h on the five investigated characteristics is reported in Figure 5.11a. A large nl_h allows more fuel massflow $\dot{m}$ through the valve. This holds until a certain threshold is achieved. After that, $\dot{m}$ tends to converge, indicating that the throttling does not longer depend on this parameter. Considering that the nl_h corresponds to the cross-section between the nozzle valve seat and the needle, a large nl_h allows the fluid to flow with lower resistance and velocity. This produces lower TKE k and spray plume angle τ. For a very low nl_h, the desired massflow cannot be provided anymore, therefore, k and τ decreases as well. Finally, the fuel plume penetration p_p has a positive relation with nl_h. This is due to the effect of the latter on $\dot{m}$ and τ: a large $\dot{m}$ and a small τ correspond to a large fuel plume penetration p_p.

Spray hole diameter sh_d. The sh_d shows almost a linear relationship with the fuel massflow $\dot{m}$, the TKE k and the spray plume angle τ, as depicted

in Figure 5.11b. This behavior results from the fact that a larger sh_d allows more $\dot{m}$, thus, the radial as well as the axial flow momentum increase, which lead to higher k and τ. The effect of sh_d on fuel plume penetration p_p and spray targeting radius r_t are omitted because negligible.

Inclination angle α. The effects of the variation of the α are depicted in Figure 5.11c. The parameter α controls the spray plume direction. Therefore, it has a positive effect on the spray targeting radius r_t. The effect of α on the other outputs are omitted because negligible.



(a) Partial dependence of needle lift height nl_h on all the analyzed outputs.

(b) Partial dependence of spray hole diameter sh_d on $\dot{m}$, τ and k.

(c) Partial dependence of inclination angle α on r_t.

Figure 5.11: Partial dependencies of needle lift height nl_h, spray hole diameter sh_d and inclination angle α for the respective most influenced inflow and spray characteristics.

Models Exploitation

Typically, it is required to set the spray targeting radius r_t and the fuel mass-flow $\dot{m}$ according to specific engine requirements. Moreover, a small spray plume angle τ to ensure a robust spray shape, a large TKE k to achieve a good fuel atomization and a small fuel plume penetration p_p to decrease the spray impingement are preferred. As described in the previous section, the geometry parameters may have different influences on the output characteristics. Therefore, the definition of a valve seat design able to achieve predefined inflow and spray characteristics corresponds most of the time to a trade-off among different requirements. For instance, a small needle lift height nl_h produces a small

fuel plume penetration p_p as well as a large TKE k, but at the same time also a large spray plume angle τ. The design definition becomes a multi-objective optimization problem with contradictive objectives.

As an illustrative example, consider that it is required to design a valve seat such that the spray hole conicity Ψ, the wall thickness wt_l, the pre-hole diameter ph_d and the spray hole diameter sh_d are fixed a priori due to manufacturing constraints. Similarly, the fuel massflow $\dot{m}$ and the spray targeting radius r_t are fixed due to engine specifications. The XGBoost models $[f_\tau(\cdot), f_{p_p}(\cdot), f_k(\cdot), f_{\dot{m}}(\cdot), f_{r_t}(\cdot)]$ can be exploited in order to define the remaining parameters, while ensuring the smallest spray plume angle τ and fuel plume penetration p_p as well as the largest TKE k possible. The optimization problem is summarized as follows:

$$\begin{aligned}
\text{Minimize} \quad & f_\tau(X),\ f_{p_p}(X) \\
\text{Maximize} \quad & f_k(X) \\
\text{with} \quad & X = \left(nl_h,\ sh_d,\ \alpha,\ wt_l,\ \Psi,\ sh_l,\ st_h,\ \beta,\ ph_d\right) \\
\text{subject to} \quad & \text{fixed } \left(\Psi,\ wt_l,\ ph_d,\ sh_d\right), \\
& f_{\dot{m}}(X) = \dot{m}_0,\ f_{r_t}(X) = r_{t0}, \\
& X_i^{(l)} \leq X_i \leq X_i^{(u)},\ X_i \in \left[nl_h,\ \alpha,\ sh_l,\ st_h,\ \beta\right].
\end{aligned} \tag{5.3}$$

A new larger DOE of ten thousand designs satisfying the constraints defined in (5.3) is generated using the Sobol sampling. The new DOE is referred as X^s. The geometry parameters are sampled within the same boundaries of the original DOE, indicated in (5.3) with $[X_i^{(l)}, X_i^{(u)}]$. The low-resource demanding and high accuracy models are adopted to predict the inflow and spray characteristics of X^s. The set of predictions is indicated with Y^s. Afterwards, a fictive output Y^f containing the desired outputs is defined. Beside the constrained $\dot{m}_0$ and r_{t0}, the fictive output Y^f contains the minimum spray plume angle τ and fuel plume penetration p_p as well as the maximum TKE k resulting from the prediction of X^s. The fictive output Y^f is summarized as follows:

$$Y^f = \left(\dot{m}_0,\ r_{t0},\ f_{\tau,min}(X^s),\ f_{k,max}(X^s),\ f_{p_p,min}(X^s) \right). \tag{5.4}$$

In order to define the best design(s) satisfying the optimization conditions, the MAE introduced in (2.40) is computed between the predictions Y^s and the

fictive output Y^f such that the designs producing output characteristics similar to the desired ones assume a smaller error.

The three best valve seat designs identified with the optimization are depicted in Figure 5.12 with a parallel plot. In the axes are reported the non-fixed geometries and the investigated characteristics; the colors indicates the optimized designs. The geometries having less influence on the modeled characteristics such as the circumferential step height st_h, the circumferential angle β and the spray hole length sh_l have more freedom in the design space. In contrast, the inclination angle α and needle lift height nl_h are forced to assume specific values in order to satisfy the optimization constraints.

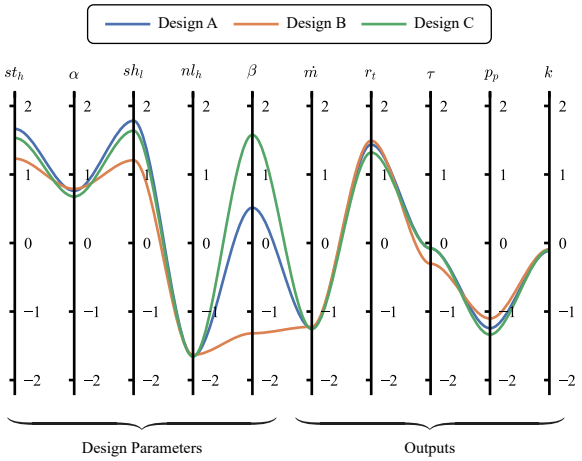


Figure 5.12: Machine learning optimized valve seat designs and their respective outputs.

5.2 Experimental Analysis of Spray

The application of the KD framework for the analysis of nozzle spray characteristics based on experiments is performed with the data generated during the GDI 6-hole injector investigation from [96, 115]. The dataset contains 1451 observations corresponding to the combination of two DOEs: one including 60 valve seat designs and the other one 33 engine operating points. Specifically, the total amount of the observations corresponds to the Cartesian product of the two DOEs satisfying specific constraints. The considered explanatory variables are eight valve seat geometries and four operating parameters, which are summarized as follows:

$$X = \left(\underbrace{\alpha\,,\ sh_l\,,\ sh_d\,,\ l/d\,,\ ph_d\,,\ \Psi\,,\ wt_l\,,}_{\text{Valve Seat Designs}}\ \underbrace{i_f\,,\ P_f\,,\ P_c\,,\ T_f\,,\ t_i}_{\text{Engine Operating Parameters}} \right). \quad (5.5)$$

Most of the geometries are reported in Figure 5.1, while the operating parameters are the fuel pressure P_f, the combustion chamber pressure P_c, the fuel temperature T_f and the injection time t_i. Additional parameters are the fuel flow rate i_f and the ratio l/d. The latter corresponds to the ratio of the spray hole length sh_l and the spray hole diameter sh_d. Experiments are constrained by stricter manufacturing limitation and test benches capabilities. Therefore a continuous design space is not always possible. Thus, the DOE plan is the result of domain knowledge and designs feasibility.

The measurements of the DOE are performed with high resolution spray cameras, producing several physical information. More details regarding the experimental environment can be found in [96]. For this analysis, three spray characteristics are selected: the spray angle γ, the spray width s_w and the fuel spray penetration p_s, depicted in Figure 5.13 and summarized as follows:

$$Y = \left(\gamma\,,\ s_w\,,\ p_s \right). \quad (5.6)$$

The spray width s_w is measured at 30 mm from the injector tip, while the spray angle γ at a distance d_γ from s_w. All the quantities are temporally averaged within the injection time. The analysis of these spray characteristics is relevant since they may influence the spray impingement as well as the air-fuel mixture formation, affecting fuel consumption and emissions. The geometries and

the output variables are anonymized with the Z-score normalization due to confidentiality data policy (see Section 2.2.3).

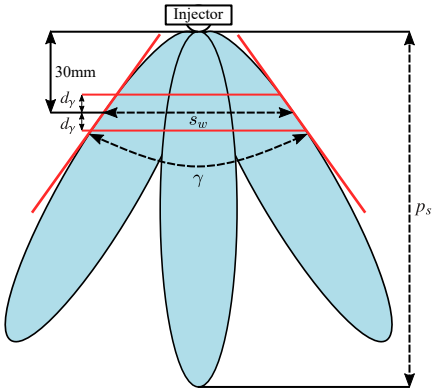


Figure 5.13: Measured nozzle spray characteristics, based on [116].

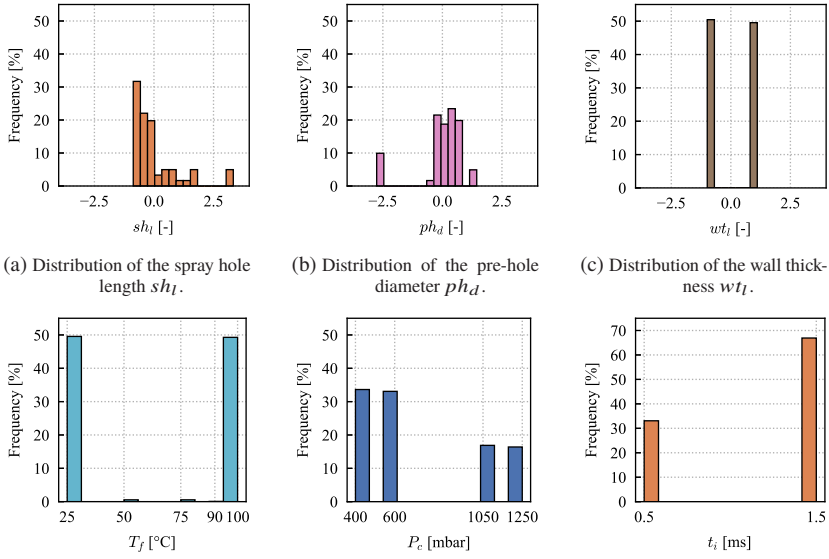
5.2.1 Data Exploration and Preprocessing

In this section, the exploration and the preprocessing steps of the data are reported. In addition, an analysis of the duplicated measurements is given in terms of dimension reduction.

Input Analysis

The most characterizing data distributions of the sampled nozzle geometries and the operating parameters are represented in Figure 5.14. The rest of the distributions are reported in Appendix A.2.1. As previously introduced, the experimental DOE is based on a discrete domain space due to manufacturing limitations. For instance, the wall thickness wt_l is only manufactured with only two different lengths, as depicted in Figure 5.14c. Similarly, the possible operating parameters are also limited by the test bench and the engine characteristics. Therefore, few variations are available for the fuel temperature T_f, the combustion chamber pressure P_c and the injection time t_i, as reported in Figure 5.14d, Figure 5.14e and Figure 5.14f respectively. The rest of the variables follow more uniform and normal distributions, with positive or negative skews,

according to the imposed constraints. In particular, the spray hole length sh_l in Figure 5.14a and the pre-hole diameter ph_d in Figure 5.14b contain a small cluster of observations far away from the rest of the data. Essentially, these clusters correspond to valve seat designs without the pre-hole, i.e. the spray hole length sh_l is equal to the wall thickness wt_l. The pre-hole diameter ph_d of these observations is set to a small arbitrary and fictive value as a placeholder. Another peculiarity present in the data are the few anomalies assumed by the fuel temperature T_f. Beside the two main cold ($25°C$) and warm ($100°C$) temperatures, the variable assumes also other three values measured only for few geometry variations. This is represented in Figure 5.15a, where the sampling between fuel temperature T_f and spray hole diameter sh_d is reported. Furthermore, due to the high experimental costs and physical limitations, only the most interesting and possible parameters combinations are measured. In particular, considering Figure 5.15b, only a longer injection time t_i is measured for large values of the combustion chamber pressure P_c, leading to an unbalanced



(a) Distribution of the spray hole length sh_l.

(b) Distribution of the pre-hole diameter ph_d.

(c) Distribution of the wall thickness wt_l.

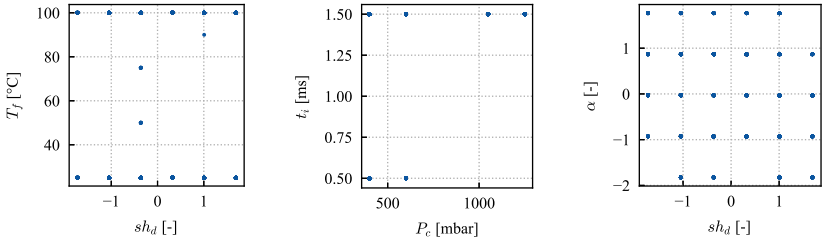
(d) Distribution of the fuel temperature T_f.

(e) Distribution of the combustion chamber pressure P_c.

(f) Distribution of the injection time t_i.

Figure 5.14: Most characterizing data distributions of the input variables.

sampling, as represented in Figure 5.14e and Figure 5.14f. The unconstrained sampling between the spray hole diameter sh_d and the inclination angle α is reported in Figure 5.15c.



(a) Temperature anomalies. (b) Constrained sampling. (c) Unconstrained sampling.

Figure 5.15: Examples of constrained and unconstrained samplings.

Considering the introduced constraints, it is necessary to investigate the presence of multicollinearity (see Section 2.2.4). In Figure 5.16, the correlation matrix is reported for the input variables. Together with the sampled geometries and operating parameters, the derived ratio l/d is included in the correlation matrix as well. As expected, the latter has a high positive correlation with the spray hole length sh_l and a high negative correlation with the spray hole diameter sh_d. Moreover, the presence of the designs without prehole generates a spurious correlation (see Section 2.2.4) between the spray hole length sh_l and the pre-hole diameter ph_d: this correlation is not related to any specific physical relation, but it is due to the fact that to the largest spray hole length sh_l is associated a constant pre-hole diameter ph_d. This relationship causes an indirect correlation between the pre-hole diameter ph_d and the ratio l/d as well. Another spurious correlation appears between the combustion chamber pressure P_c and the injection time t_i due the constrained sampling introduced with Figure 5.15b. Finally, the correlation between the spray hole diameter sh_d and the fuel flow rate i_f is an example of redundant information: the size of the spray hole diameter sh_d determines the fuel flow rate i_f.

The following preprocessing operations are performed to remove the multicollinearity and clean the dataset:

- the designs without prehole are neglected. The knowledge extraction may result biased by these few special cases;
- the fuel temperatures T_f between the two extreme values are neglected;

- the ratio l/d is neglected in favor of the spray hole length sh_l and the spray hole diameter sh_d;
- the fuel flow rate i_f is neglected in favor of the spray hole diameter sh_d.

Both the combustion chamber pressure P_c and the injection time t_i are kept due to the impossibility of removing one of them without major information loss.

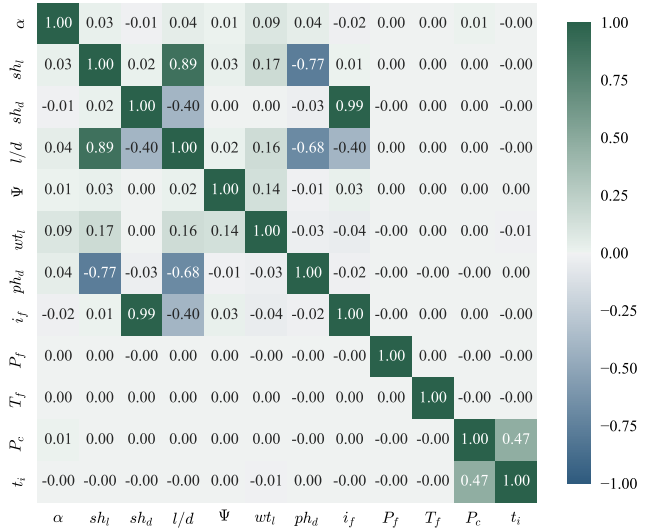


Figure 5.16: Correlation matrix of the explanatory variables.

The cleaned input dataset contains 1290 observations and 10 features, which are summarized as follows:

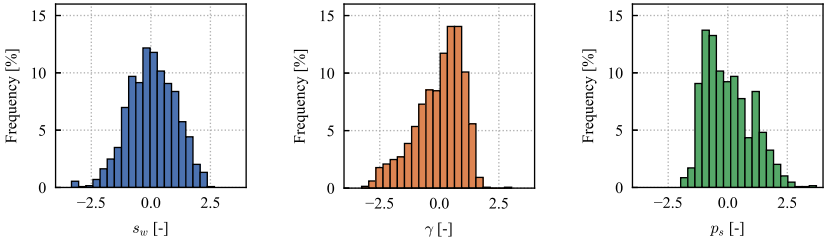
$$X^{prep} = \left(\alpha\,,\ sh_l\,,\ sh_d\,,\ ph_d\,,\ \Psi\,,\ wt_l\,,\ P_f\,,\ P_c\,,\ T_f\,,\ t_i \right). \tag{5.7}$$

A final remark regarding the considered sparse discrete input sampling has to be done. In case a variable sampled on few extreme points has a large influence on the investigated characteristics, the response variables would follow different behaviors according to the values assumed from that variable. This situation may cause a bias in the knowledge extraction. For instance, it is known from the domain expertise that the fuel temperature T_f and the combustion chamber

pressure P_c have a strong impact on spray shape. This effect is called *flash-boiling* [115, 117]. Therefore, it may be necessary to split the dataset on these variables and analyze the portions of data separately. A more detailed analysis of this aspect is reported in the next sections.

Output Analysis

Due to acquisition software errors or high uncertainty in the measurements, experimental datasets are more sensitive to missing data. In the considered dataset, only two values are missing for the spray angle, corresponding to the 0.16% of the available observations. Considering the small portion of missing values, the corresponding entries can be neglected for the analysis, which can be performed with the remaining 1288 datapoints. The data distributions of the measured spray characteristics are reported in Figure 5.17. All the characteristics tend to assume a normal distribution, without large skews. Since no further information indicate the presence of anomalies and that the observations are plausible, all the datapoints are kept for the modeling phase.



(a) Distribution of the spray width s_w.

(b) Distribution of the spray angle γ.

(c) Distribution of the fuel spray penetration p_s.

Figure 5.17: Distributions of the investigated experimental spray characteristics.

Sampling Limitation

After the preprocessing, the dataset is composed of 54 distinct valve seat designs, each one measured at least at 23 different operating points. Therefore, according to the relevant dimensions in the data, different observations may be associated to the same results. As instance, assuming that the operating points have much lower influence on the outputs with respect to the valve seat

geometries (or vice versa), for some observations the variation in the input space would not generate any variation in the output space. In this case, the validation of the modeling performance with a classical K-fold CV or a training-test split may be compromised by the leakage of information from the training to the test set.

In order to quantify how the grid combination of geometry designs with operating points may affect the model quality, a specific validation procedure based on the stratified sampling (see Instance Reduction in Section 2.2.4) is performed. This methodology is represented in Figure 5.18 and described with four steps as follows.

