Towards Efficient and Explainable Automated Machine Learning Pipelines Design : Application to Industry 4.0 Data

Moncef Garouani

Doctoral thesis

Academic Field : Informatique and applications
Speciality : Sciences et Technologies de l'Information et de la Communication

Presented at *Ecole Doctorale en Sciences Technologie et Santé (ED 585)* **of**
Université du Littoral Côte d'Opale
and
at Ecole doctorale Sciences et Applications
of Université Hassan II de Casablanca

By

Moncef GAROUANI

In order to become Doctor from Université du Littoral Côte d'Opale

Towards Efficient and Explainable Automated Machine Learning Pipelines Design

Application to Industry 4.0 Data

Defended on September 27, 2022, after the approval of the reviewers, in front of the examining board:

Noura Yousfi, Professor, Université Hassan II de Casablanca	**President**
Pierre Parrend, Professor, EPITA Strasbourg	**Reviewer**
Abdellah Azmani, Professor, Université Abdelmalek Essaadi	**Reviewer**
Sébastien Verel, Professor, Université du Littoral Côte d'Opale	**Examiner**
Aicha Majda, Professor, Université Moulay Ismail de Meknès	**Examiner**
Sebastian Ventura, Professor, Université de Cordoue	**Examiner**
Mourad Bouneffa, Associate Professor HDR, ULCO	**Thesis supervisor**
Mohamed Hamlich, Professor, Université Hassan II de Casablanca	**Thesis supervisor**

Colophon

Doctoral dissertation entitled “Towards Efficient and Explainable Automated Machine Learning Pipelines Design”, written by Moncef Garouani, completed on October 13, 2022, typeset with the document preparation system LaTeX and the yathesis class dedicated to theses prepared in France.

Doctoral thesis

Academic Field : Informatique and applications
Speciality : Sciences et Technologies de l'Information et de la Communication

Presented at *Ecole Doctorale en Sciences Technologie et Santé (ED 585)* **of**
Université du Littoral Côte d'Opale
and
at Ecole doctorale Sciences et Applications
of Université Hassan II de Casablanca

By

Moncef GAROUANI

In order to become Doctor from Université du Littoral Côte d'Opale

Towards Efficient and Explainable Automated Machine Learning Pipelines Design

Application to Industry 4.0 Data

Defended on September 27, 2022, after the approval of the reviewers, in front of the examining board:

Noura Yousfi, Professor, Université Hassan II de Casablanca	**President**
Pierre Parrend, Professor, EPITA Strasbourg	**Reviewer**
Abdellah Azmani, Professor, Université Abdelmalek Essaadi	**Reviewer**
Sébastien Verel, Professor, Université du Littoral Côte d'Opale	**Examiner**
Aicha Majda, Professor, Université Moulay Ismail de Meknès	**Examiner**
Sebastian Ventura, Professor, Université de Cordoue	**Examiner**
Mourad Bouneffa, Associate Professor HDR, ULCO	**Thesis supervisor**
Mohamed Hamlich, Professor, Université Hassan II de Casablanca	**Thesis supervisor**

Thèse de Doctorat

Mention : Informatique et applications
Spécialité : Sciences et Technologies de l'Information et de la Communication

présentée à *l'Ecole Doctorale en Sciences Technologie et Santé (ED 585)* **de**
l'Université du Littoral Côte d'Opale
et
à l'Ecole doctorale Sciences et Applications
de l'Université Hassan II de Casablanca

par

Moncef GAROUANI

pour obtenir le grade de Docteur de l'Université du Littoral Côte d'Opale

Vers une automatisation efficace et explicable des processus d'apprentissage automatique

Application à l'Industrie 4.0

Soutenue le 27 Septembre 2022, après avis des rapporteurs, devant le jury d'examen :

Noura Yousfi, Professeur, Université Hassan II de Casablanca	**Président**
Pierre Parrend, Professeur, EPITA Strasbourg	**Rapporteur**
Abdellah Azmani, Professeur, Université Abdelmalek Essaadi	**Rapporteur**
Sébastien Verel, Professeur, Université du Littoral Côte d'Opale	**Examinateur**
Aicha Majda, Professeur, Université Moulay Ismail de Meknès	**Examinateur**
Sebastian Ventura, Professeur, Université de Cordoue	**Examinateur**
Mourad Bouneffa, Maître de Conférences HDR, ULCO	**Directeur de thèse**
Mohamed Hamlich, Professeur, UH2C	**Directeur de thèse**

Keywords: data analysis, machine learning, automl, explainable ai

Mots clés : analyse de données, apprentissage automatique, automl, explicabilité de l'ia

This thesis has been prepared at the following research units.