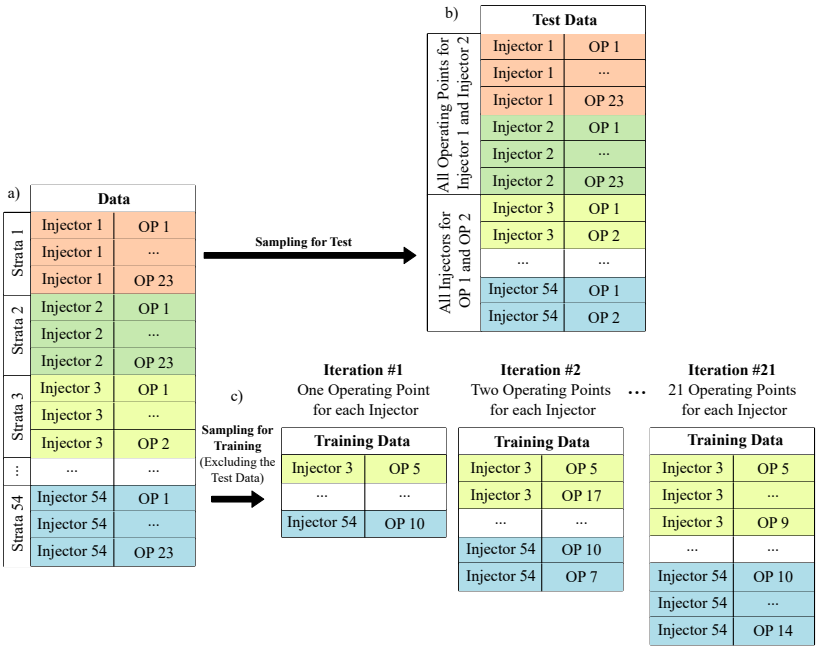


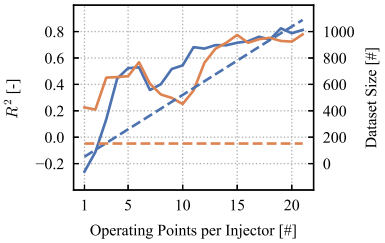
Figure 5.18: Example of stratified sampling applied to validate the models of the experimental spray characteristics. Totally, 54 injectors and 23 operating points are considered.

1. Stratification. The dataset is divided into several subsets, called strata. Each strata is composed by all the observations of a specific injector: considering the 54 available injectors, then 54 strata are defined, see a) in Figure 5.18.

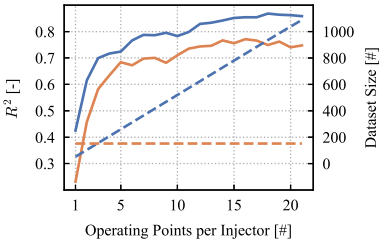
2. Test Set Definition. The test set is composed of all the observations related to two different injector designs and two different operating points. Therefore, the test set contains the measurements of all the available operating points for the two selected injectors, as well as all the available injectors for the two selected operating points, see b) in Figure 5.18. In this way, the remaining data used for the training would not contain any observation regarding the chosen inputs combination, reducing the risk of information leakage and ensuring a correct evaluation of the generalization capacity of the models.

3. Training Set Definition. The model validation is performed iteratively: at each iteration, the modeling performance is evaluated on a new training set. The latter is composed by the random sampling of i observations from each strata, excluding the measurements selected for the test set. This means, that the new training set contains i operating points for each injector. In this case, $i \in [1, 21]$, where 21 corresponds to the minimum number of measurements for each injector, excluding the test set. See c) in Figure 5.18

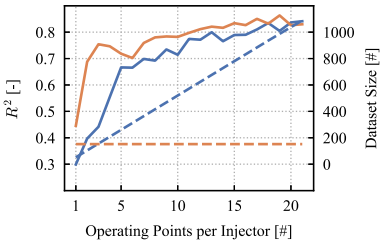
4. Learning Curves. In Figure 5.19, the results of the iterative stratified sampling for each modeled characteristics are depicted. This plots are referred as learning curves. On the x-axis are reported the number of operating points per injector considered at each iteration. The left y-axis indicates the modeling score in terms of R^2, while the right y-axis represents the size of the test and the training sets. The learning algorithm adopted is the XGBoost with its default hyperparameters, which are able to provide good results to evaluate the learning potential (see Section 4.3.3). The training set is evaluated with a 5-fold CV. For all the analyzed outputs, the training and the test scores globally increase along the iterations, i.e. with a larger number of observations for the training set. Only the test score of the spray angle γ stops increasing after about 15 operating points per injector. After this point, it does not drop significantly and remains stable. Therefore, all the learning curves do not show any overfitting situation. The models are able to generalize well on unknown datapoints and all the measured observations are used to build the models.



(a) Learning curve of spray width s_w.



(b) Learning curve of spray angle γ.



(c) Learning curve of fuel spray penetration p_s.

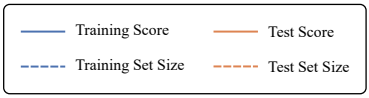


Figure 5.19: Learning curves based on the number of operating points per injector.

5.2.2 Model Selection and Validation

The preprocessed dataset containing 1288 datapoints and 10 features is used for the modeling phase. The XGB-NSGAII and the polynomial regression are applied with the same settings introduced in Section 5.1.2.

In Table 5.2, the scores resulting from the modeling methods are reported in terms of μ_{R^2}, μ_{MAE}, σ_{MAE} and e_{MAE}. The XGBoost outperforms the polynomials in any dataset. This indicates the presence of high non-linearities in the data, which cannot be properly modeled by the latter. The XGBoost performance improves in general between 4% and 22% for the μ_{R^2} and between 34% and 38% for the μ_{MAE} with respect to the polynomial. The results obtained through the XGBoost ensure that the models captured the physics behind the data. The lowest μ_{R^2} is 0.876 achieved by the spray width s_w; right after it, there are the fuel spray penetration p_s with 0.884 and the spray angle γ with 0.947. In addition, the models are able to provide a good generalization on unknown points.

Table 5.2: Modeling scores of the experimental spray analysis.

	XGBoost				Polynomial				
Data	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	degree
γ	**0.947**	**0.140**	0.009	0.133	0.910	0.212	0.012	0.189	3
s_w	**0.876**	**0.242**	0.014	0.237	0.719	0.393	0.012	0.379	3
p_s	**0.884**	**0.207**	0.017	0.209	0.802	0.320	0.017	0.312	3

In Figure 5.20, the μ_{MAE}, the σ_{MAE} and the e_{MAE} are depicted for each model. The test error is always in the interval $[\mu_{MAE} \pm 2\sigma_{MAE}]$, which ensures the reliability of the models and excludes overfitting.

In order to investigate the effect of the sparse discrete sampling on the modeling performance, the XGB-NSGAII is performed separately on the observations measured at cold and at warm fuel temperatures T_f. In Figure 5.20, the scores resulting from the K-fold CV on the split datasets are reported as μ_{MAE} *Warm* and μ_{MAE} *Cold*. The fuel temperature T_f does not have a large impact on the spray width s_w performance: the accuracy computed on the whole dataset and on the temperature splits are similar. In contrast, the models of the spray angle γ and the fuel spray penetration p_s achieve different accuracy levels according to the fuel temperature T_f. In particular, the behavior of the investigated characteristics is better captured for the cold temperature. Nevertheless, the performance on the complete dataset corresponds approximately to the average of the accuracy achieved on the single temperature splits. The tree-based structure of the XGBoost is able to isolate the behavior of the spray

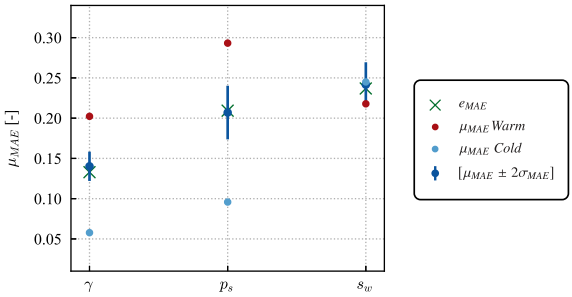


Figure 5.20: Evaluation of the XGBoost modeling quality in terms of the scores μ_{MAE}, σ_{MAE} and e_{MAE} from the experimental spray analysis.

characteristics at different temperatures from the complete dataset and model them as they were analyzed separately. For this reason, the accuracy of the global models corresponds to the average of the split ones. It is not necessary to split the data for the modeling phase.

5.2.3 Knowledge Extraction

In this section, the knowledge extraction for the analysis of the nozzle spray characteristics based on experiments is performed in terms of models exploration and models exploitation.

Model Exploration

The XGBoost models computed on the investigated spray characteristics are explored through feature importance and partial dependencies (see Section 2.4). In Figure 5.21, the extracted relative feature importance are reported. On the x-axis the valve seat geometries and the operating parameters are listed, while on the y-axis their relative feature importance on each modeled output is reported. As expected, one of the dominating parameters is the fuel temperature T_f due to the flash-boiling conditions. The second most influencing variable is the inclination angle α, while the other parameters apparently do not show a large effect on the investigated outputs. Differently from the modeling accuracy, where the learning algorithm is able to generalize the influence of the sparse fuel temperature T_f, the extracted knowledge may result biased: the

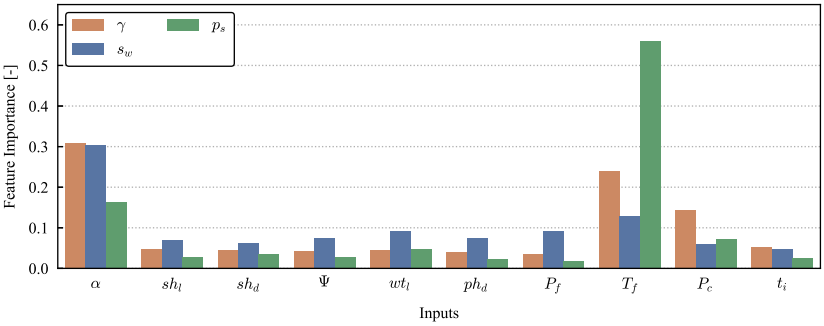


Figure 5.21: Feature importance of the experimental spray analysis.

real effects of the other variables may be masked by the strong influence of the fuel temperature T_f. For instance, it is known that the combustion chamber pressure P_c contributes to the flash-boiling as well, but its influence on the spray reported in Figure 5.21 is low. Therefore, it is necessary to investigate the effects of the fuel temperature T_f and of the other input parameters separately.

The partial dependence of the fuel temperature T_f on the three investigated spray characteristics is depicted in Figure 5.22. The represented dependencies between the two measured temperatures are interpolated. On the x-axis is reported the temperature variation and on the y-axis its effect on the investigated characteristics. Due to the flash-boiling, the fuel temperature T_f has a strong effect on the spray angle γ and on the fuel spray penetration p_s, but almost negligible on the spray width s_w. To be specific, warm fuel temperatures generally lead to higher fuel spray penetration p_s and lower spray angle γ.

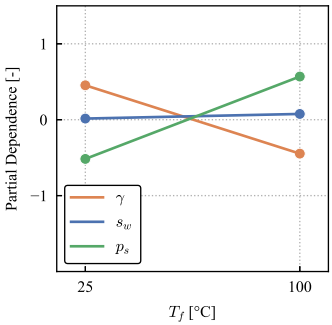
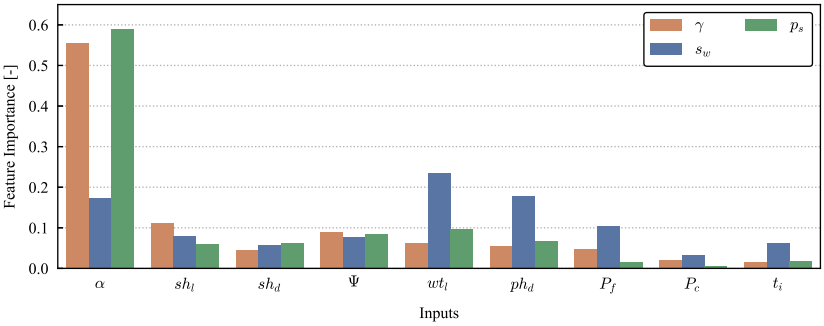
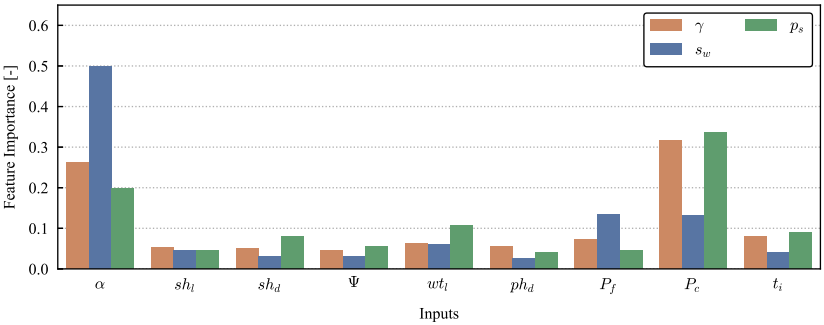


Figure 5.22: Partial dependence of the fuel temperature T_f on the spray characteristics.

The feature importance of the models computed on the portions of cold and warm observations are reported in Figure 5.23. Similarly to the modeling accuracy, the feature importance extracted from the complete dataset in Figure 5.21 corresponds to the averaged importance computed on the split datasets. For cold temperatures, the inclination angle α dominates the spray angle γ and fuel spray penetration p_s, while its effect on the spray width s_w is shared with other geometries like the wall thickness wt_l and the pre-hole diameter ph_d. In contrast, for warm temperatures, the inclination angle α dominates the spray width s_w and it has a much lower effect on the other two outputs. In this case, the operating parameters have a larger effect on the spray characteristics with respect to the geometries, especially the combustion chamber pressure P_c.



(a) Feature importance for cold ($25°C$) fuel temperature T_f.

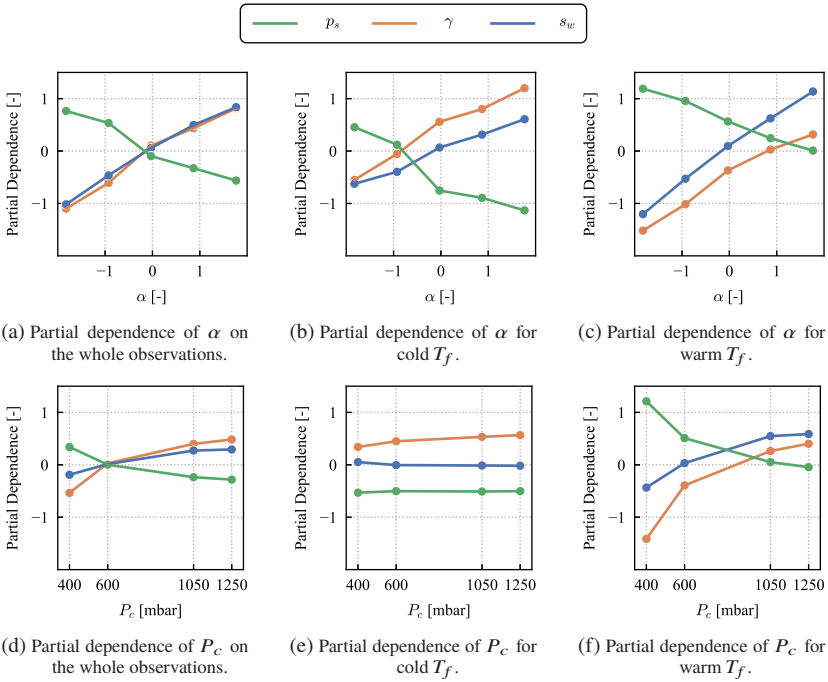


(b) Feature importance for warm ($100°C$) fuel temperature T_f.

Figure 5.23: Experimental spray analysis feature importance based on the fuel temperature T_f.

The partial dependencies of the two most characterizing variables, i.e. the inclination angle α and the combustion chamber pressure P_c, are reported in Figure 5.24 for the complete dataset as well as for cold and warm fuel temperatures. The remaining partial dependencies are reported in Appendix A.2.2. The partial dependencies extracted from the complete dataset correspond to the averaged partial dependencies of the two portion of data split on the temperatures. Although the general behavior represented by the complete dataset is correct, the possibility to extend the investigation to smaller domain spaces and analyze the underlying influences of the parameters is essential. The partial dependencies of the main influencing parameters are deeper analyzed for all the temperature conditions as follows.

Inclination angle α. The influence of inclination angle α on the spray characteristics is reported in Figure 5.24a-c. The inclination angle α has a positive influence on the spray angle γ and the spray width s_w. Considering the definition of α in Figure 5.1 and of γ in Figure 5.13, then $\gamma \approx 2\alpha$. Therefore, γ has a positive correlation with α, which in turns is valid for s_w as well. In contrast, the fuel spray penetration p_s is negatively influenced by α. As already described for the nozzle numerical analysis in Section 5.1.3, a larger spray angle produces a broader but a shorter penetration. The effects of α are amplified in case of warm fuel temperatures.



(a) Partial dependence of α on the whole observations.

(b) Partial dependence of α for cold T_f.

(c) Partial dependence of α for warm T_f.

(d) Partial dependence of P_c on the whole observations.

(e) Partial dependence of P_c for cold T_f.

(f) Partial dependence of P_c for warm T_f.

Figure 5.24: Partial dependencies of inclination angle α and combustion chamber pressure P_c for the investigated spray characteristics based on the whole observations as well as the warm ($100°C$) and cold ($25°C$) fuel temperature T_f.

Combustion chamber pressure P_c. The influence of the combustion chamber pressure P_c on the spray characteristics is reported in Figure 5.24d-f. For cold fuel temperatures T_f, the P_c has almost no influence on the spray characteristics. In contrast, its effect changes dramatically for warm fuel temperatures. In this case, a low P_c favors the flash-boiling effect, producing a large fuel spray penetration p_s together with a small spray angle γ. Increasing P_c, the flash-boiling effect reduces as well as its effect on the spray characteristics. The influence of the P_c on the spray width s_w is less accentuate compared to the other two characteristics. The effect of P_c computed on the completed dataset results overestimated for cold fuel temperatures T_f and underestimated for warm fuel temperatures T_f.

Model Exploitation

Generally, it is necessary to design a robust injector against the flash-boiling conditions. For instance, given a warm fuel temperature T_f and a low combustion chamber pressure P_c, the injector design should be able to generate a low fuel spray penetration p_s. Furthermore, consider the fuel pressure P_f as well as the injection time t_i to be constant due to engine specifications. The valve seat can be optimized through the previously trained XGBoost model $f_{p_s}(\cdot)$ in order to minimize the fuel spray penetration p_s. The optimization problem is summarized as follows:

$$\begin{aligned} \text{Minimize} \quad & f_{p_s}(X) \\ \text{with} \quad & X = \left(\alpha,\ sh_l,\ sh_d,\ ph_d,\ \Psi,\ wt_l,\ P_f,\ P_c,\ T_f,\ t_i\right) \\ \text{subject to} \quad & \text{fixed } \left(T_f,\ P_c,\ P_f,\ t_i\right), \\ & X_i^{(l)} \leq X_i \leq X_i^{(u)},\ X_i \in \left[\alpha,\ sh_l,\ sh_d,\ ph_d,\ \Psi,\ wt_l\right]. \end{aligned} \tag{5.8}$$

Similarly to the model exploitation performed in the nozzle numerical analysis in Section 5.1.3, a new larger DOE of ten thousand designs is generated using the Sobol sampling method, according to the constraints defined in (5.8). The three best designs are depicted in Figure 5.25 as parallel plot together with their respective outputs. The optimized designs are all able to produce a low fuel spray penetration p_s with different spray angle γ and spray width s_w. The inclination angle α for these designs tends to assume large values. As already discussed and shown in Figure 5.24c, large values of α provide a lower p_s. The rest of the geometry parameters have low influences on p_s for warm fuel

temperature T_f, as reported in Figure 5.23b. Therefore, those parameters have generally more freedom in the design space. The spray hole conicity Ψ and wall thickness wt_l are fixed for all the optimized designs. The reason of this result may be associated to the low density of the parameter space of these two parameters.

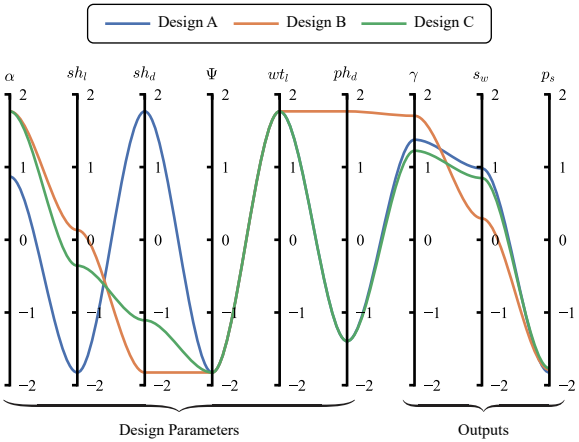


Figure 5.25: Machine learning optimized designs and their respective outputs.

5.3 Summary

In this chapter, the potential of the data-driven development based on the novel KD framework has been presented on the analysis of the high pressure injector. First, the results of 3D-CFD simulations are adopted to investigate the influence of injector nozzle geometries on fuel massflow, spray plume angle, TKE, spray targeting radius and fuel plume penetration. Second, spray chamber experiments of multi-hole injectors are considered to analyze the effect of nozzle geometries and operating parameters on spray angle, spray width and fuel spray penetration.

After a general introduction of the datasets, the data exploration and pre-processing are performed. Input and outputs variables are analyzed in order to handle redundant information, spurious correlations and missing values. The

presence of outliers is investigated as well. The risk of overfitting caused by the sampling strategy is examined with an iterative stratified model validation for the experimental analysis. The preprocessed data are used to build robust and reliable models through the XGB-NSGAII. The modeled inflow and spray characteristics are then explored and exploited with the guidance of feature importance and partial dependencies plots. These information revealed a proper representation of the physics behind the data, which is further validated with the knowledge domain. The presence of sparse discrete samplings of relevant features demanded the analysis of the physical phenomena on portions of the dataset. Finally, the machine learning models are applied to select optimal injector designs and operating points able to accomplish predefined constraints given by manufacturing limitation and engine specifications.

6 Engine Analysis

In this chapter, the KD framework is applied to analyze the effects of nozzle geometries, spray targeting and injection strategies on engine mixture formation, emissions and fuel consumption. The analysis is performed through numerical investigations and experiments.

In Figure 6.1, a section view of the cylinder of a Bosch internal turbo-charged GDI two-cylinder research engine with central mounted injector is depicted. The figure corresponds to a sketch of the GDI system represented in Figure 1.1 and introduced in Section 1.1. The spray plumes in Figure 6.1 are simplified with cones.

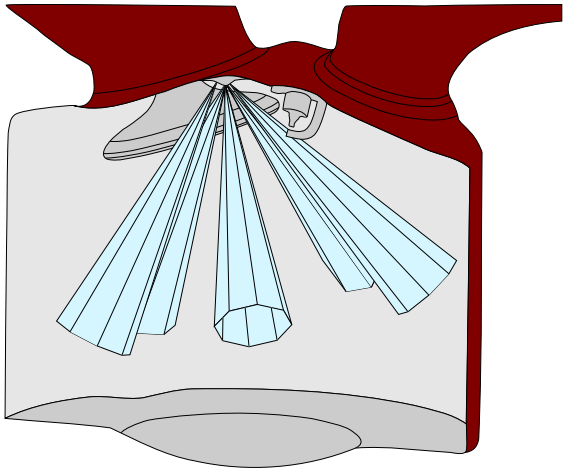


Figure 6.1: Section view of the cylinder of a Bosch internal turbo-charged GDI engine.

6.1 Numerical Analysis of Mixture Formation

In order to investigate how the interaction between spray targeting and injection strategies affects the engine mixture formation, the KD framework is applied on 3D-CFD engine results. The considered dataset represents a DOE of 500 spray targeting and injection strategy variations sampled using the Sobol approach (see Section 3.2.1). The spray targeting is characterized by the variation of the spray directions of a 5-hole injector in terms of plumes targeting coordinates $(x_t, y_t)_1, \ldots, (x_t, y_t)_5$ on a fixed Z-plane. The injection strategy is described through two start of injections, indicated as SOI_1 and SOI_2, as well as through the percentage of injected fuel mass at each injection, defined as m_{split1} and m_{split2}. The spray targeting coordinates as well as the output variables are anonymized with the Z-score normalization (see Section 2.2.3) due to confidentiality data policy. The input design space is summarized as follows:

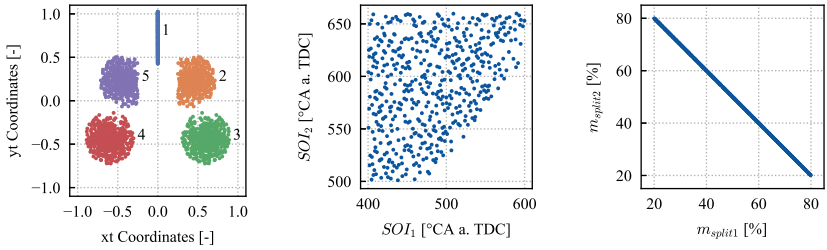
$$X = \Big((x_t, y_t)_1 \,,\; \ldots \,,\; (x_t, y_t)_5 \,,\; SOI_1 \,,\; SOI_2 \,,\; m_{split1} \,,\; m_{split2} \Big). \tag{6.1}$$

The remaining engine and injector geometries as well as the engine operating points are kept constant. In order to ensure the feasibility of the observations, specific constraints are imposed during the DOE sampling according to domain expertise. These are listed in Definition 6.1.1 and depicted in Figure 6.2.

Definition 6.1.1. DOE constraints applied for the engine numerical analysis:

- C_1: minimum distance between the spray plumes, see Figure 6.2a;
- C_2: the x_t-coordinate of the spray plume one is constant, see Figure 6.2a;
- C_3: the spray plumes are symmetric about the first plume, see Figure 6.2a;
- C_4: the two injections are in sequence and separated by a minimum pause interval $\rightarrow SOI_2 \geq SOI_1 + c_4$, see Figure 6.2b;
- C_5: the total amount of fuel injected is fixed $\rightarrow m_{split1} + m_{split2} = 100\%$, see Figure 6.2c.

The 500 designs are evaluated with a full automated 3D-CFD workflow specialized for in-cylinder dynamic simulations of mixture formation, which required ~240.000 $CPU * h$ on a HPC. A single simulation produces a large amount of spray and mixture information. In this work, four physical quantities are investigated: the impinged spray mass m_s, the liner impinged mass m_l, the piston



(a) Constraints C_1, C_2, C_3 on the five spray plumes.

(b) Constraint C_4 on the start of injections.

(c) Constraint C_5 on the mass splits.

Figure 6.2: Constrained samplings, following Definition 6.1.1.

impinged mass m_p and the homogeneity index λ_{90}, which are summarized as follows:

$$Y = \left(\; m_s \, , \; m_l \, , \; m_p \, , \; \lambda_{90} \; \right) . \tag{6.2}$$

The liner impinged mass m_l and piston impinged mass m_p correspond to the total amount of fuel injected on the liner and the piston respectively. The impinged spray mass m_s consists of the sum of all the impinged droplets in the cylinder. The impinged fuel results into delayed mixture formation, which may cause higher emissions and fuel consumption. Furthermore, the mixture formation is characterized by the homogeneity index λ_{90} as well, which is computed from the distribution of the air-fuel ratio λ in the combustion chamber. As reported in Figure 6.3, the homogeneity index λ_{90} corresponds to the length of the interval containing 90% of the air-fuel ratio values. For homogeneous GDI engines, the air-fuel ratio λ should be uniform in the whole combustion chamber. Thus, a small homogeneity index λ_{90} is aimed.

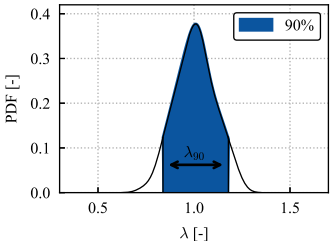


Figure 6.3: Air-fuel ratio λ distribution and evaluation of the homogeneity index λ_{90}.

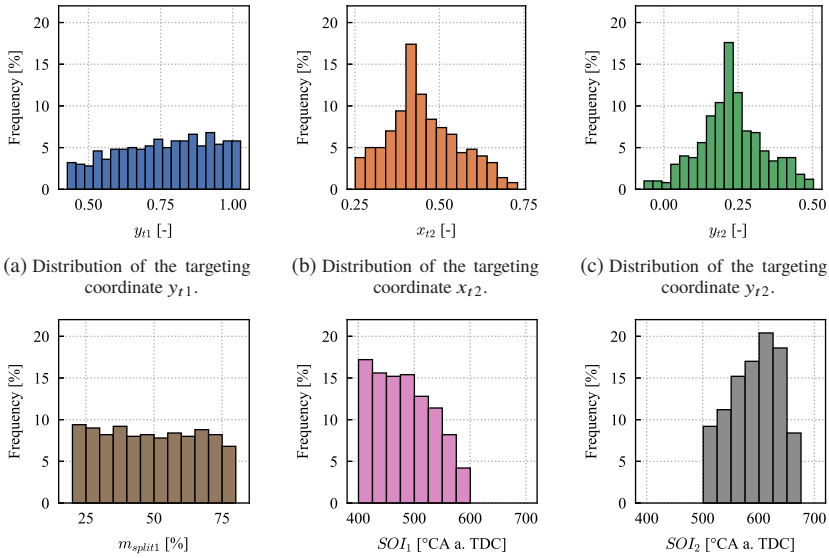
The investigated outputs correspond to the spatially averaged quantities along the considered surfaces at the TDC.

6.1.1 Data Exploration and Preprocessing

In this section, the exploration and the preprocessing steps of the data are reported.

Input Analysis

The most characterizing data distributions of the sampled spray targeting and injection strategies are represented in Figure 6.4. The remaining distributions are depicted in Appendix A.3.1. Similarly to the nozzle numerical analysis in Section 5.1, the data distributions sampled with the Sobol approach are uniform and highly dense. However, considering the constraints reported



(a) Distribution of the targeting coordinate y_{t1}.

(b) Distribution of the targeting coordinate x_{t2}.

(c) Distribution of the targeting coordinate y_{t2}.

(d) Distribution of the first injection mass split m_{split1}.

(e) Distribution of the first start of injection SOI_1.

(f) Distribution of the second start of injection SOI_2.

Figure 6.4: Most characterizing data distributions of the input variables.

in Definition 6.1.1, distribution skews are present. The distributions of the two start of injections SOI_1 and SOI_2 present a positive and negative skew respectively, ensuring the order of injections and a minimal interval between them. The spray targeting coordinatess (x_t, y_t) are uniformly sampled as polar coordinates, which then results into normal distributions in the Cartesian system. The constraints applied during the sampling have further influences on the distributions. For instance, the y-coordinate of the first spray plume y_{t1} tends to be uniform with a minimal negative skew. This is due to the absence of variation on the x-coordinate of that spray plume and it ensures a minimum distance from the fifth and second plumes, as depicted in Figure 6.2a.

Given the constraints in Definition 6.1.1, it is necessary to investigate the presence of multicollinearity (see Section 2.2.4). In Figure 6.5, the correlation matrix for the input variables is reported. The x-coordinate of the first plume x_{t1} is neglected because it is constant. Three main correlation clusters can be observed in the matrix: one for the targeting coordinates (x_t, y_t), one for the mass splits m_{split1} and m_{split2} and one for the start of injections SOI_1 and SOI_2. The constraint C_1 causes the small correlations between the various targeting coordinates, while the symmetry imposed by the constraint C_3 is responsible for the significant correlations between the couple of coordinates (x_{t2}, x_{t5}), (y_{t2}, y_{t5}), (x_{t3}, x_{t4}), (x_{t3}, y_{t3}). The two start of injections SOI_1 and SOI_2 are slightly correlated due to the constraint C_4. Finally, the constraint C_5 induces the correlation between the mass splits m_{split1} and m_{split2}: from one mass split it is possible to obtain the other one, i.e. they produce redundant information (see also Figure 6.2c).

The following preprocessing operations are performed to clean the data:

- for each couple of symmetric coordinates, one of them is neglected. In this way, the representation of the input space remains unchanged;
- the second injection mass split m_{split2} is neglected in favor of the first one; a new variable referred as fuel mass split m_{split} indicates the m_{split1}.

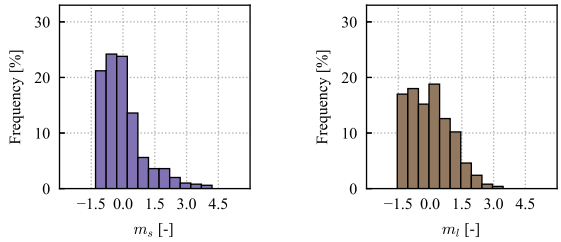
Both start of injections SOI_1 and SOI_2 are kept due to the impossibility of removing one of them without information loss. The preprocessed input space considered for the modeling phase is summarized as follows:

$$X^{prep} = \left(\; y_{t1} \,,\; (x_t, y_t)_2 \,,\; (x_t, y_t)_3 \,,\; SOI_1 \,,\; SOI_2 \,,\; m_{split} \; \right). \tag{6.3}$$

Figure 6.5: Correlation matrix of the explanatory variables.

Output Analysis

The distribution of the considered engine characteristics resulting from 3D-CFD simulations are reported in Figure 6.6. All the distributions tend to assume a positive skew, especially the piston impinged mass m_p. The datapoints far away from the average of the observations are not completely isolated to consider them anomalies. Moreover, the goal of the analysis corresponds to identify the targeting and injection strategy achieving lower impingement and homogeneity index λ_{90}. Therefore, considering the observations with high impingement and large homogeneity index λ_{90} as possible anomalies and neglecting them would bias the models, since they would not be trained on the portion of results to avoid.



(a) Distribution of the impinged spray mass m_s.

(b) Distribution of the liner impinged mass m_l.

(c) Distribution of the piston impinged mass m_p.

(d) Distribution of the homogeneity index λ_{90}.

Figure 6.6: Distribution of the investigated 3D-CFD engine characteristics.

6.1.2 Model Selection and Validation

The preprocessed dataset containing 500 observations and 6 features can be used for the modeling step. The XGB-NSGAII and the polynomial regression are applied with the same settings introduced in Section 5.1.2.

In Table 6.1, the scores resulting from the modeling methods are reported in terms of μ_{R^2}, μ_{MAE}, σ_{MAE} and e_{MAE}. The XGBoost outperforms the poly-

Table 6.1: Modeling scores of the 3D-CFD engine analysis.

	XGBoost				Polynomial				
Data	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	**degree**
m_s	**0.926**	**0.189**	0.019	0.175	0.600	0.461	0.061	0.393	3
m_l	**0.855**	**0.275**	0.040	0.268	0.714	0.420	0.047	0.413	2
m_p	**0.974**	**0.087**	0.011	0.093	0.652	0.395	0.060	0.325	3
λ_{90}	**0.733**	**0.386**	0.017	0.353	0.696	0.403	0.020	0.374	2

nomials in any dataset. In particular, its performance on the impingement variables improves in general between 20% and 30% for the μ_{R^2} and between 35% and 78% for the μ_{MAE} with respect to the polynomial. The improvement on the homogeneity index λ_{90} is less substantial. The results obtained through the XGBoost ensure that the models captured the physics behind the data. The lowest μ_{R^2} is 0.733 achieved by the homogeneity index λ_{90}; right after it, there is the liner impinged mass m_l with 0.855, while the rest of the variables are modeled with scores higher than 0.9. In addition, the models are able to provide a good generalization on unknown points. In Figure 6.7, the μ_{MAE}, the σ_{MAE} and the e_{MAE} are depicted. The test error is always in the interval $[\mu_{MAE} \pm 2\sigma_{MAE}]$, which ensures the reliability of the models and excludes overfitting.

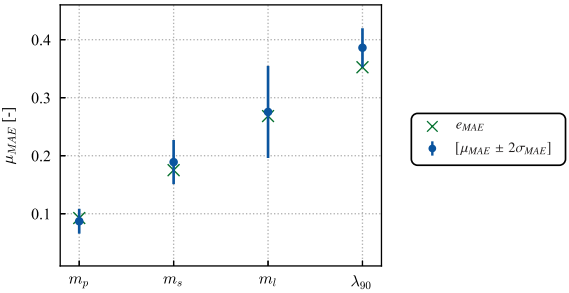


Figure 6.7: Evaluation of the XGBoost modeling quality in terms of the scores μ_{MAE}, σ_{MAE} and e_{MAE} from the 3D-CFD engine analysis.

6.1.3 Knowledge Extraction

In this section, the knowledge extraction for the analysis of how the interaction between spray targeting and injection strategies affects the engine mixture formation is performed in terms of models exploration and models exploitation.

Model Exploration

Once the robust and precise XGBoost models are available, the model exploration based on feature importance and partial dependencies can be performed

(see Section 2.4). In Figure 6.8, the relative feature importance retrieved during the modeling of the engine characteristics are reported. On the x-axis the investigated spray targeting coordinates and injection strategies are listed, while on the y-axis their relative feature importance on each modeled output is reported. The parameters dominating the considered output variables are the two start of injections SOI_1 and SOI_2 as well as the fuel mass split m_{split}. The piston impinged mass m_p together with the impinged spray mass m_s are mainly characterized by the first start of injection SOI_1, while the homogeneity index λ_{90} and the liner impinged mass m_l by the second start of injection SOI_2. The fuel mass split m_{split} does not dominate any characteristics, yet it has a larger impact than the targeting coordinates. Based on the discussed feature importance, it is possible to focus the investigation of the engine characteristics in a lower design space, including only the input parameters having a larger impact. Furthermore, it can be stated that the injection strategy has a larger impact on the investigated variables compared to the spray targeting.

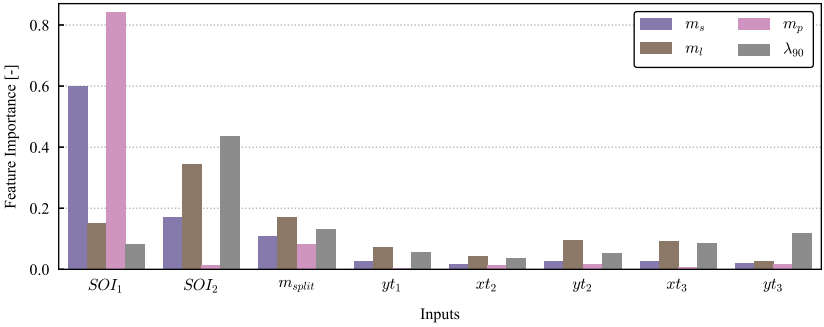
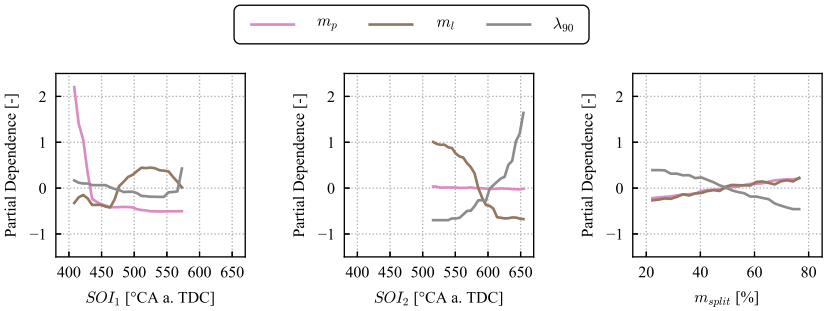


Figure 6.8: Feature importance of the 3D-CFD engine analysis.

The single interactions of SOI_1, SOI_2 and m_{split} are extracted from the models and depicted as partial dependencies in Figure 6.9. The complete partial dependencies are depicted in Appendix A.3.2. The influence of the parameters on the impinged spray mass m_s is omitted because it corresponds to the sum of the other impingements. On the x-axes are reported the injection strategies variations and on the y-axes their effect on the investigated characteristics. The partial dependencies are explained and validated through the domain knowledge as follows.



(a) Partial dependencies of SOI_1.