Laboratoire d'Informatique Signal et Image de la Côte d'Opale

Maison de la Recherche Blaise Pascal
50, rue Ferdinand Buisson
BP 719
62228 Calais Cedex
France

☎ (33)(0)3 21 46 36 53
🖷 (33)(0)3 21 46 55 75
✉ secretariat@lisic.univ-littoral.fr
Web Site https://www-lisic.univ-littoral.fr/

Complex Cyber Physical System Laboratory

150 Avenue Nile Sidi Othman
20670 Casablanca
Maroc

☎ (212)(0)5 22 56 42 22
✉ contact@ensam-casa.ma
Web Site http://ensam-casa.ma/

Study and Research Center for Engineering and Management

293 Boulevard Ghandi
20410 Casablanca
Maroc

☎ (212)(0)5 22 34 17 23
✉ contact@hestim.ma
Web Site http://www.hestim.ma/

Towards Efficient and Explainable Automated Machine Learning Pipelines Design
Application to Industry 4.0 Data

Abstract

Machine learning (ML) has penetrated all aspects of the modern life, and brought more convenience and satisfaction for variables of interest. However, building such solutions is a time consuming and challenging process that requires highly technical expertise. This certainly engages many more people, not necessarily experts, to perform analytics tasks. While the selection and the parametrization of ML models require tedious episodes of trial and error. Additionally, domain experts often lack the expertise to apply advanced analytics. Consequently, they intend frequent consultations with data scientists. However, these collaborations often result in increased costs in terms of undesired delays. It thus can lead risks such as human-resource bottlenecks. Subsequently, as the tasks become more complex, similarly the more support solutions are needed for the increased ML usability for the non-ML masters. To that end, Automated ML (AutoML) is a data-mining formalism with the aim of reducing human effort and readily improving the development cycle through automation.

The field of AutoML aims to make these decisions in a data-driven, objective, and automated way. Thereby, AutoML makes ML techniques accessible to domain scientists who are interested in applying advanced analytics but lack the required expertise. This can be seen as a democratization of ML. AutoML is usually treated as an algorithms selection and parametrization problem. In this regard, existing approaches include Bayesian optimization, evolutionary algorithms as well as reinforcement learning. These approaches have focused on providing user assistance by automating parts or the entire data analysis process, but without being concerned on its impact on the analysis. The goal has generally been focused on the performance factors, thus leaving aside other important and even crucial aspects such as computational complexity, confidence and transparency. In contrast, this thesis aims at developing alternative methods that provide assistance in building appropriate modeling techniques while providing the rationale for the selected models. In particular, we consider this important demand in intelligent assistance as a meta-analysis process, and we make progress towards addressing two challenges in AutoML research. First, to overcome the computational complexity problem, we studied a formulation of AutoML as a recommendation problem, and proposed a new conceptualization of a Meta-Learning (MtL)-based expert system capable of recommending optimal ML pipelines for a given task; Second, we investigated the automatic explainability aspect of the AutoML process to address the problem of the acceptance of, and the trust in such black-boxes support systems.

To this end, we have designed and implemented a framework architecture that leverages ideas from MtL to learn the relationship between a new set of datasets meta-data and mining algorithms. This eventually enables recommending ML pipelines according to their potential impact on the analysis. To guide the development of our work, we chose to focus on the Industry 4.0 as a main field of application for all the constraints it offers. Finally, in this doctoral thesis, we focus on the user assistance in the algorithms selection and tuning step. We devise an architecture and build a tool, `AMLBID`, that provides users support with the aim of improving the analysis and decreasing the amount of time spent in algorithms selection and parametrization. It is a tool that for the first time does not aim at providing data analysis support only, but instead, it is oriented towards positively contributing to the trust-in such powerful support systems by automatically providing a set of explanation levels to inspect the provided results.