(b) Partial dependencies of SOI_2.

(c) Partial dependencies of m_{split}.

Figure 6.9: Partial dependencies of first start of injection SOI_1, second start of injection SOI_2 and fuel mass split m_{split} for the engine characteristics.

First start of injection SOI_1. A very early SOI_1 produces a large piston impinged mass m_p: until 450°*CA a. TDC* the piston is in the first half of its movement towards the BDC and it can be reached by the spray plumes. In this phase, the liner impinged mass m_l is not influenced by the SOI_1, since the cylinder liner is covered by the piston. As the latter moves away from the TDC, the m_p decreases until the first injection cannot reach the piston surface anymore. In contrast, during the piston descent, the m_l starts to increase until the area of the impinged liner remains constant. At 540°*CA a. TDC*, when the compression cycle starts, the spray plumes contract themselves reducing their penetration. Thus, the m_l decreases and the m_p stays unaltered. The SOI_1 has a low effect on the homogeneity index λ_{90} until 550°*CA a. TDC*: the fuel has enough time to be properly mixed with the air until the end of the cycle. The explanation of the large λ_{90} for a later SOI_1 is reported with the description of the SOI_2 influence.

Second start of injection SOI_2. The SOI_2 does not have a significant effect on the piston impinged mass m_p: in the interval where the second injection may start, the piston is either far away from the TDC or the compression phase reduces the spray penetration. The compression is responsible also for the monotonic behavior of the liner impinged mass m_l with the variation of the SOI_2: as the piston moves towards the TDC, m_l decreases together with the spray penetration. The homogeneity index λ_{90} is more sensitive to the SOI_2:

a later second start of injection corresponds to a shorter time for the fuel to be mixed with the air. Therefore, a good homogenization cannot be reached. In addition, the SOI_2 is responsible for the large λ_{90} in case of a late SOI_1: if the first injection happens after 550°*CA a. TDC*, the second injection has to start after the former, producing a larger λ_{90}.

Fuel mass split m_{split}. The influence of the m_{split} on the impingements is an indirect effect of the start of injections: injecting a larger mass during the second injection, i.e. a small m_{split}, the probability of impinging either the piston or the liner is low because of the reduced penetration. In contrast, the homogeneity index λ_{90} has in this case a negative relation with the m_{split}: a larger mass injected during the second injection is not able to generate a proper air-fuel mixture in the short interval before the end of the cycle.

Model Exploitation

Besides enabling the understanding of the investigated physical phenomena, the partial dependencies in Figure 6.9 can be analyzed to define a proper injection strategy. For instance, in order to achieve a low impingement, it is preferred to choose the first start of injection SOI_1 approximately between 450°*CA a. TDC* and 475°*CA a. TDC*. The selection of the second start of injection SOI_2 and the fuel mass split m_{split} may be more complicated. A low impingement is achieved with small values of both second start of injection SOI_2 and fuel mass split m_{split}. However, a low homogeneity index λ_{90} is given by large values of the latter variables. Due to the described conflict, the design definition becomes a multi-objective optimization problem with contradictive objectives.

The high accuracy models can be exploited in order to define the optimal injection strategy given a specific spray targeting. The optimization aims to minimize the impinged spray mass m_s corresponding to the sum of the whole impingements in the combustion chamber together with the homogeneity index λ_{90}. The optimization problem is summarized as follows:

$$\begin{aligned} \text{Minimize} \quad & f_{m_s}(X),\ f_{\lambda_{90}}(X) \\ \text{with} \quad & X = \Big(y_{t1},\ (x_t, y_t)_2,\ (x_t, y_t)_3,\ SOI_1,\ SOI_2,\ m_{split}\Big) \\ \text{subject to} \quad & \text{fixed}\ \Big(y_{t1},\ (x_t, y_t)_2,\ (x_t, y_t)_3\Big), \\ & X_i^{(l)} \le X_i \le X_i^{(u)},\ X_i \in \big[SOI_1, SOI_2, m_{split}\big]. \end{aligned} \tag{6.4}$$

Similarly to the model exploitation performed in the nozzle numerical analysis in Section 5.1.3, a new larger DOE of ten thousand designs is generated using the Sobol sampling method, according to the constraints defined in (6.4). The three best designs are depicted in Figure 6.10 as parallel plot together with their respective outputs. The resulting optimized designs are conform to the information extracted from the partial dependencies plots in Figure 6.9. The first start of injection SOI_1 lays between 450° and 475°*CA a. TDC*, where the impingement is minimum. The second start of injection SOI_2 having a contradictive effect on liner impinged mass m_l and homogeneity index λ_{90} can vary in the interval $[550°, 600°]$, where both objectives are minimized together, as shown in Figure 6.9b. Finally, although fuel mass split m_{split} has a contradictive effect on the impingement and homogeneity index λ_{90}, it has a stronger influence on the latter (see Figure 6.9c). Therefore, a larger fuel mass split m_{split} is selected for the optimized design.

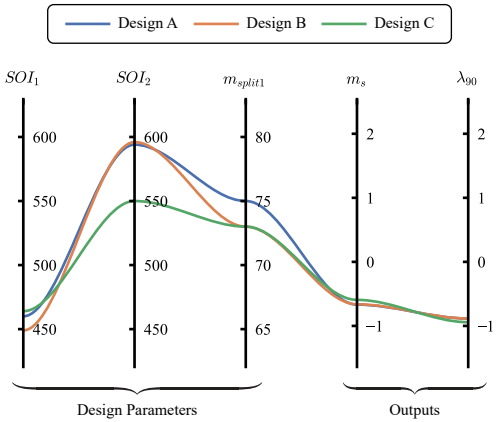


Figure 6.10: Machine learning optimized designs and their respective outputs.

6.2 Experimental Analysis of Emissions

The investigation of the nozzle experimental analysis reported in Section 5.2 has been extended to study the influence of valve seat geometries and operating points on engine emissions [96, 115]. In this case, the DOE of the geometries is reduced from 60 to 55 variations, while a new, second DOE containing 192 engine operating points is defined. The two DOEs are combined together as in the nozzle experimental analysis with a Cartesian product. Besides the eight geometrical valve seat parameters presented in Section 5.2, seven engine operating parameters are considered for this analysis. The investigated explanatory variables are summarized as follows:

$$X = \Big(\alpha\,,\ sh_l\,,\ sh_d\,,\ l/d\,,\ ph_d\,,\ \Psi\,,\ wt_l\,,\ i_f\,, IMEP\,,\ P_f\,, \\ T_w\,,\ l_f\,,\ SOI\,,\ n \Big) \tag{6.5}$$

The operating parameters are the indicated mean effective pressure $IMEP$, the fuel pressure P_f, the cooling water temperature T_w, the engine throttle valve position l_f, the start of injection SOI and the engine speed n. Similarly to the nozzle experimental analysis, the engine operating points are limited by the high measurement costs and the test bench capabilities. Therefore, the definition of the DOE plan represents a combination of domain knowledge and feasibility.

Out of the several physical information that can be measured, three emission quantities are selected for this analysis: the hydrocarbons HC, the nitrogen oxides NO_x and the particulate number PN, which are summarized as follows:

$$Y = \Big(\ HC\,,\ NO_x\,,\ PN\ \Big) \tag{6.6}$$

The geometries and the output variables are anonymized with the Z-score normalization (see Section 2.2.3) due to confidentiality data policy. More information regarding the experimental environment can be found in [96].

6.2.1 Data Exploration and Preprocessing

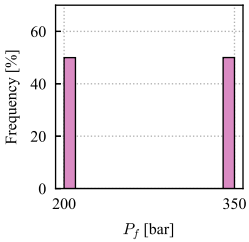
In this section, the exploration and the preprocessing steps of the data are reported. In addition, an analysis of the duplicated measurements is given in terms of dimension reduction.

Data Selection

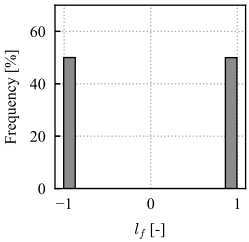
The dataset available for this investigation contains 27 817 points, which do not only corresponds to the observations defined in the DOE, but also to calibration and reference operating points as well as duplicated measurements. To ensure valid results, engine investigations require the evaluation of calibration and reference points as well as multiple measurements to check the plausibility of the observations. Furthermore, for each observation, the dataset does not include the desired operating points, but the ones measured during the experiment. In particular, the operating point given in the test bench is reached within predefined tolerance intervals. Therefore, a simple query on the dataset searching for the DOE operating points would not be successful. For this reason, a search algorithm, taking into account the desired operating points and the possible tolerance intervals is adopted. Out of the 27 817 observations in the original dataset, 20 597 datapoints corresponding to the DOE plan are isolated and the remaining 7220 including calibration and reference points are neglected. The handling of the duplicated observations is reported in the section Output Analysis. For the modeling phase, the desired operating points from the DOE are considered as training data instead of the measured ones. This allows the models to learn the test bench tolerances in the outputs, ensuring proper predictions on new given data.

Input Analysis

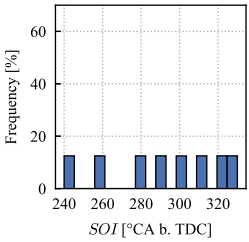
The preprocessing of the geometry parameters corresponds to the one performed for the nozzle experimental analysis in Section 5.2. Although the engine speed n assumes different values in the raw data for calibration and reference points, it is not part of the DOE plan. Thus, it is not considered for the modeling. In Figure 6.11, the data distributions of the sampled operating points excluding the duplicated measurements are represented. Similarly to the sampling of the geometry parameters, the operating points are defined in a discrete domain space. For instance, the cooling water temperature T_w can be either cold or warm. Similarly, the engine throttle valve position l_f can be either fully or partially opened, represented with the values 1 and -1 respectively. The start of injection SOI assumes a uniform distribution, while the $IMEP$ and the cooling water temperature T_w present a skew due to an imposed sampling constraint. The latter is introduced in order to focus the research on more interesting areas of the domain space. To be specific, it is known



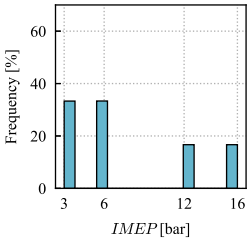
(a) Distribution of the fuel pressure P_f.



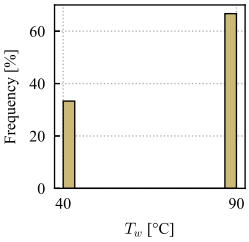
(b) Distribution of the engine throttle valve position l_f.



(c) Distribution of the start of injection SOI.



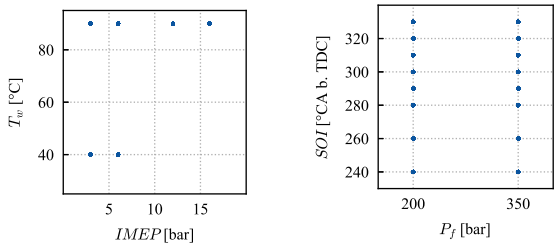
(d) Distribution of the indicated mean effective pressure $IMEP$.



(e) Distribution of the cooling water temperature T_w.

Figure 6.11: Data distributions of the engine operating points.

from the domain expertise that for high *IMEP* and low cooling water temperature T_w, large emissions are generated due to the impossibility of achieving a proper air-fuel mixture. Therefore, these operating points are not included in the DOE. This constraint is reported in Figure 6.12a, while the unconstrained sampling between the start of injection SOI and the fuel pressure P_f is reported in Figure 6.12b. Considering the introduced constraint, it is necessary to investigate the presence of multicollinearity (see Section 2.2.4). In Figure 6.13, the correlation matrix of the operating parameters is reported. The constraint between the *IMEP* and the cooling water temperature T_w produces a relevant correlation. In contrast, the rest of unconstrained parameters do not present any multicollinearity issue. Since the cooling water temperature T_w and the *IMEP* may have a strong impact on the emissions, both are kept for the modeling but their multicollinearity is taken into account during the model exploration.



(a) Constrained sampling. (b) Unconstrained sampling.

Figure 6.12: Examples of constrained and unconstrained samplings.

The cleaned dataset after the preprocessing on the input variables contains 18 792 observations and 11 features, which are summarized as follows:

$$X^{prep} = \left(\alpha\,,\ sh_l\,,\ sh_d\,,\ ph_d\,,\ \Psi\,,\ wt_l\,,\ IMEP\,,\ P_f\,,\ T_w\,,\ l_f\,,\ SOI\right) \quad (6.7)$$

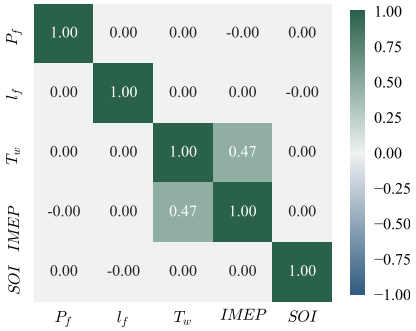


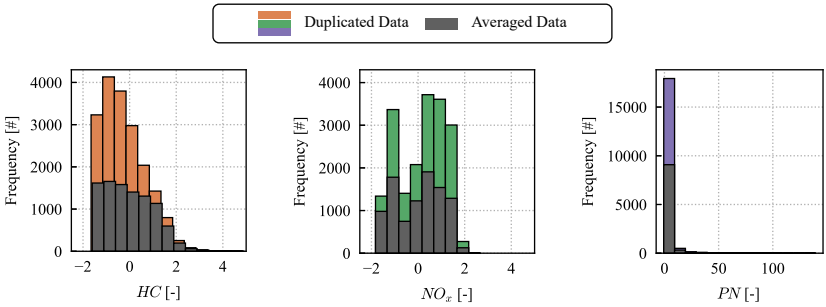
Figure 6.13: Correlation matrix of the explanatory variables.

Output Analysis

As already introduced, the dataset contains duplicated observations. Each injector is measured at the same engine operating point from two to eight

times. In order to achieve robust measurements, two metrics are considered: the average and the standard deviation of the duplicated measurements. The procedure adopted to remove the duplicated values and possible outliers is described as follows.

Duplicates. In Figure 6.14, the comparison between the distributions of the measured emissions before and after the duplicates averaging are reported. While the NO_x distribution has a multimodal behavior, the HC and the PN present a positive skew, especially the latter. The size of the reduced dataset corresponds almost to the half of the original one: the 18 792 partially cleaned observation resulting from the preprocessing of the input are reduced to 9597.

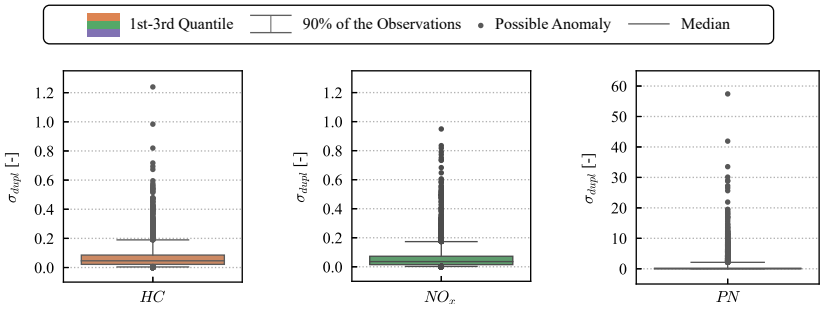


(a) Distribution of HC. (b) Distribution of NO_x. (c) Distribution of PN.

Figure 6.14: Distribution of the emissions observations with and without the duplicates.

Outliers. The data reduction can be adopted as outlier detection to investigate the plausibility of the observations. The standard deviation of the duplicated measurements, referred as σ_{dupl}, indicates how different the repeated observations are. Thus, σ_{dupl} can be considered an outlier indicator. In case of large σ_{dupl}, the measurement of the same input parameters produces discordant results and their average would introduce wrong information into the data. In Figure 6.15, the standard deviations computed on the multiple observations are reported as box plots. The box indicates the interval between the first and third quantile. The bar within the box corresponds to the median of the data, while the external bars indicate the interval where 90% of the points lies. The box plots of the standard deviations demonstrate the robustness and the reliability

of the observations: most of the measurements are able to be reproduced with a small standard deviation. However, large differences between the duplicated values are present, especially for the PN.



(a) Box-plot for HC duplicates standard deviations.

(b) Box-plot for NO_x duplicates standard deviations.

(c) Box-plot for PN duplicates standard deviations.

Figure 6.15: Box-plots of the standard deviations of the duplicated observations.

Based on the standard deviations and the averaged values of the duplicated observations, the outlier detection is performed in two steps. First, Chebyshev outlier detection is applied on the standard deviation in order to identify more accurately the non-reproducible measurements, i.e. those with large deviations. Second, the same outlier detection is applied on the remaining averaged observations to further improve the quality of the data. In Table 6.2 the identified outliers on the σ_{dupl} and on the remaining averaged measurements are summarized together with the sizes of the final reduced datasets. As expected, most of the anomalies are determined for the PN due to the large skew of its distribution. Nevertheless, more than 8500 datapoints are available for the modeling of the three emissions quantities.

Table 6.2: Results of the outlier detection on the duplicated observations.

Variable	Anomalies in σ_{dupl} [#]	Anomalies in Avg. Meas. [#]	Final Dataset Size [#]
HC	251	17	9329
NO_x	270	0	9327
PN	403	675	8519

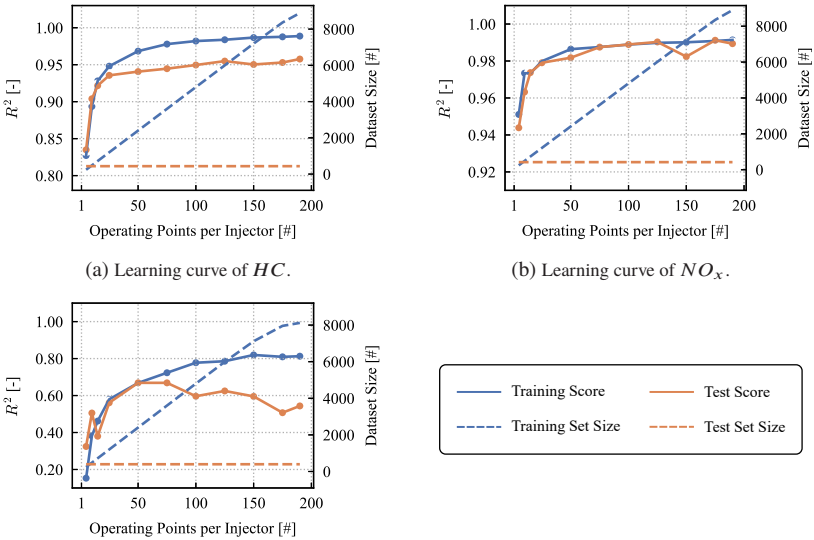
Sampling Limitation

After the preprocessing steps, the datasets are composed of 50 distinct injector designs measured at least at 190 different operating points. In order to quantify how the grid combination of geometry design with operating parameters may affect the model quality, the validation procedure introduced in Sampling Limitation in Section 5.2.1 and depicted in Figure 5.18 for the nozzle experimental analysis is applied. In this case, considering 50 Injectors, 50 strata are defined. Due to the different number of observations for each investigated characteristic after the preprocessing step (see Table 6.2), it is not possible to define a single test set to validate all the models. Therefore, two different injectors and engine operating points are chosen for each model to compose the test sets, which contain about 400 observations. The model validation is performed with 11 iterations. The training set is updated every time by sampling i operating points for each injector, with $i \in [1, 190]$, where 190 corresponds to the minimum number of measurements for each injector.

In Figure 6.16, the learning curves of the modeled emissions based on the iterative stratified sampling are depicted. The number of considered operating points per injector at each iteration is reported on the x-axis. The left y-axis indicates the training and test scores in terms of R^2, while the right y-axis represents the size of the corresponding datasets. Considering the resulting learning curves, a training set composed of a large amount of operating points per injector either causes overfitting or does not imply a considerable improvement, especially compared to the costs of the measurements. The test score of the HC follows the training score until around 25 operating points per injector, where it already achieves a satisfying accuracy. Afterwards, the test score increases with a lower gradient, until it drops at around 125 operating points per injector, indicating a slight overfit. Similarly, the NO_x requires few datapoints to provide a good model as well. Here, the test score follows the training score until 125 operating points per injector. In contrast to the other two, a strong overfitting is present for the PN. Its test score follows the training score until 50 operating points per injector. After about 75 operating points per injector, the test score starts to deviate from the training score drastically. Beside showing the most pronounced case of overfitting, the PN modeling achieves the lowest score with respect to the other emissions.

Testing the models with a combination of elements completely outside the domain space highlights possible overfitting and avoid the leakage of information in the validation phase. For the nozzle experimental analysis in Section 5.2,

the overfitting is not present due to the reduced size of the measured operating points per injector (23) compared to the ones in the engine experimental analysis (190). Given the overfitting situations and the progress of the learning curves along the increasing size of the training set, 75 operating points per injector correspond to an acceptable amount of observations in order to achieve good and reliable scores for all the modeled characteristics. Nevertheless, a further improvement in terms of precision and generalization can be achieved with the HPO.



(a) Learning curve of HC.

(b) Learning curve of NO_x.

(c) Learning curve of PN.

Figure 6.16: Learning curves based on the number of operating points per injector.

6.2.2 Model Selection and Validation

The reduced and cleaned datasets contains about 3600 observations and 11 features for each emission. In order to demonstrate the importance of the test set selection in case of combining discrete design spaces with a Cartesian product, the models are first validated through a generic $(80-20)\%$ training-test split and afterwards, through the test sets defined in Sampling Limitation

in Section 6.2.1. In both cases, the training score is evaluated with a 5-fold CV. The considered metrics are the training scores μ_{R^2}, μ_{MAE} and σ_{MAE} as well as the test scores e_{MAE} and e_{R^2}. The rest of the modeling settings are the same introduced in Section 5.1.2.

In Table 6.3, the results of the modeling validation with the (80 – 20)% training-test split are reported. The accuracy achieved by both learning approaches is similar. A larger difference is present only for the PN, where the XGBoost is able to achieve better performances. Apparently, no overfitting is present considering the resulting training and test scores. In Table 6.4, the results of the validation based on the arbitrary test sets defined in the previous section are reported. The evaluations of the training set are barely influenced by the choice of the validation procedure: the training scores reported in Table 6.3 and in Table 6.4 are similar for both learning methods. Likewise, the choice of the validation method has a neglectable impact on the test scores for the XGBoost. However, the test scores e_{R^2} and e_{MAE} of the polynomials indicate a massive overfitting. The models are not only unable to achieve a similar precision between the test and the training data, but the e_{MAE} indicates that the predictions are completely outside the domain space. In addition, the large negative e_{R^2} indicates a wrong model (see Evaluation Metrics in Section 2.3.2). The explicit selection of the observations for the test set allows to minimize the leak of information between the training and test set and to deeply evaluate the generalization performance. The polynomials are very

Table 6.3: Scores of the modeling validation with a (80 – 20)% training-test split.

	XGBoost					**Polynomial**					
Data	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	e_{R^2}	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	e_{R^2}	**degree**
HC	**0.980**	**0.099**	0.002	0.088	0.986	0.963	0.144	0.004	0.146	0.965	3
NO_x	**0.991**	**0.068**	0.002	0.068	0.992	0.987	0.087	0.002	0.086	0.987	3
PN	**0.774**	**0.247**	0.020	0.222	0.768	0.651	0.406	0.028	0.376	0.662	3

Table 6.4: Scores of the modeling validation with an arbitrary test set.

	XGBoost					**Polynomial**					
Data	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	e_{R^2}	μ_{R^2}	μ_{MAE}	σ_{MAE}	e_{MAE}	e_{R^2}	**degree**
HC	**0.984**	**0.092**	0.001	0.138	0.953	0.964	0.144	0.002	5e+4	-3e+9	3
NO_x	**0.991**	**0.067**	0.001	0.084	0.989	0.987	0.087	0.002	1e+4	-3e+8	3
PN	**0.767**	**0.244**	0.005	0.248	0.735	0.659	0.402	0.010	3e+3	-4e+7	3

sensitive to discrete and sparse observations: sharp divisions of the domain space hinder the polynomial to learn the behavior of observations not included in the training set. The XGBoost is capable instead to individuate the correct relationship in the data, independently from the adopted sampling. The models validated with the arbitrary selected test set are used for next steps.

In Figure 6.17, the μ_{MAE}, the σ_{MAE} and the e_{MAE} achieved with the XGB-NSGAII are depicted for each modeled emission. The test scores of the NO_x and the HC are not included in the confidence interval $[\mu_{MAE} \pm 2\sigma_{MAE}]$. Nevertheless, the results reported in Table 6.4 indicate a global good generalization on new data since both μ_{R^2} and e_{R^2} are above 0.95. Differently, the complexity of PN affects the modeling quality. The explored input space is not sufficient to detect the correct behavior of the PN emission: other physical effects not considered in this analysis largely contributes to the generation of PN [96], e.g. the engine oil dilution [118]. However, since those quantities cannot be directly measured, the irreducible error (see Section 2.1.3) for the PN modeling is larger with respect to the other emissions.

Similarly to temperature analysis described with the nozzle experimental analysis in Section 5.2, in this case, it is known from the domain knowledge that the cooling water temperature T_w has a large influence on the emissions. Since the sampling of the temperature is limited to two discrete values, the scores resulting from the XGB-NSGAII run on cold and warm observations are reported in Figure 6.17. As expected, the precision achieved including all the observations tends to correspond to the average of the results obtained with the separated datasets.

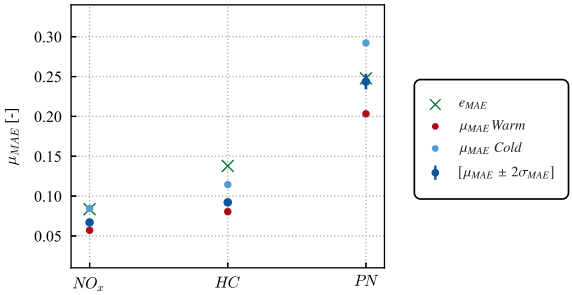


Figure 6.17: Evaluation of the XGBoost modeling quality in terms of the scores μ_{MAE}, σ_{MAE} and e_{MAE} from the experimental engine analysis.

6.2.3 Knowledge Extraction

In this section, the knowledge extraction for the analysis of the influence of valve seat geometries and operating points on engine emissions is performed in terms of models exploration and models exploitation.

Model Exploration

The XGBoost models computed on the investigated emissions are explored through feature importance and partial dependencies (see Section 2.4). In Figure 6.18, the extracted relative feature importance are depicted. On the x-axis, the valve seat geometries and the engine operating parameters are listed, while on the y-axis, their relative feature importance on each modeled output is reported. The parameters dominating the considered emissions are the cooling water temperature T_w and the *IMEP*. The *HC* is influenced almost exclusively by the T_w and with a lower magnitude by the *IMEP*. In contrast, the NO_x is mostly influenced by the *IMEP* and less by the T_w. Finally, the *PN* is not exclusively dominated by any single variable, but both injector geometries and operating points contribute to its variation.

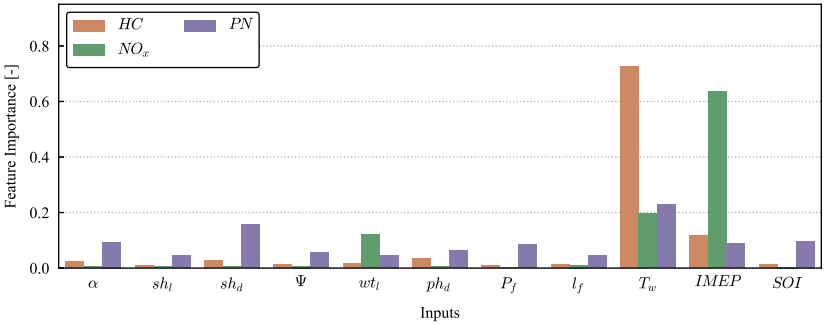


Figure 6.18: Feature importance of the experimental engine analysis.

As already introduced in Section 5.2, the extracted knowledge may result biased by the large effect of a sparsely sampled discrete variable, which corresponds in this case to the cooling water temperature T_w. Its partial dependence on the investigated emissions is reported in Figure 6.19. The represented dependence between the two measured temperatures is interpolated. On the

x-axes is reported the temperature variation and on the y-axes its effect on the investigated characteristics. A warm environment leads to a larger production of NO_x. Similarly, at high temperatures the impinged fuel evaporates, reducing the amount of PN emissions. In contrast, at lower temperatures, cold cylinder wall may cause flame quenching [119]. The latter contributes to a larger amount of HC due to improperly burned fuel.

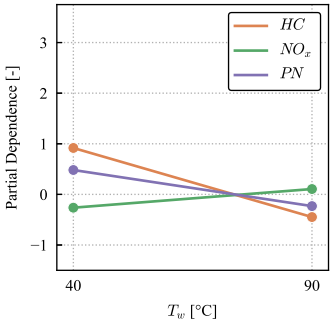
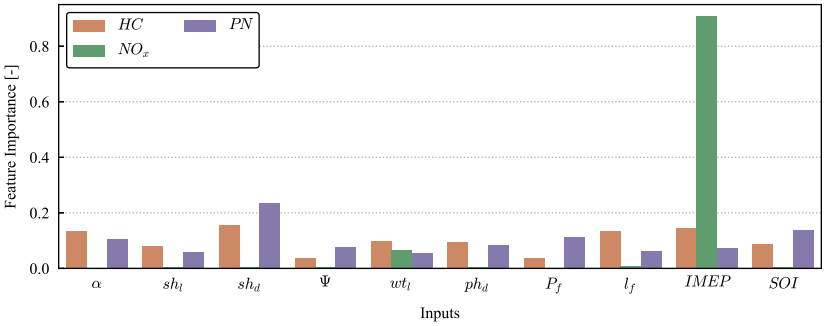
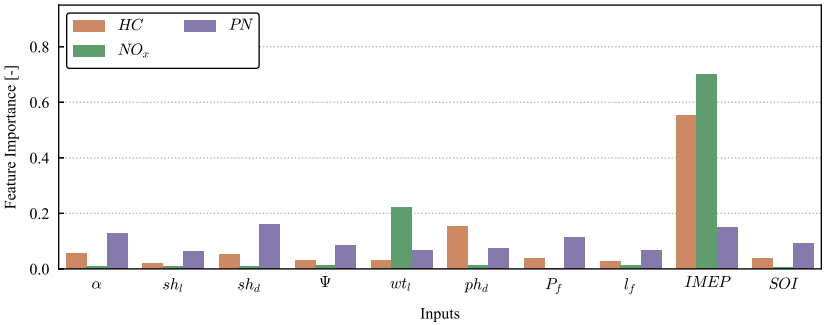


Figure 6.19: Partial dependence of the cooling water temperature T_w on the emissions.

The feature importance computed on the portions of cold and warm observations is reported in Figure 6.20. The feature importance extracted on the complete dataset represented in Figure 6.18 corresponds to the averaged importance computed on the split datasets. Since the cooling water temperature T_w has a lower influence on the NO_x and on the PN, their feature importance on the different temperatures does not show any critical difference: for cold temperatures, the spray hole diameter sh_d and the start of injection SOI have a slightly larger effect on the PN, while the wt_l influence on the NO_x increases with warm temperatures. Apparently, the emission mainly affected by the cooling water temperature T_w is the HC. In particular, no specific input variable dominates it for cold operations, while it is mostly affected by the $IMEP$ for warm temperatures. However, this result is biased by the multicollinearity between the cooling water temperature T_w and the $IMEP$, as presented in Figure 6.12a and Figure 6.13. The explored design space of the $IMEP$ at different temperatures is not balanced. Therefore, it is not possible to compare the information extracted from the cold and from the warm observations. Indeed, it is known from the domain knowledge that the $IMEP$ has a large influence on the HC for cold temperatures as well.



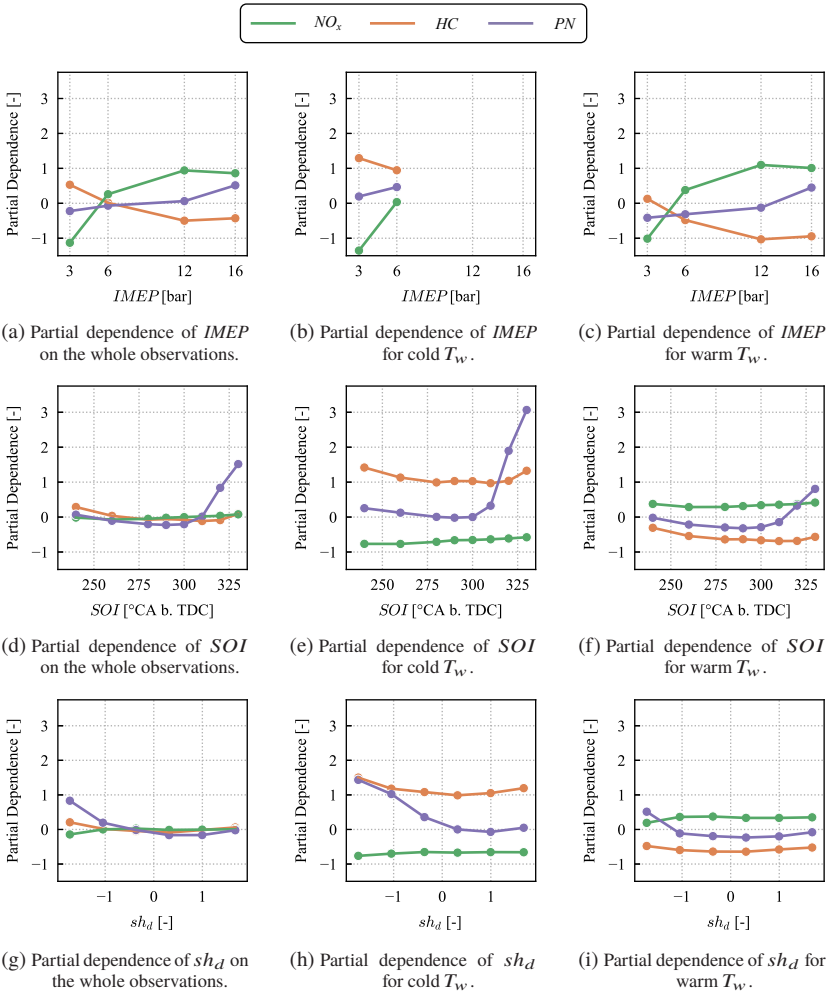
(a) Feature importance for cold cooling water temperature T_w.



(b) Feature importance for warm cooling water temperature T_w.

Figure 6.20: Experimental engine analysis feature importance based on the cooling water temperature T_w.

The partial dependencies of the *IMEP*, of the start of injection *SOI* and of the spray hole diameter sh_d are reported in Figure 6.21. They are analyzed for the complete dataset as well as for cold and warm cooling water temperatures. The remaining partial dependencies are reported in Appendix A.4.1. The partial dependencies extracted from the complete dataset correspond to the averaged partial dependencies of the two portion of data separated on the temperatures. In particular, the effects on the emissions are attenuated or amplified according to the cooling water temperature T_w. The partial dependencies are explained and validated through the domain knowledge as follows.



(a) Partial dependence of *IMEP* on the whole observations.

(b) Partial dependence of *IMEP* for cold T_w.

(c) Partial dependence of *IMEP* for warm T_w.

(d) Partial dependence of SOI on the whole observations.

(e) Partial dependence of SOI for cold T_w.

(f) Partial dependence of SOI for warm T_w.

(g) Partial dependence of sh_d on the whole observations.

(h) Partial dependence of sh_d for cold T_w.

(i) Partial dependence of sh_d for warm T_w.

Figure 6.21: Partial dependencies of *IMEP*, start of injection SOI and spray hole diameter sh_d for the investigated engine characteristics based on the whole observations as well as the warm ($90°C$) and cold ($40°C$) cooling water temperature T_w.

Indicated mean effective pressure $IMEP$. The effect of the $IMEP$ for cold cooling water temperature T_w can be extracted only for low pressure values due to the sampling constraints. Nevertheless, the behavior of the emissions based on the $IMEP$ variation is independent from the T_w. The magnitude of the effect is still influenced by the latter. Typically, higher $IMEP$ values are reached injecting more fuel, which implies higher temperatures in the combustion chamber for a constant air-fuel ratio λ equal to one. Consequently, the production of NO_x increases. In contrast, with higher temperatures in the combustion chamber, the flame quenching effect is reduced. Thus, the HC decreases as well. Finally, a larger amount of injected fuel produces more impingement, which means potentially more locally fuel rich zones. Therefore, more PN are generated.

Start of injection SOI. Independently from the cooling water temperature T_w, the SOI has almost a neglectable effect on the NO_x and on the HC. Similarly, the PN is not affected by the SOI until $300°CA\ b.\ TDC$. After this value, the piston being close to the TDC is impinged by the injected fuel. Therefore, larger PN emissions are produced. This effect is reduced for warm temperatures, where the impinged fuel evaporates earlier.

Spray hole diameter sh_d. The sh_d has a major effect on the PN. To inject the same amount of fuel, a small sh_d requires a longer injection. This affects the generation of a proper air-fuel mixture before the ignition, producing higher PN values. For lower cooling water temperature T_w a larger sh_d is required to ensure a better homogenization.

Additionally, it may be possible to investigate how the other discrete binary inputs affect the emissions by further splitting the datasets. For example, on the fuel pressure P_f or on the engine throttle valve position l_f.

Model Exploitation

A typical development task is the design of a valve seat geometry according to specific engine operating points. For instance, given a warm cooling water temperature and a high combustion chamber pressure, the valve seat geometry is optimized in case of a full opened engine throttle valve position for both fixed $IMEP$ and start of injection. The goal is to minimize the NO_x, the HC and the PN as a multi-objective optimization problem with contradictive objectives.

The optimization problem is summarized as follows:

$$\begin{aligned} \text{Minimize} \quad & f_{HC}(X),\ f_{NO_x}(X),\ f_{PN}(X) \\ \text{with} \quad & X = \big(\alpha,\ sh_l,\ sh_d,\ ph_d,\ \Psi,\ wt_l,\ P_f,\ IMEP,\ T_w, \\ & \qquad\qquad\qquad\qquad\qquad\qquad SOI,\ l_f\big) \\ \text{subject to} \quad & \text{fixed}\ \big(IMEP,\ T_w,\ l_f,\ P_f,\ SOI\big), \\ & X_i^{(l)} \le X_i \le X_i^{(u)},\ X_i \in \big[\alpha,\ sh_l,\ sh_d,\ ph_d,\ \Psi,\ wt_l\big]. \end{aligned} \tag{6.8}$$

Similarly to the model exploitation performed in the nozzle numerical analysis in Section 5.1.3, a new larger DOE of ten thousand designs is generated using the Sobol sampling method, according to the constraints defined in (6.8). The four best designs are depicted in Figure 6.22 as parallel plot together with their respective outputs. Most of the input variables assume optimized values within large intervals, i.e. they have a lower effect on the emissions. Nevertheless, the optimized spray hole diameters sh_d are larger than -1: as depicted in Figure 6.21i, this corresponds to the threshold above which lower PN emissions are ensured. According to the knowledge extraction, a small wall thickness wt_l reduces the NO_x for warm cooling water temperature (see

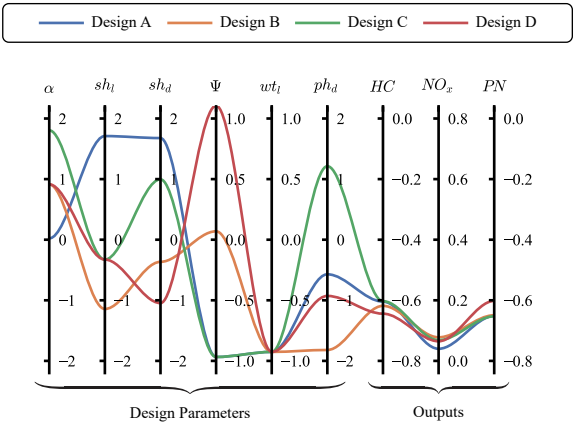


Figure 6.22: Machine learning optimized designs and their respective outputs.