Keywords: data analysis, machine learning, automl, explainable ai

Vers une automatisation efficace et explicable des processus d'apprentissage automatique
Application à l'Industrie 4.0

Résumé

L'industrie du futur introduit de nouveaux concepts, processus et pratiques conduisant à des mutations profondes dans le pilotage des systèmes d'information associés. Une des problématiques cruciales est l'utilisation de la quantité importante de données, notamment celles produites par les différents dispositifs d'acquisition de données (Cyber Physical Systems, etc.), pour en extraire de la connaissance destinée à la maîtrise des processus de l'entreprise à travers un système d'information évolutif, réactif et adapté aux spécificités de l'industrie 4.0.

L'intelligence artificielle et plus particulièrement l'apprentissage automatique fournit les algorithmes, méthodes et outils permettant l'extraction de connaissances et de modèles à partir des données représentant l'activité d'une entreprise et son environnement, et l'apport de plus d'automatisation des processus sous-jacents. Cependant, de nombreuses entreprises ne disposent pas de moyens humains leur permettant de déployer efficacement des solutions d'apprentissage automatique. Cela s'explique notamment par le fait que la construction de telles solutions est un processus long et difficile qui nécessite une expertise hautement technique et intersectorielle et qui est une ressource limitée. Nous nous intéressons donc à ce besoin d'assistance à l'analyse de données, qui commence à recevoir une certaine attention des communautés scientifiques, donnant naissance au domaine dit d'apprentissage automatique automatisé.

L'apprentissage automatique automatisé est devenu un domaine en plein essor qui vise à rendre l'application des méthodes d'apprentissage automatique aussi dépourvue d'intervention humaine que possible. A cet égard, les approches existantes se révèlent souvent similaires et peu abouties. Ces approches sont concentrées sur l'assistance de l'utilisateur en automatisant une partie ou l'ensemble du processus d'analyse de données, mais sans se soucier de son impact sur l'analyse. L'objectif a généralement été axé sur les facteurs de performance, laissant ainsi de côté d'autres aspects importants, voire cruciaux, tels que la complexité du calcul, la confiance et la transparence.

Cette observation nous a amenés à orienter nos recherches vers le domaine du Meta-Apprentissage (MtL) et à développer des méthodes alternatives qui apportent une aide à la construction des techniques de modélisation appropriées tout en fournissant le rationnel des modèles ML sélectionnés. En particulier, nous considérons cette demande importante d'assistance intelligente comme un processus de méta-analyse, et nous progressons vers la résolution de deux défis de la recherche en AutoML. Dans un premier temps, pour palier au problème de la complexité du calcul, nous avons étudié une formulation de l'AutoML en tant que problème de recommandation, puis proposé une nouvelle conceptualisation d'un système expert basé sur le MtL capable de recommander des pipelines ML optimaux pour une tâche donnée. Dans un second temps, nous avons traité l'explicabilité du processus d'aide à la décision de l'AutoML pour prendre en compte la problématique de l'acceptation et la confiance en ces systèmes généralement vus comme des boîtes noires.

Mots clés : analyse de données, apprentissage automatique, automl, explicabilité de l'ia

Remerciements

I would first like to express my sincere gratitude to Dr. Mourad Bouneffa and Dr. Mohamed Hamlich, my main PhD supervisors, for all their support and help during the past two years. Granting me with trust, freedom, and yet always the right level of scientific expectations, allowed me to thrive during this rich period. I am very grateful for their guidance, support and encouragement throughout the development of my thesis and daily life. Words cannot express how grateful I am to have worked alongside theme. Their constant feedback, time and dedication, inspired me to give my best every day.

I was lucky to have not just two but three advisors : professors Mourad Bouneffa, Mohamed Hamlich, and Adeel Ahmad. Thank you Dear Adeel for all the precious insights, suggestions, hard questions, and gentle pushes.

I would like to express my special gratitude to the honorable Committee members for their time and hard work to review my thesis, and for giving me valuable remarks to improve it.

LISIC provided me with not only excellent working conditions and brilliant supervisors, but also a supportive community of fellow researchers and students. More generally, I would like to thank the members of the LISIC laboratory, with special thanks to the Knowledge Engineering team (IC), for providing a rich and stimulating research environment. I'm also grateful for everyone I had the pleasure to work with at ULCO Calais, an activity I've deeply enjoyed, and that allowed me to get me out of my research from time to time. A big thank you to the administrative team for always being helpful.