Figure A.11e). For this reason, the optimization leads to a single value of wt_l for all the designs.

6.3 Summary

In this chapter, the potential of the data-driven development based on the presented KD framework to analyze the GDI engine has been presented. First, 3D-CFD simulations results are adopted to investigate the influence of spray targeting coordinates and injection strategies on impingements and air-fuel mixture homogeneity. Second, engine measurements of multi-hole injectors are considered to analyze the effect of nozzle geometries and engine operating points on nitrogen oxides, hydrocarbons and particulate number.

The structure of the investigation is similar to the one utilized in Chapter 5 for the analysis of the nozzle characteristics. For the engine measurements analysis, an example of handling duplicated observations is presented. These may interfere with the models validation. Therefore, specific preprocessing procedures are applied in order to ensure the extraction of reliable knowledge. Additionally, it is demonstrated how the combination of large discrete input domains through a Cartesian product affects the quality and the reliability of the models. In particular, an iterative stratified model validation allows to choose the correct amount of required data to avoid the overfit.

7 Limitations and Risks

In this chapter the limitations and risks of the application of machine learning in the product development based on the use cases presented in this work are summarized.

The first challenge corresponds to the sampled input space: the models are not able to represent domain spaces not included in the training phase. The collected observations have to cover, at least partially, the phenomena to be investigated. For instance, consider the engine numerical analysis in Section 6.1. In this case, the spray targeting coordinates are sampled such that the spray plumes are not directed towards the intake valve. Therefore, even though some information regarding the intake valve impingement is present in the dataset, this does not correspond to a proper representation of the phenomena. The DOE plan may be extended by including additional samples with different intake impingement. Similarly, the amount of information available for the modeling of the spray plume angle and the fuel plume penetration in the nozzle numerical analysis in Section 5.1 as well as of the particulate number in the engine experimental analysis in Section 6.2 was not enough to achieve good accuracy models compared to the other variables. Therefore, the domain expertise is essential to evaluate a proper coverage of the input space in order to ensure enough variation in the output and minimize the irreducible error (see Section 2.1.3).

The domain expertise is fundamental in the data preprocessing as well. Being aware of which redundant information can be neglected without affecting the final results is essential. For instance, keeping both the cooling water temperature and the *IMEP* in the engine experimental analysis in Section 6.2, despite their high correlation, allowed to deeper investigate the effect of the temperature on the engine emissions.

Another potential limitation corresponds to the application of the models. For example, a model trained on data collected on a specific engine geometry cannot be always applied to predict observations for a different engine. This can be done after ensuring that the model is able to generalize its precision in an extended or in a different input space. Similarly, the extracted knowledge

in form of feature importance and partial dependencies is valid only within the considered parameter space. Models trained on other observations or including additional explanatory variables may generate different results. Therefore, the extracted knowledge has to be contextualized within the considered investigation.

Finally, the application of wrong learning methods or validation procedures may also generate invalid results. An example has been reported with the engine experimental analysis in Section 6.2. In this case, the selection of a non-suitable test set or the adoption of a simple method like the polynomial regression led to overfitting situations.

These limitations can be summarized in three main points: inappropriate sampling of the input observations, the lack of domain expertise or the application of not suitable algorithms.

8 Conclusions and Outlook

The new emission regulations as well as the demand of high engine efficiency and performance require advanced analysis tools able to open new frontiers for improvement and optimization of GDI systems. Despite the continuous development of simulation tools and measurement procedures, the complex interactions in the combustion systems require a significant effort in the generation and in the evaluation of the results. Modern machine learning techniques exploit the rising computational power capabilities, enabling precise modeling and advanced knowledge extraction from already available and new data.

In this work a KD framework based on machine learning has been developed in order to meet the specific demands of the GDI systems development. The KD framework is written in the programming language Python and named pyMICE, which stands for ***py****thon* ***M****ining* ***I****nternal* ***C****ombustion* ***E****ngines*. The essential feature of the framework is its modularity. Every single step is considered to be a module, which is developed to be abstract, autonomous and flexible. This allows the framework to be compatible with different data, independently from their physical meaning, source, format and naming. Furthermore, the modularity allows the framework to be based on the dataflow. The KD process is iterative and each step may require the supervision of the domain expertise to define the next actions: the direction of the analysis can be dynamically changed by skipping some operations or by moving to a previous stage, according to the intermediate results. The modular design is strengthened by a proper storage system able to extract raw data with different structures and transform them into standardized formats as well as to store and restore intermediate results. The fundamental modules includes data extraction and transformations, data analysis and optimization. The usability is another building block of the framework. Specific interfaces are available in order to support the user achieving the required task. Special data preprocessing procedures combining the several operations implemented in the framework can be defined through APIs. The whole KD process can be autonomously run through a CLI. Finally, the extracted knowledge can be visualized and analyzed with a web-based GUI.

In order to handle the heterogeneity of the datasets available in the GDI context, a novel, parameter-free, fast and dynamic data-driven model selection is presented. In particular, this is a universal algorithm able to select the best meta-parameters of a learning algorithms for a specific dataset. This approach is referred as XGB-NSGAII and it couples the XGBoost with the genetic algorithm NSGA-II. Since genetic algorithms are known to demand high computational resources, the proposed algorithm is designed to run on distributed computing. The goal of this procedure is to ensure robust and precise function estimations of physical phenomena. This is achieved with the definition of a proper optimization problem, evaluating both the accuracy of the models and their generalization on new data. The presented approach outperforms state-of-the-art methods like Random Search, Grid Search and Hyperband. Results showing the effect of the choice of different genetic parameters are reported as well. This approach is not restricted to combustion engine development only, but it is shown that it can be extended to other data and applications.

Besides the typical machine learning prediction tasks, the KD framework allows the interpretability of its results from a human point-of-view, stepping beyond the concept of black-box AI. The knowledge extracted from the data can lead future investigations towards unexplored areas, producing new data to be analyzed. Therefore, it is possible to achieve an iterative continuous product improvement. The domain expertise is still required to interpret and to validate the extracted knowledge as well as to properly define the investigated problem and to collect correct and valid observations.

In this work, four use cases have been presented as successful applications of the KD framework. The considered data concern different sources as well as components and systems. First, the results of nozzle 3D-CFD simulations and spray chamber experiments are adopted to investigate the influence of injector nozzle geometries and operating parameters on flow dynamics and spray characteristics. Afterwards, the results of engine 3D-CFD simulations and experiments are considered to investigate the influence of nozzle geometries, spray targeting and engine operating points on mixture formation, emissions and fuel consumption. The knowledge extracted with these analyses has been validated with the domain expertise and it revealed a proper representation of the physics behind the data. Finally, the machine learning models are applied to select optimal geometrical and operational designs able to achieve predefined constraints and goals, ensuring higher performance and lower emissions.

The framework can be extended by adding new learning algorithms. Considering the large focus of the research community on machine learning, any new

discovered method can be implemented enhancing the overall performance of the framework.

In case of new research fields, the domain expertise may not be always available. For this purpose, novel approaches able to deal with unknown domain spaces are available in the literature. Instead of extracting information from already available data or to plan a large DOE aiming to cover as best the domain space, advanced machine learning methodologies like *Active Learning* [120] interact actively with the source of the data in order to explore the domain space according to the scope of the analysis. In particular, based on the responses of the previous samples, the learning algorithm evaluates iteratively which observations are required in order to achieve some predefined goals, e.g. minimize the engine emissions or improve the predictive accuracy of the model. The resulting data can be used as input for the methodology presented in this work.

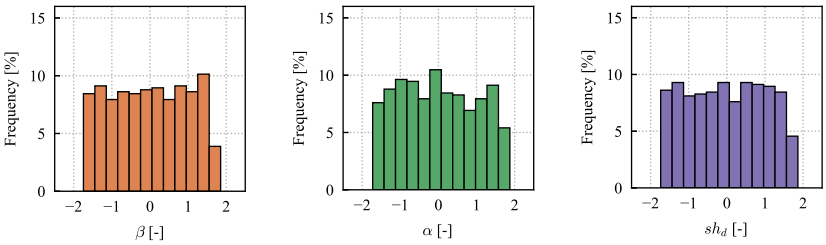
The KD framework is developed and tailored to overcome the main issues in the GDI context. Nevertheless, it can be extended to any other investigation area in the product development. Whenever the influence of geometric parameters, operating points, processes settings as well as application conditions has to be analyzed and optimized, the KD framework can be adopted to deeper understand their physical effects on the problem objectives. Concrete applications examples on new technologies are hydrogen gas injectors, electric machines or safety-critical hardware for advanced driver assistance systems, such as cameras and processing units. Hydrogen gas injectors are essential components of a fuel cell electric vehicle. Similarly to gasoline injectors, specific geometries and operating points have to be identified in order to develop efficient and low consumption solutions. Electric machines have to be adapted to the different powertrain solutions, such as fuel cell, hybrid or battery electric, ensuring high performance and long range driving. The development of safety-critical hardware for autonomous driving requires intensive studies on material properties as well as on designs in order to provide robust, reliable and lasting solutions.

Appendices

A Additional Plots

A.1 Injector Nozzle Analysis

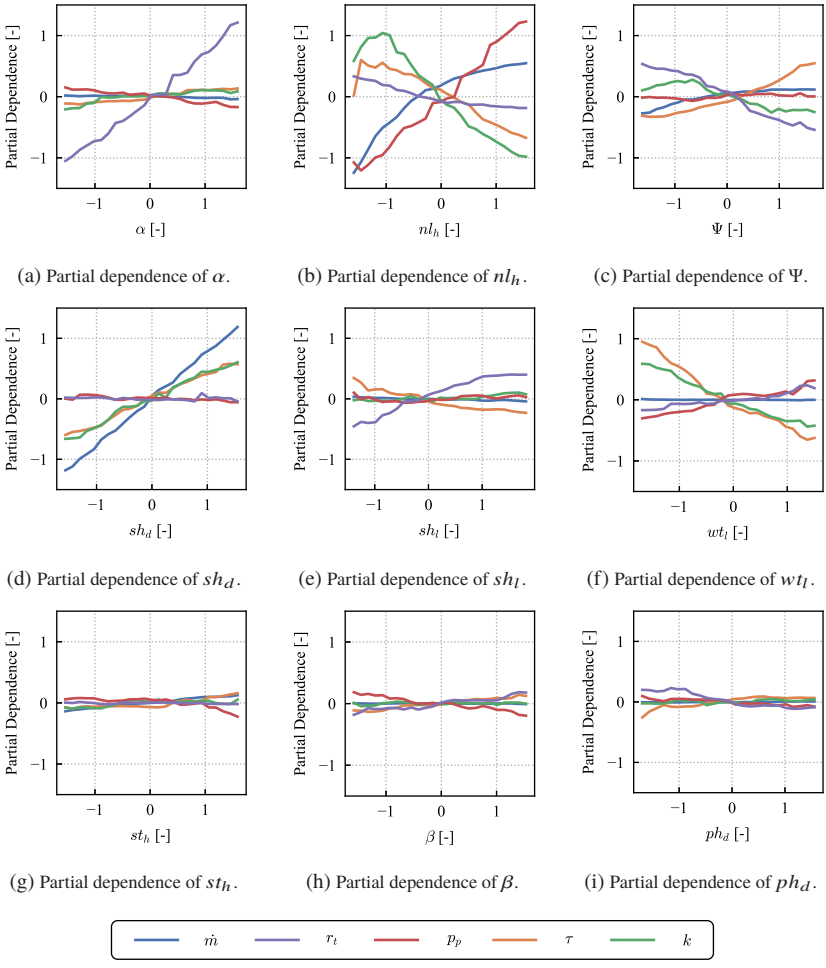
A.1.1 Input Data



(a) Distribution of β. (b) Distribution of α. (c) Distribution of sh_d.

Figure A.1: Distribution of the input data not included in Figure 5.3.

A.1.2 Partial Dependencies



(a) Partial dependence of α. (b) Partial dependence of nl_h. (c) Partial dependence of Ψ.

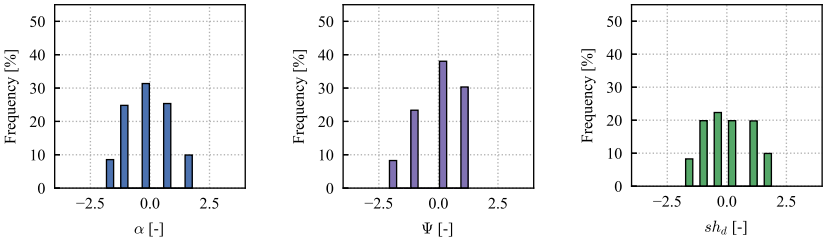
(d) Partial dependence of sh_d. (e) Partial dependence of sh_l. (f) Partial dependence of wt_l.

(g) Partial dependence of st_h. (h) Partial dependence of β. (i) Partial dependence of ph_d.

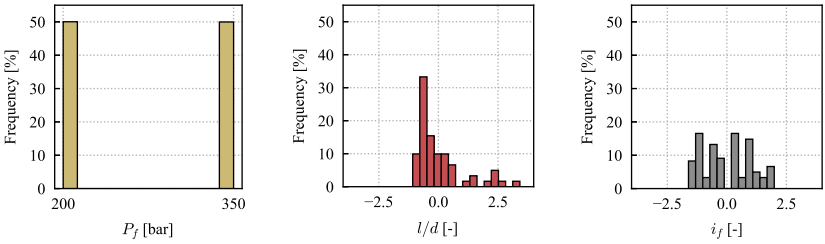
Figure A.2: Complete partial dependencies, as extension of Figure 5.11.

A.2 Nozzle Spray Experimental Analysis

A.2.1 Input Data



(a) Distribution of α. (b) Distribution of Ψ. (c) Distribution of sh_d.

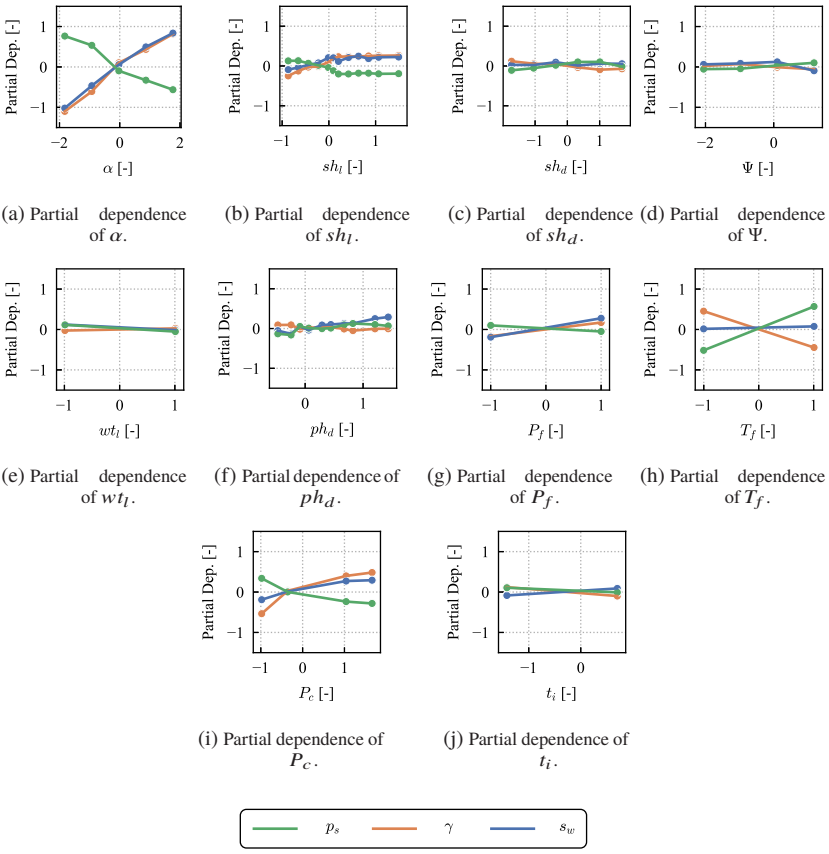


(d) Distribution of P_f. (e) Distribution of l/d. (f) Distribution of i_f.

Figure A.3: Distribution of the input data not included in Figure 5.14.

A.2.2 Partial Dependencies

Complete Dataset



(a) Partial dependence of α.

(b) Partial dependence of sh_l.

(c) Partial dependence of sh_d.

(d) Partial dependence of Ψ.

(e) Partial dependence of wt_l.

(f) Partial dependence of ph_d.

(g) Partial dependence of P_f.

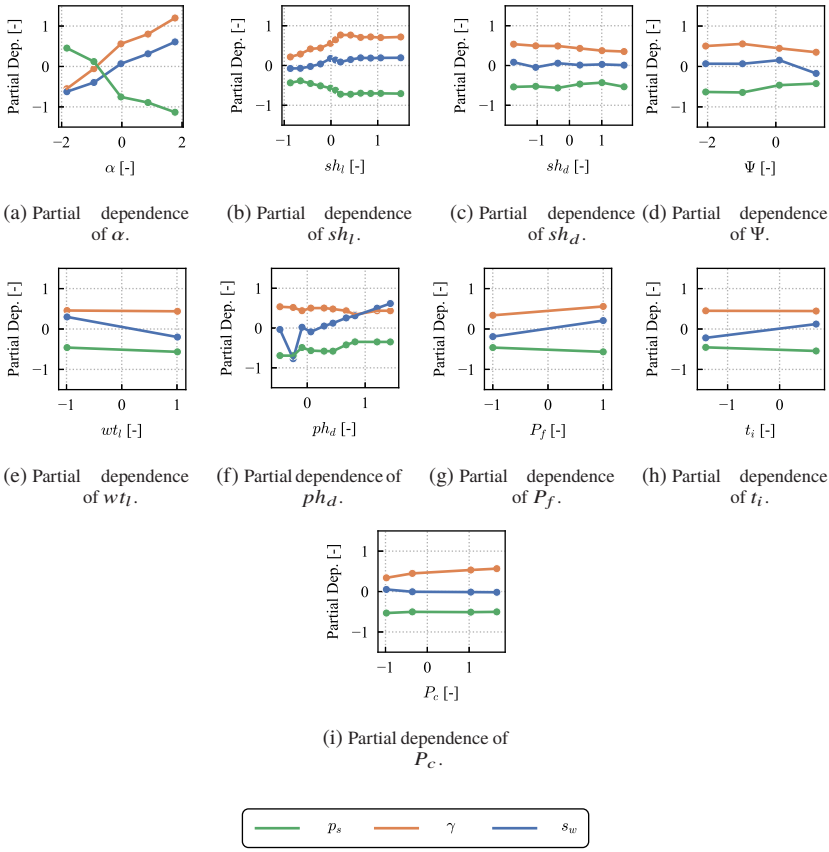
(h) Partial dependence of T_f.

(i) Partial dependence of P_c.

(j) Partial dependence of t_i.

Figure A.4: Complete partial dependencies for the whole data, as extension of Figure 5.24.

Cold Dataset



(a) Partial dependence of α.

(b) Partial dependence of sh_l.

(c) Partial dependence of sh_d.

(d) Partial dependence of Ψ.

(e) Partial dependence of wt_l.

(f) Partial dependence of ph_d.

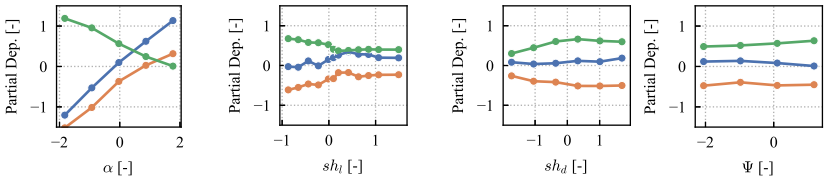
(g) Partial dependence of P_f.

(h) Partial dependence of t_i.

(i) Partial dependence of P_c.

Figure A.5: Complete partial dependencies for the cold observations, as extension of Figure 5.24.