To my parents and life coaches, no words can express my feelings of gratitude. Thank you for being there every moment, showering me with love, support, encouragement, sincere prayers, and care. I am extremely grateful for all the sacrifices you have done for making my future brighter. To my brothers : Mohamed, and Chakib, thank you for all of the sacrifices that you have made on my behalf. I love you.

Special thanks to the University of the Littoral Opal Coast (ULCO), HESTIM Engineering and Business School, and the National Center for Scientific and Technical Research (CNRST) for supporting this work with a scholarship grant.

To you all, my most sincere thanks.

Table of Contents

List of Tables

List of Figures

Acronyms

A | B | C | D | G | H | I | K | L | M | N | R | S | X

A

Acc Accuracy.
AE Auto-encoders.
AI Artificial intelligence.
AMLBID Automated Machine Learning tool for Big Industrial Data.
ANNs Artificial neural networks.
ASP Algorithm Selection Problem.
AUC Area Under Curve.
AutoML Automated Machine Learning.

B

BID Big Industrial Data.
BO Bayesian Optimization.

C

CASH Combined Hyperparameters Selection and Optimization.
CV Cross-Validation.

D

DA Data Analysis.
DM Data Mining.
DSS Decision Support System.
DT Decision Tree.

G

GA Genetic Algorithms.
GS Grid Search.

H

HPO Hyperparameters Ooptimization.
HPs Hyperparameters.

I

I4.0 Industry 4.0.

K

KB Knowledge Base.
KNN k-Nearest Neighbors.

L

LOO-CV Leave One Out Cross-Validation.
LR Logistic Regression.

M

ML Machine Learning.
MtL Meta-Learning.

N

NNS Neural Networks.

R

RF Random Forest.
RS Random Search.

S

SMBO Sequential Model-based Optimization.
SVM Support Vector Machines.

X

XAI Explainable Artificial Intelligence.
XAutoML Explainable Automated Machine Learning.

Symbols

$\mathcal{A}$ Machine learning algorithms space

$\mathcal{A}^{(i)*}$ Tuned version of an algorithm$\in \mathcal{A}$

$\mathcal{D}$ Dataset

E_i A predicive performace measure (e.g. Accuracy, Recall, AUC)

$\mathcal{H}$ Hyperparameters space

$\mathcal{L}$ Loss function

l_i^n Number of neurons in the i^{th} layer

m Meta-features vector

$\mathcal{P}$ Problems or tasks space

$\mathcal{T}$ Task (e.i. problem, dataset)

Introduction

Outline of the current chapter

Background and motivation

The rapid growth of data in terms of volume, variety and velocity has led to a great development of ML tools, methods and large-scale models evidenced by successes in robotics [1], healthcare [2], and autonomous driving [3]. However, in general, capability of analyzing data lags far behind the capability of collecting it. This is due to the fact that data analytics consists of several challenging and time consuming steps, which have been grouped into the following [4]: *data selection*, *data pre-processing*, *data mining*, *evaluation and interpretation*(cf. Figure 1).

Briefly, *data selection* represents the task of sifting out the data that may not be relevant for the analysis. *Data pre-processing* represents the task of cleaning/wrangling the data, such that it will be ready for the analysis. Next, *data mining* is the broad task of applying a ML/statistical modeling algorithm on top of the pre-processed data (e.g., supervised learning, unsupervised learning). Finally, *interpretation* is the task of interpreting the predictive results.

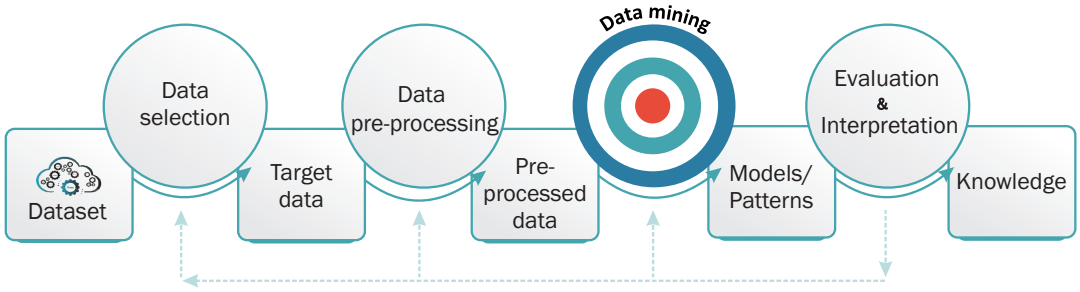

Figure 1: Data analytics process, sometimes synonymously referred to as knowledge discovery; adapted from [4]

Usually, most of computational time and resources are spent on the *Data mining* step (selecting and tuning the algorithm(s) that can deliver the optimal performance), while data scientists spend up to 80% of their time on setting up the ML pipeline [5, 6].

Progress in this area allowed a large adoption of ML solutions by the industry. Building ML solutions often involve several tasks, which include comparing many algorithms, optimizing their hyperparameters (HPs), and exploring different features representations. However, recent trends towards larger models and search spaces have drastically increased the computational cost of hyperparameters optimization. For example, training a simple state-of-the-art neural network translation architecture can take days to weeks [7] and searching for quality image classification architectures can require evaluating over 20k architectures from a search space for neural architectures search [8]. For these types of large-scale problems, trial and error strategies like grid search and random search that allocate a uniform training resource to every configuration are prohibitively slow and expensive. The long wait before receiving a feedback on the performance of a well-tuned model limits users productivity, while the high cost limits the broad accessibility of these solutions. Consequently, faster and more cost efficient ML model selection and parametrization methods are necessary for modern machine learning methods.

Multiple previous research efforts have tried to tackle the problem of users (novices and experts) support by automating parts-to-whole data analysis pipeline [9–12]— which is also the focus of this thesis. In this regard, existing

approaches include Bayesian optimization, evolutionary algorithms as well as reinforcement learning have been proposed to select, rank, or predict the best HPs settings of ML algorithms. Recentlly, Meta-Learning (MtL) [13] has emerged as a promising paradigm to automate the algorithms selection and parametrization process. The use of MtL by itself or in combination with optimization techniques tends to be more computationally efficient when compared with the use of only optimization techniques [13].

The use of Automated Machine Learning may relieve data scientists from the repetitive and time-consuming steps in the design of a full data science solution, including the data preparation, algorithm design and optimization tasks. Hence, they can allocate their time on more significant and probably difficult tasks. Thus, the goal of AutoML is to facilitate and increase the use of data science techniques by non-experts and to support data scientists in their work, not to replace them [12]. In this thesis, AutoML is considered for predictive tasks, in particular, supervised classification tasks using the manufacturing industry as a train and validation environment. Nevertheless, the issues investigated in this thesis can be easily extended to other domains and tasks.

Although the plethora of state-of-the art techniques and tools that can be used to achieve the automatic algorithms selection and parametrization, a majority still lack in their ability to effectively address the major challenges of big data on the fly. For example, some of the existing AutoML tools are not effective and scalable enough to handle real world problems (large data streams) [14]. A highly acceptable AutoML method or tool should be able to address the three major challenges of big data and should be flexible enough to adapt to changes within the organization. Secondly, most of the open source systems require little programming and/or data-science skills. When the user has no experience in programming, cloud-based paid systems should be chosen, since they offer an easy to use interface. Another limitation of many AutoML systems is the huge time and tedious process spent in finding the best ML algorithm to use on multivarying datasets, which is not always available. Lastly, another limitation of the most, if not all the AutoML systems is that they are often designed to focus the performance factors, thus leaving aside other important or even crucial aspects such as computational complexity, confidence and transparency.

The absence of explanations for predicted performing factors makes these decision support systems (DSS) usually black boxes, allowing the only prominent exhibition of input and output parameters but concealed visibility of inherent associations among them. It is more preferably desired to avoid such lack of transparency in real-time/ critical applications such that in clinical and industrial manufacturing processes. Since, these systems may imply critical decision choices; it is necessary to have some justifications of individual predictions which are perceived trough an AI algorithm, more particularly, in an automated environment. Therefore, the acceptance of, and the trust in, an AutoML DSS is highly dependent on the transparency of the recommendations.

This thesis addresses these requirements with the design and the development of a novel meta-learning based decision support and expert system focused on the automatic selection and parametrization of ML models. The major goal is to achieve an optimal performance for a given task while providing the rationale traceability behind a recommendation or decision. The designed system is particularly aimed at the provision of explanations of such rational traceability and promising trend analysis of the area of industrial big data.