Warm Dataset

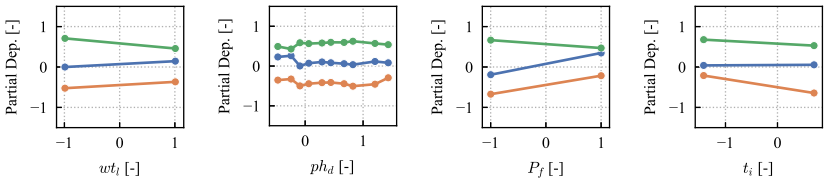


(a) Partial dependence of α.

(b) Partial dependence of sh_l.

(c) Partial dependence of sh_d.

(d) Partial dependence of Ψ.

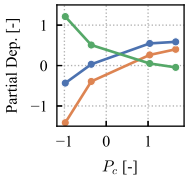


(e) Partial dependence of wt_l.

(f) Partial dependence of ph_d.

(g) Partial dependence of P_f.

(h) Partial dependence of t_i.

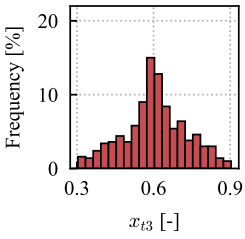


(i) Partial dependence of P_c.

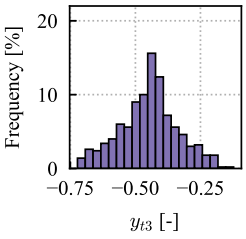
Figure A.6: Complete partial dependencies for the warm observations, as extension of Figure 5.24.

A.3 Spray Mixture Numerical Analysis

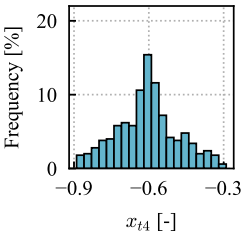
A.3.1 Input Data



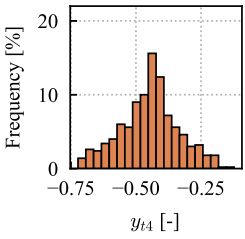
(a) Distribution of x_{t3}.



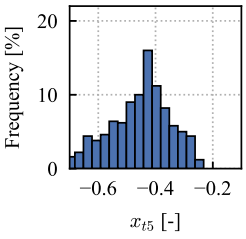
(b) Distribution of y_{t3}.



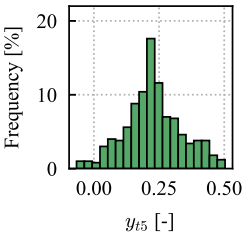
(c) Distribution of x_{t4}.



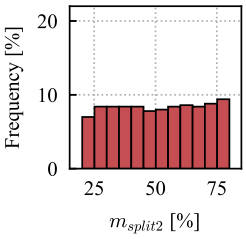
(d) Distribution of y_{t4}.



(e) Distribution of x_{t5}.



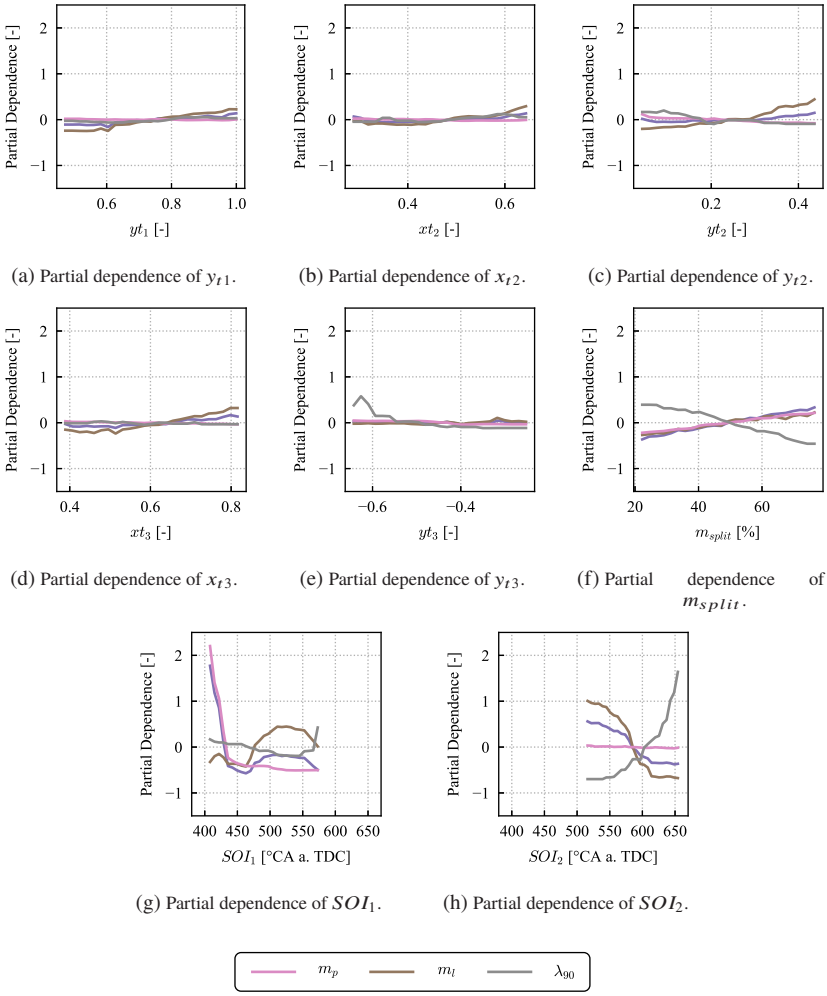
(f) Distribution of y_{t5}.



(g) Distribution of m_{split2}.

Figure A.7: Distribution of the input data not included in Figure 6.4

A.3.2 Partial Dependencies



(a) Partial dependence of y_{t1}. (b) Partial dependence of x_{t2}. (c) Partial dependence of y_{t2}.

(d) Partial dependence of x_{t3}. (e) Partial dependence of y_{t3}. (f) Partial dependence of m_{split}.

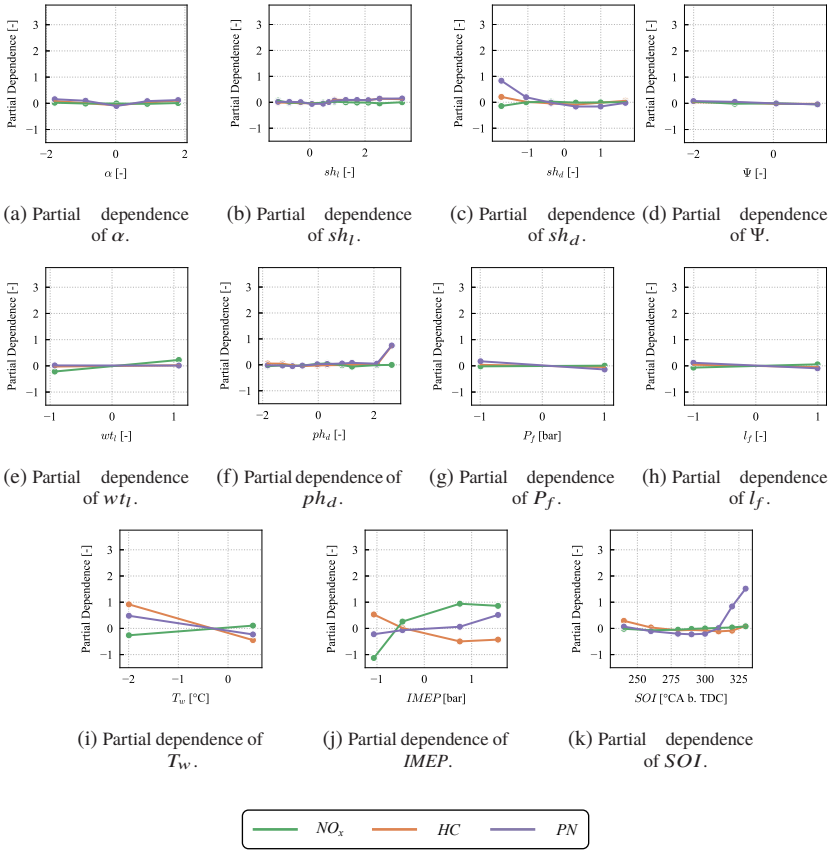
(g) Partial dependence of SOI_1. (h) Partial dependence of SOI_2.

Figure A.8: Complete partial dependencies, as extension of Figure 6.9.

A.4 Emissions Experimental Analysis

A.4.1 Partial Dependencies

Complete Dataset



(a) Partial dependence of α.

(b) Partial dependence of sh_l.

(c) Partial dependence of sh_d.

(d) Partial dependence of Ψ.

(e) Partial dependence of wt_l.

(f) Partial dependence of ph_d.

(g) Partial dependence of P_f.

(h) Partial dependence of l_f.

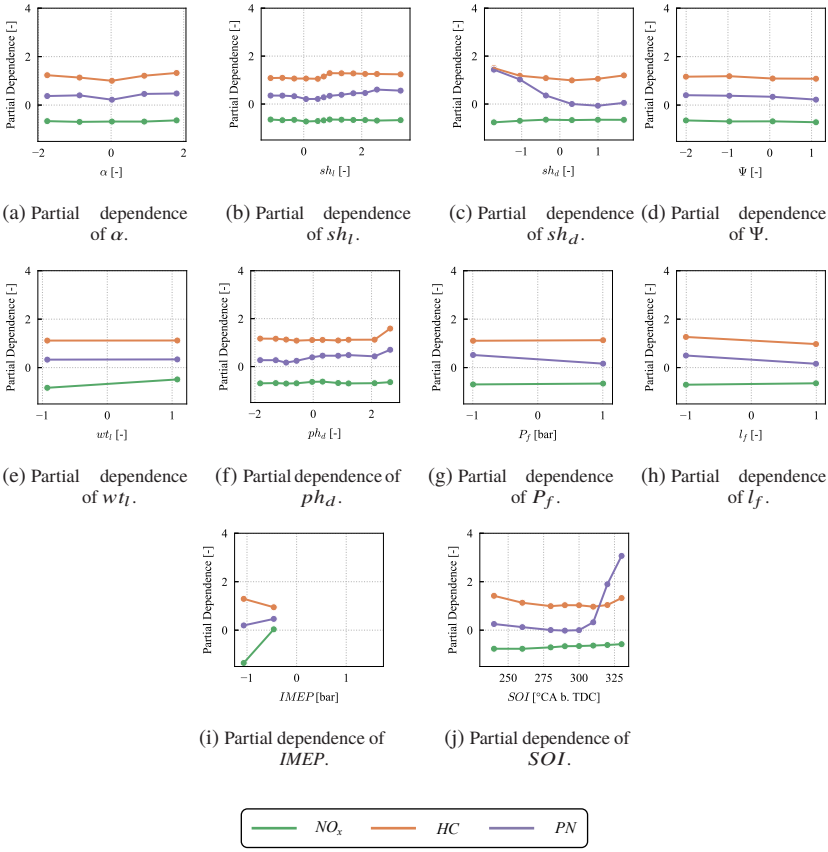
(i) Partial dependence of T_w.

(j) Partial dependence of $IMEP$.

(k) Partial dependence of SOI.

Figure A.9: Complete partial dependencies for the whole data, as extension of Figure 6.21.

Cold Dataset



(a) Partial dependence of α.

(b) Partial dependence of sh_l.

(c) Partial dependence of sh_d.

(d) Partial dependence of Ψ.

(e) Partial dependence of wt_l.

(f) Partial dependence of ph_d.

(g) Partial dependence of P_f.

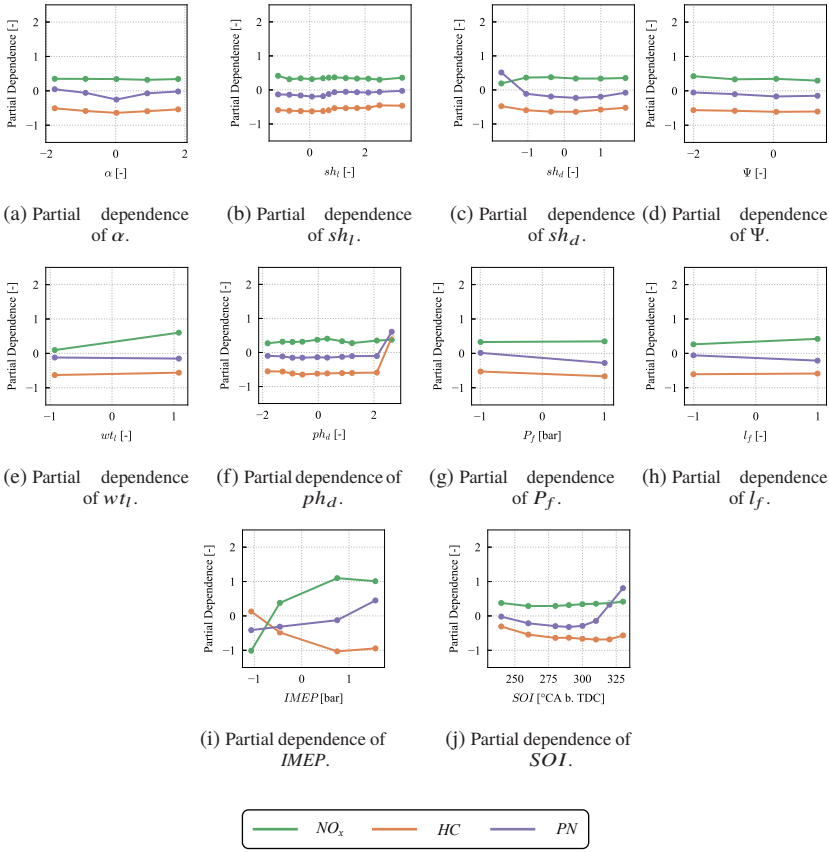
(h) Partial dependence of l_f.

(i) Partial dependence of *IMEP*.

(j) Partial dependence of SOI.

Figure A.10: Complete partial dependencies for the cold observations, as extension of Figure 6.21.

Warm Dataset



(a) Partial dependence of α.

(b) Partial dependence of sh_l.

(c) Partial dependence of sh_d.

(d) Partial dependence of Ψ.

(e) Partial dependence of wt_l.

(f) Partial dependence of ph_d.

(g) Partial dependence of P_f.

(h) Partial dependence of l_f.

(i) Partial dependence of *IMEP*.

(j) Partial dependence of *SOI*.

Figure A.11: Complete partial dependencies for the warm observations, as extension of Figure 6.21.

B User Interface

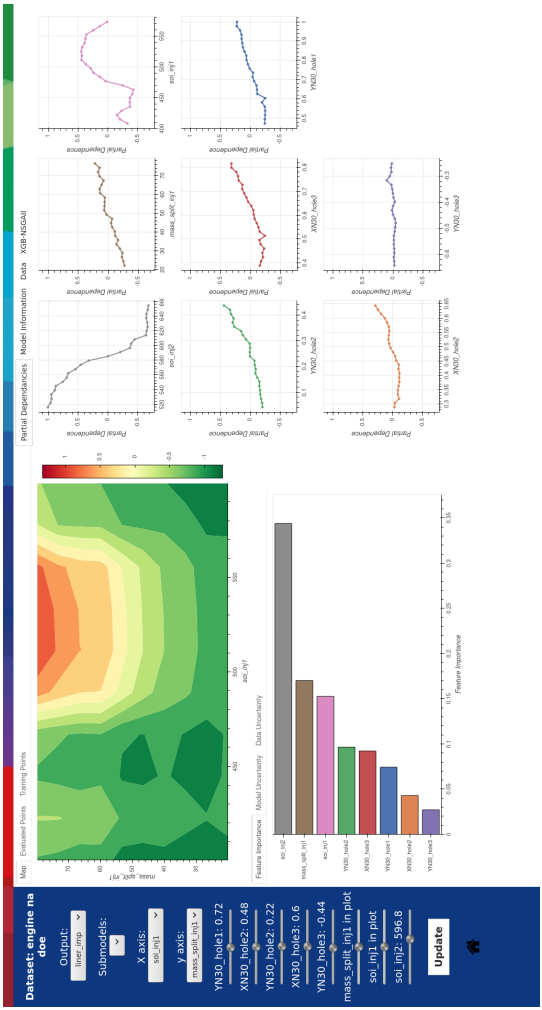


Figure B.1: Graphic User Interface of the KD framework.

References

[1] European Environment Agency, "Greenhouse gas emission intensity of fuels and biofuels for road transport in Europe." https://www.eea.europa.eu/ds_resolveuid/31dee9b1f9cf469e929c804aa985fd30, Nov. 2020. [Online; accessed: 04-12-2020].

[2] European Environment Agency, "Emissions of air pollutants from transport." https://www.eea.europa.eu/ds_resolveuid/49029c39065b4d728a5e8543736caf0b, Dec. 2019. [Online; accessed: 04-12-2020].

[3] European Union, "Long-term low greenhouse gas emission development strategy of the EU and its Member States." https://unfccc.int/documents/210328, Mar. 2020. [Online; accessed: 04-12-2020].

[4] European Environment Agency, *Electric Vehicles in Europe*. Publications Office of the European Union, Luxembourg, Sept. 2016.

[5] N. Ortar and M. Ryghaug, "Should All Cars Be Electric by 2025? The Electric Car Debate in Europe," *Sustainability*, vol. 11, p. 1868, Mar. 2019.

[6] T. Pauer, H. Yilmaz, J. Zumbrägel, and E. Schünemann, "New Generation Bosch Gasoline Direct-Injection Systems," *MTZ worldwide*, vol. 78, pp. 16–23, July 2017.

[7] F. Leach, T. Knorsch, C. Laidig, and W. Wiese, "A Review of the Requirements for Injection Systems and the Effects of Fuel Quality on Particulate Emissions from GDI Engines," in *International Powertrains, Fuels & Lubricants Meeting*, PF&L, (September 17-19, 2018, Heidelberg, Germany), Society of Automotive Engineers (SAE), Warrendale, PA, United States, Sept. 2018.

[8] K. Reif, *Gasoline Engine Management. Systems and Components.* Bosch Professional Automotive Information, Springer Vieweg, Wiesbaden, Germany, 2015.

[9] Robert Bosch GmbH, "Bosch mobility solutions." http://www.bosch-mediaspace.de/. [Online; accessed: 12-05-2020].

[10] Robert Bosch GmbH, "Bosch mobility solutions." http://www.bosch-mediaspace.de/. [Online; accessed: 27-07-2015].

[11] R. Basshuysen, *Ottomotor mit Direkteinspritzung. Verfahren, Systeme, Entwicklung, Potenzial.* Springer Vieweg, Wiesbaden, Germany, 3. ed., 2013.

[12] M. Bertsch, M. Lippisch, W. Wiese, and E. Schünemann, "Combined application of 3D-CFD and high-speed video endoscopy on a turbocharged GDI engine," in *Internationaler Motorenkongress 2019*, (February 26-27, 2019, Baden-Baden, Germany), pp. 139–154, Springer Fachmedien, Wiesbaden, Germany, May 2019.

[13] A. Kufferath, W. Wiese, W. Samenfink, H. Dageförde, T. Knorsch, and P. Jochmann, "Assessment of feasible system solutions for future particle emission requirements," in *Fuel Systems for IC Engines Conference*, (March 10-11, 2015, London, UK), IMechE, London, UK, Mar. 2015.

[14] R. Hellmann, P. Jochmann, P. Leick, K. G. Stapf, E. Schünemann, and D. Thévenin, "Multi-Objective optimization of high-pressure gasoline injector nozzles using Genetic Algorithm coupled with Computational Fluid Dynamics (CFD): exploiting the manufactural design space," in *Proceeding of the 14th International Conference on Liquid Atomization and Spray Systems*, ICLASS 2018, (July 22-26, 2018, Chicago, IL, United States), University of Illinois, Chicago, IL, United States, July 2018.

[15] G. G. Wang and S. Shan, "Review of Metamodeling Techniques in Support of Engineering Design Optimization," *Journal of Mechanical Design*, vol. 129, pp. 370–380, May 2006.

[16] R. Hellmann, P. Jochmann, K. G. Stapf, E. Schünemann, L. Daróczy, and D. Thévenin, "Towards design optimization of high-pressure gasoline injectors using Genetic Algorithm coupled with Computational Fluid Dynamics (CFD)," in *Proceeding of the 28th European conference on Liquid Atomization and Spray Systems*, ILASS-Europe 2017, (September 06-08, 2017, València, Spain), Universitat Politècnica de València, València, Spain, Sept. 2017.

[17] T. Hwang, "Computational Power and the Social Impact of Artificial Intelligence," *arXiv, arXiv:1803.08971*, Mar. 2018.

[18] J. Gantz and D. Reinsel, "Extracting value from chaos," *IDC's iView*, vol. 1142, 2011. Digital Universe Study sponsored by EMC Corporation.

[19] M. Karpinski-Leydier, R. Nagamune, and P. Kirchen, "A Machine Learning Modeling Approach for High Pressure Direct Injection Dual Fuel Compressed Natural Gas Engines," in *International Powertrains, Fuels & Lubricants Meeting*, PF&L, (September 22-24, 2020, Virtual Event), Society of Automotive Engineers (SAE), Warrendale, PA, United States, Sept. 2020.

[20] M. Lucido and J. Shibata, "Learning Gasoline Direct Injector Dynamics Using Artificial Neural Networks," in *WCX World Congress Experience*, WCX 18, (April 10-12, 2018, Detroit, MI, United States), Society of Automotive Engineers (SAE), Warrendale, PA, United States, Apr. 2018.

[21] Y.-H. Pu, J. Keshava Reddy, and S. Samuel, "Machine learning for nano-scale particulate matter distribution from gasoline direct injection engine," *Applied Thermal Engineering*, vol. 125, pp. 336–345, Oct. 2017.

[22] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The KDD Process for Extracting Useful Knowledge from Volumes of Data," *Communications of the ACM*, vol. 39, pp. 27–34, Nov. 1996.

[23] C. Kamath, *Scientific Data Mining: A Practical Perspective*. Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, United States, 2009.

[24] O. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook*. Springer New York, NY, United States, 2. ed., 2010.

[25] M. Bramer, *Principles of Data Mining*. Springer London, UK, 4. ed., 2020.

[26] P. Bernus and O. Noran, "Data Rich - But Information Poor," in *Collaboration in a Data-Rich World*, (September 18-20, 2017, Vicenza, Italy), pp. 206–214, Springer Cham, Switzerland, Sept. 2017.

[27] C. C. Aggarwal, *Data Mining: The Textbook*. Springer Cham, Switzerland, 2015.

[28] M. Gaber, *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*. Springer Berlin, Heidelberg, Germany, 2010.

[29] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. Springer New York, NY, United States, 2013 (Corrected at 8th printing 2017).

[30] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer New York, NY, United States, 2. ed., 2009 (Corrected at 12th printing 2017).

[31] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Springer Cham, Switzerland, 2015.

[32] C. Fox, A. Levitin, and T. Redman, "The notion of data and its quality dimensions," *Information Processing & Management*, vol. 30, pp. 9–19, Jan. 1994.

[33] W. Kim, B.-J. Choi, E.-K. Hong, S.-K. Kim, and D. Lee, "A Taxonomy of Dirty Data," *Data Mining and Knowledge Discovery*, vol. 7, pp. 81–99, Jan. 2003.

[34] D. M. Hawkins, *Identification of outliers*. Springer Dordrecht, Netherlands, 1980.

[35] B. G. Amidan, T. A. Ferryman, and S. K. Cooley, "Data outlier detection using the Chebyshev theorem," in *2005 IEEE Aerospace Conference*, (March 05-12, 2005, Big Sky, MT, United States), pp. 3814–3819, IEEE, Piscataway, NJ, United States, Dec. 2005.

[36] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-Based Anomaly Detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, Mar. 2012.

[37] H. Knebl, "Trees," in *Algorithms and Data Structures: Foundations and Probabilistic Methods for Design and Analysis*, pp. 129–204, Springer Cham, Switzerland, 2020.

[38] R. Murphy, *On Tests for Outlying Observations*. PhD thesis, Princeton University, Princeton, NJ, United States, 1951.

[39] R. Serfling and S. Wang, "General foundations for studying masking and swamping robustness of outlier identifiers," *Statistical Methodology*, vol. 20, pp. 79–90, Sept. 2014.

[40] K. Unnebrink and J. Windeler, "Intention-to-treat: methods for dealing with missing values in clinical trials of progressively deteriorating diseases," *Statistics in Medicine*, vol. 20, pp. 3931–3946, Dec. 2001.

[41] H. Kim, G. H. Golub, and H. Park, "Missing value estimation for DNA microarray gene expression data: local least squares imputation," *Bioinformatics*, vol. 21, pp. 187–198, Jan. 2005.

[42] T. Y. T. Lin, "Attribute transformations for data mining I: Theoretical explorations," *International Journal of Intelligent Systems*, vol. 17, pp. 213–222, Jan. 2002.

[43] R. E. Bellman, *Adaptive control processes: a guided tour*. Princeton Legacy Library, Princeton University Press, Princeton, NJ, United States, 1961.

[44] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, vol. 454 of *The Springer International Series in Engineering and Computer Science*. Springer New York, NY, United States, 1998.

[45] A. Jain and D. Zongker, "Feature selection: evaluation, application, and small sample performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 153–158, Feb. 1997.

[46] L. Rokach and O. Maimon, *Data Mining with Decision Trees*, vol. 81 of *Series in Machine Perception and Artificial Intelligence*. World Scientific, Singapore, 2. ed., 2013.

[47] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, pp. 1189–1232, Oct. 2001.

[48] M. H. Kutner, W. Li, C. J. Nachtsheim, and J. Neter, *Applied Linear Statistical Models*. Mcgraw-Hill/Irwin, New York, NY, United States, 5. ed., 2005.

[49] P. Sabine and C. Plumpton, *Statistics*. Core Books in Advanced Mathematics, Palgrave Macmillan, London, UK, 1985.

[50] K. Pearson and F. Galton, "VII. Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, pp. 240–242, Dec. 1895.

[51] M. H. Herzog, G. Francis, and A. Clarke, "Correlation," in *Understanding Statistics and Experimental Design : How to Not Lie with Statistics*, pp. 95–102, Springer Cham, Switzerland, 2019.

[52] B. D. Haig, "What Is a Spurious Correlation?," *Understanding Statistics*, vol. 2, pp. 125–132, Apr. 2003.

[53] I. T. Jolliffe, "Principal Components in Regression Analysis," in *Principal Component Analysis*, pp. 167–198, Springer, New York, NY, United States, 2. ed., 2002.

[54] H. Liu and H. Motoda, "On Issues of Instance Selection," *Data Mining and Knowledge Discovery*, vol. 6, pp. 115–130, Apr. 2002.

[55] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., Sebastopol, CA, United States, 2. ed., 2019.

[56] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning series, The MIT Press, Cambridge, MA, United States, 2012.

[57] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education, Hoboken, NJ, United States, 3. ed., 2010.

[58] 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*, KDD 16, (August 13-17, 2016, San Francisco, CA, United States), pp. 785–794, Association for Computing Machinery (ACM), New York, NY, United States, Aug. 2016.

[59] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, pp. 367–378, Feb. 2002.

[60] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification And Regression Trees*. Chapman & Hall/CRC, Boca Raton, FL, United States, 1984.

[61] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.

[62] T. Hastie and R. Tibshirani, "Generalized Additive Models," *Statistical Science*, vol. 1, pp. 297–310, Aug. 1986.

[63] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. The MIT Press, Cambridge, MA, United States, 2012.

[64] L. Breiman, "Arcing classifier (with discussion and a rejoinder by the author)," *The Annals of Statistics*, vol. 26, pp. 801–849, June 1998.

[65] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quarterly of Applied Mathematics*, vol. 2, pp. 258–261, Oct. 1944.

[66] J. B. Copas, "Regression, Prediction and Shrinkage," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 45, pp. 311–354, July 1983.

[67] T. Hastie and R. Tibshirani, *Generalized Additive Models*. No. 43 in Monographs on Statistics and Applied Probability, Chapman & Hall/CRC, Boca Raton, FL, United States, 1990.

[68] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, Aug. 1996.

[69] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," *arXiv, arXiv:1309.0238*, Sept. 2013.

[70] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS 17, (December 04-09, 2017, Long Beach, CA, United States), pp. 3146–3154, Curran Associates Inc., Red Hook, NY, United States, Dec. 2017.

[71] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased Boosting with Categorical Features," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS 18, (December 03-08, 2018, Montréal, Canada), pp. 6639–6649, Curran Associates Inc., Red Hook, NY, United States, Dec. 2018.

[72] D. H. Wolpert, "The Lack of A Priori Distinctions Between Learning Algorithms," *Neural Computation*, vol. 8, pp. 1341–1390, Oct. 1996.

[73] A. Zheng, *Evaluating Machine Learning Models*. O'Reilly Media, Inc., Sebastopol, CA, United States, 2015.

[74] M. Feurer and F. Hutter, "Hyperparameter Optimization," in *Automated Machine Learning: Methods, Systems, Challenges*, pp. 3–33, Springer Cham, Switzerland, 2019.

[75] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, Feb. 2012.

[76] D. C. Montgomery, *Design and Analysis of Experiments*. John Wiley & Sons Inc., Hoboken, NJ, United States, 8. ed., 2013.

[77] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, P. Prabhat, and R. P. Adams, "Scalable Bayesian

Optimization Using Deep Neural Networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, ICML 15, (July 06-11, 2015, Lille, France), pp. 2171–2180, Journal of Machine Learning Research, July 2015.

[78] T. Jansen, "Evolutionary Algorithms and Other Randomized Search Heuristics," in *Analyzing Evolutionary Algorithms: The Computer Science Perspective*, pp. 7–29, Springer Berlin, Heidelberg, Germany, 2013.

[79] T. Zhang and B. Yu, "Boosting with Early Stopping: Convergence and Consistency," *The Annals of Statistics*, vol. 33, pp. 1538–1579, Aug. 2005.

[80] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, pp. 1–52, Apr. 2018.

[81] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. The MIT Press, Cambridge, MA, United States, 1992.

[82] K. Deb, "An introduction to genetic algorithms," *Sadhana*, vol. 24, pp. 293–315, Aug. 1999.

[83] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II," in *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, (September 18-20, 2000, Paris, France), pp. 849–858, Springer Berlin, Heidelberg, Germany, Sept. 2000.

[84] N. Srinivas and K. Deb, "Muiltiobjective Optimization Using Non-dominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, pp. 221–248, Sept. 1994.

[85] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley Interscience Series in Systems and Optimization, John Wiley & Sons Ltd., Chichester, UK, 2001.

[86] K. Deb, "Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction," in *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, pp. 3–34, Springer London, UK, 2011.

[87] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine Learning Interpretability: A Survey on Methods and Metrics," *Electronics*, vol. 8, July 2019.

[88] D. Doran, S. Schulz, and T. R. Besold, "What Does Explainable AI Really Mean? A New Conceptualization of Perspectives," *arXiv, arXiv:1710.00794*, Oct. 2017.

[89] Python Software Foundation, "The Python Language Reference." `https://docs.python.org/3/reference/`. [Online; accessed: 01-07-2021].

[90] C. Baldwin and K. Clark, *Design Rules Volume I: The Power of Modularity*. The MIT Press, Cambridge, MA, United States, 2000.

[91] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and Analysis of Computer Experiments," *Statistic Science*, vol. 4, pp. 409 – 423, Nov. 1989.

[92] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley Series in Probability and Statistics, John Wiley & Sons Inc., Hoboken, NJ, United States, 3. ed., 2009.

[93] A. Giunta, S. Wojtkiewicz, and M. Eldred, "Overview of Modern Design of Experiments Methods for Computational Simulations (Invited)," in *41st Aerospace Sciences Meeting and Exhibit*, (January 06-09, 2003, Reno, Nevada), American Institute of Aeronautics and Astronautics, Jan. 2003.

[94] I. Sobol', "On the distribution of points in a cube and the approximate evaluation of integrals," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, pp. 86–112, Jan. 1967.

[95] S. Burhenne, D. Jacob, and G. P. Henze, "Sampling based on Sobol' sequences for Monte Carlo techniques applied to building simulations," in *Proceedings of Building Simulation 2011*, (November 14-16, 2011, Sydney, Australia), pp. 1816–1823, International Building Performance Simulation Association, Nov. 2011.

[96] T. Gawlica, W. Samenfink, E. Schünemann, and T. Koch, "Model-based optimization of multi-hole injector spray targeting for gasoline direct injection," in *11. Tagung Einspritzung und Kraftstoffe 2018*, (November 28-29, 2018, Berlin, Germany), pp. 271–289, Springer Fachmedien, Wiesbaden, Germany, Dec. 2018.

[97] W. H. Inmon, *Building the Data Warehouse*. John Wiley & Sons Inc., New York, NY, United States, 3. ed., 2002.

[98] S. Koranne, "Hierarchical Data Format 5 : HDF5," in *Handbook of Open Source Tools*, pp. 191–200, Springer, Boston, MA, United States, 2011.

[99] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in Neurorobotics*, vol. 7, Dec. 2013.

[100] J. Rubin, C. Potes, M. Xu-Wilson, J. Dong, A. Rahman, H. Nguyen, and D. Moromisato, "An ensemble boosting model for predicting transfer to the pediatric intensive care unit," *International Journal of Medical Informatics*, vol. 112, pp. 15–20, Apr. 2018.

[101] Y. Xia, C. Liu, Y. Li, and N. Liu, "A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring," *Expert Systems with Applications*, vol. 78, pp. 225–241, July 2017.

[102] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS 11, (December 12-15, 2011, Granada, Spain), pp. 2546–2554, Curran Associates Inc., Red Hook, NY, United States, Dec. 2011.

[103] L. Laugier, D. Bash, J. Recatala, H. K. Ng, S. Ramasamy, C.-S. Foo, V. R. Chandrasekhar, and K. Hippalgaonkar, "Predicting thermoelectric properties from crystal graphs and material descriptors - first application for functional materials," *arXiv, arXiv:1811.06219v1*, Nov. 2018.

[104] T. Fonseca, Y. Gorodetskaya, G. Tavares, C. Ribeiro, and L. Goliatt, "A Gradient Boosting Model Optimized by a Genetic Algorithm for Short-term Riverflow Forecast," *Revista Mundi Engenharia, Tecnologia e Gestão*, vol. 4, May 2019.

[105] S. Deng, C. Wang, J. Li, H. Yu, H. Tian, Y. Zhang, Y. Cui, F. Ma, and T. Yang, "Identification of Insider Trading Using Extreme Gradient Boosting and Multi-Objective Optimization," *Information*, vol. 10, Nov. 2019.

[106] Dask Development Team, "Dask : Library for dynamic task scheduling." `https://dask.pydata.org/`, 2016. [Online; accessed: 01-07-2021].

[107] F.-A. Fortin, F.-M. D. Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary Algorithms Made Easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, July 2012.

[108] XGBoost developers, "XGBoost documentation." `https://xgboost.readthedocs.io/`, 2021. [Online; accessed: 01-07-2021].

[109] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, pp. 79–82, Dec. 2005.

[110] J. H. Friedman, "Multivariate Adaptive Regression Splines," *The Annals of Statistics*, vol. 19, pp. 1–67, Mar. 1991.

[111] D. Vrajitoru, "Large Population or Many Generations for Genetic Algorithms? Implications in Information Retrieval," in *Soft Computing in Information Retrieval*, vol. 50 of *Studies in Fuzziness and Soft Computing*, pp. 199–222, Physica-Verlag HD, Heidelberg, Germany, 2000.

[112] O. Roeva, S. Fidanova, and M. Paprzycki, "Influence of the population size on the genetic algorithm performance in case of cultivation process modelling," in *2013 Federated Conference on Computer Science and Information Systems*, FedCSIS 2013, (September 08-11, 2013, Krakow, Poland), pp. 371–376, IEEE, Piscataway, NJ, United States, Nov. 2013.

[113] K. Kalaiselvi and A. Kumar, "An empirical study on effect of variations in the population size and generations of genetic algorithms in cryptography," in *2017 IEEE International Conference on Current Trends in Advanced Computing*, ICCTAC, (March 02-03, 2017, Bangalore, India), IEEE, Piscataway, NJ, United States, Jan. 2018.

[114] R. Hellmann, *Geometrieoptimierung von Benzin-Hochdruck-Einspritzventilen mit Hilfe numerischer Strömungsmechanik und genetischer Algorithmen*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Fakultät für Verfahrens- und Systemtechnik, 2021.

[115] T. Gawlica, M. Lippisch, W. Samenfink, E. Schünemann, and T. Koch, "Untersuchung zum Einfluss des Spray Targetings von Mehrloch-Injektoren für Benzin-Direkteinspritzung auf das Kennfeldverhalten von Spray und Brennverfahren," in *10. Tagung Diesel- und Benzindirekteinspritzung 2016*, (November 24-25, 2016, Berlin, Germany), pp. 325–343, Springer Fachmedien, Wiesbaden, Germany, Jan. 2017.

[116] M. O. Paschke, *Entwicklung einer Methode zum softwaregestützten Abgleich eines CFD-Spraymodells mit High-Speed-Sprayaufnahmen*. Master Thesis, Universität Stuttgart Institut für Thermodynamik der Luft- und Raumfahrt (ITLR), 2017.

[117] P. Leick, M. Schmitt, T. Gawlica, T. Kubis, and K. Stapf, "Impact of flash-boiling on gasoline sprays: From fundamental physical insights to engine-measured PN emissions," in *13th International AVL Symposium on Propulsion Diagnostics*, (June 26, 2018, Baden-Baden, Germany), AVL, Graz, Austria, June 2018.

[118] T. Hu, H. Teng, X. Luo, and B. Chen, "Impact of Fuel Injection on Dilution of Engine Crankcase Oil for Turbocharged Gasoline Direct-Injection Engines," *SAE International Journal of Engines*, vol. 8, pp. 1107–1116, Apr. 2015.

[119] S. Desai, T. M. Nguyen, and J. Chen, "Fuel wall film effects on flame quenching and emissions in Gasoline Direct Injection (GDI) engines," in *73rd Annual Meeting of the APS Division of Fluid Dynamics*, (November 22-24, 2020, Virtual Event), American Physical Society, Nov. 2020.

[120] B. Settles, *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Springer Cham, Switzerland, 2012.

Own Publications and Conferences

Publications

- M. Botticelli; R. Hellmann; P. Jochmann; K. G. Stapf; E. Schünemann. *Application of Machine Learning to Gasoline Direct Injection Systems: Towards a Data-Driven Development*. In 19th IEEE International Conference on Machine Learning and Applications (ICMLA). Miami, USA. IEEE, 2020, pages 805-810. DOI: 10.1109/ICMLA51294.2020.00131.
- M. Botticelli; R. Hellmann; P. Jochmann; K. G. Stapf; E. Schünemann. *Model Selection for Gasoline Direct Injection Characteristics Using Boosting and Genetic Algorithms*. In 2021 International Symposium on Electrical, Electronics and Information Engineering (ISEEIE). Seoul, Republic of Korea. ACM, 2021, pages 241-248. DOI: 10.1145/3459104.3459145.
- M. Botticelli; R. Hellmann; P. Jochmann; K. G. Stapf; E. Schünemann. *AI-Driven Gasoline Direct Injection Development: A Knowledge-Discovery Framework for Comprehensible Evaluations of Complex Physical Phenomena*. In Internationaler Motorenkongress 2021. Baden-Baden, Germany. Springer Vieweg, 2021, pages 469-484. DOI: 10.1007/978-3-658-35588-3_27.

Conferences

- M. Botticelli. *AI-Based Knowledge-Discovery in GDI Combustion Systems*. In 2021 Bosch Artificial Intelligence Conference (AICON). Renningen, Germany. Bosch, 2021.
- M. Botticelli; R. Hellmann; P. Jochmann; K. G. Stapf. *Explore and Exploit Design Spaces based on AI-Driven Virtual Development*. In AVL International Simulation Conference 2021. Graz, Austria. AVL, 2021.