Research focus and values

In effectively designing an ML system, the first step after defining the achievable goal, usually entails the process of deciding which ML approach or model to select. However, since there is no clear and strictly defined "recipes" for building the optimal ML pipeline for the problem at hand, many attempts at AutoML systems both in academia and industry have been proposed [15]. Although some AutoML systems (e.g. TPOT [11], AutoWEKA [10] and Auto-sklearn [12]) discussed in the section 2.1.4 of this thesis, are efficient in their own ways for model selection, they still far from being widely adopted due to the computational complexity and expertise requirements [6]. Some limitations other than those mentioned before they still include : complexity (much time and resources needed to run), generalizability (data type requirements), interoperability (hardware and software requirements), and explainability (black-box solutions that lack the effective explanations of the predicted performance factors).

These limitations among others, form part of the problems and motivations for undergoing this thesis. The main objective of this thesis is twofold : first, investigate and develop an efficient AutoML-based decision support and expert system able to recommend the best ML pipeline for a given problem and task. This objective is accomplished by proposing a MtL-based recommender system able to recommend the optimal or near-optimal ML pipelines (algorithms with related hyperparameters configuration) according to a desired predictive metric (e.g. Accuracy, Recall, F1-score). The second objective addresses the trust in such black-boxes decision support systems by the conceptualization and development of a transparent and self-explainable AutoML system for recommending the most adequate ML configuration for a given problem, and explain the rationale traceability behind a recommendation. It may further allow to analyze the predictive results in an interpretable and reliable manner.

As a field of application, we chose the Industry 4.0 (I4.0), and in particular, the manufacturing industry. The goal is not to develop solutions specifically and solely for the manufacturing industry domain but to use this domain to guide our choices by its specific constraints and difficulties. Indeed, the I4.0 environment is a messy concept that intrinsically poses a certain number of difficulties to analyze : grey areas of interpretation, many exceptions, non-stationarity, deductive and inductive reasoning, non-classical logic, various types of data. Statistical models often act as a black-box which is redhibitory for practical applications. In other words, the I4.0 domain combines some of the most challenging elements of today's machine learning. Therefore, by imposing ourselves the constraints of this specific field, we hope to design better AutoML decision support systems.

Main contributions to the research area

This thesis deals with several aspects of the algorithms selection and HPs tuning problems, and makes contributions to ML, MtL and AutoML. The main contributions of this thesis can be summarized as follows.

Contributions to ML and AutoML :

Motivated by the trend towards ever more expensive models, larger search spaces and limits of existing tools, our first research challenge was specifically

related to the problem of user support in the data analysis process. To be more precise, the challenge was about defining new efficient, interoperable and easy to use methods for providing user assistance in data analysis. Therefore, our response was to conceptualize and develop a framework system with such an aim. To this end, *we proposed a framework system that leveraging ideas from meta-learning able to provide support with the aim of improving the analysis and decreasing the amount of time spent in algorithms selection and parametrization. It is a tool that for the first time does not aim at providing data analysis support only for the sake of algorithms selection and parametrization, but instead, it is oriented towards positively contributing to the trust-in such powerful DSS by automatically providing a set of explanation levels to inspect the provided results without having to depend on a data scientist to generate and interpret all the extreme plots and tables.*

We implemented a prototype of the proposed framework, `AMLBID`, on a Client-Server architecture, where the server coordinates as the AutoML support system, that given a problem (dataset), a desired predictive metric (i.e., Accuracy, Recall, F1-score) recommends ML algorithms with related hyperparameters configuration that are ranked according to their impact on the final result of the analysis, while the client-side is composed of a user-friendly graphical toolbox that facilitates datasets entry, support visual simulation of various ML scenarios, and facilitate the interpretation of the obtained results. Meanwhile we implement a rule-based module that guides end-users, in case of the unsatisfying results returned by the AutoML, intended to improve the predictive performances. Thence, it may increase the transparency, controllability, and the trust-in AutoML.

Contributions to MtL :

Meta-learning is an alternative approach for addressing the algorithms selection problem (ASP). One aim of meta-learning is to assist the identification of the most appropriate learning algorithm(s) for the problem at hand by mapping datasets characteristics to the predicted data mining performance. To this end, meta-learning systems use a set of data characteristics, called meta-features, to represent and characterize data mining tasks, and search to identify the correlations between these attributes and the performance of learning algorithms [13]. The proper identification of data properties is essential to map tasks to learning

mechanisms. As a data-driven approach, the effectiveness of meta-learning is largely dependent on the tasks description (i.e., meta-features). Meta-learning requires meta-features that represent the primary learning tasks or datasets to transfer knowledge across them.

As a response to this challenge, *we assessed the currently available approaches and methods with respect to meta-features used as input to quantify the tasks similarity in the meta-learning process. We thoroughly analyzed the effectiveness of different sets of meta-features, and we performed a comprehensive metadata classification, identifying important aspects that was overlooked by the current approaches. We proposed a novel neural network-based meta-model architecture that learns intrinsic meta-features from families of traditional ones. The goal is to produce new meaningful and more informational meta-features of higher quality from the initial data characteristics.*

The empirical evaluation of the proposed meta-model has shown that it can provide successful suggestions as to which learning algorithm and HPs configuration is more appropriate for a specific dataset. Furthermore, the meta-learing models constructed by the inducers applied on the meta-learning problems will allow us to have better understanding of these dataset characteristics that affect the performance of the learning algorithms. Thus improve the effectiveness of meta-learning and open up new directions of future works in which they are applied to help solve similar problems presented by other traditional meta-models.

Reproducibility of experiments :

AMLBID is implemented as an open source Python-package to reproduce experiments, analyses, and allow further analysis. While `AMLBID` is still in its initial stages, the package was downloaded more than **17893**[1] times on PyPI[2](excluding mirrors thereof) in its first year. Feedback from the community is highly positive, and several new applications have been proposed in addition to multiple *industrial* requests.

[1] `https://pypistats.org/packages/amlbid`
[2] `https://pypi.org/project/AMLBID/`

Publications

The content of this doctoral thesis is based on the following publications :

Journals

- Moncef Garouani *et al*. "Towards big industrial data mining through explainable automated machine learning". In: *The International Journal of Advanced Manufacturing Technology* (2022). doi :10.1007/s00170-022-08761-9

- Moncef Garouani *et al*. "AMLBID: An auto-explained Automated Machine Learning tool for Big Industrial Data". In: *SoftwareX* 17 (2022). doi :10.1016/j.softx.2021.100919

- Moncef Garouani *et al*. "Using meta-learning for automated algorithms selection and configuration: an experimental framework for big industrial data". *Journal of Big Data* 9, 57 (2022). doi :10.1186/s40537-022-00612-4

- Moncef Garouani *et al*. "Autoencoder-kNN meta-model based data characterization approach for an automated selection of AI algorithms". [submitted to *Journal of Big Data*]

- Moncef Garouani *et al*. "AMLBID2.0: An auto-explained Automated Machine Learning tool for Big Industrial Data". [submitted to *SoftwareX*]

International Conferences

- Moncef Garouani *et al*. "Towards the Automation of Industrial Data Science: A Meta-Learning Based Approach". In: *23rd International Conference on Enterprise Information Systems*. 2021, pp. 709–716.doi :10.5220/0010457107090716

- Moncef Garouani *et al*. "Towards meta-learning based data analytics to better assist the domain experts in industry 4.0". In: *Lecture Notes on Data Engineering and Communications Technologies* (ICABDE'21). Springer, Cham. doi :10.1007/978-3-030-97610-1_22

- Moncef Garouani *et al*. "Towards an Automatic Assistance Framework for the Selection and Configuration of Machine-Learning-Based Data Analytics Solutions in Industry 4.0". In: *The Fifth International Conference on Big Data and Internet of Things* (BDIoT'21). [**In press**]

Posters

- Moncef Garouani *et al*. "Towards industrial data science through explainable automated machine learning". POSTER In *MTE Pole's Doctoral Day*(2021), ULCO University, Calais, France
- Moncef Garouani *et al*. "Towards explainable Automated Machine Learning". POSTER In *IA² – Institut d'Automne en Intelligence Artificielle* (2021), Sorbonne Université, Paris, France

Thesis structure

This thesis is organized in *two parts*. The **first part** provides a review of the state of the art and relevant background in the various research fields covered by the thesis in order to contextualize our work. The **Chapter 1** presents a detailed literature review on the machine learning algorithms selection and HPs parametrization principles and tools in the ML research community and also present detailed discussions on the industrial big data analysis. In **Chapter 2**, we define the context of the thesis where we will introduce the problem of supporting users in the modeling of their data mining processes. We present the state-of-the-art approaches, which fall broadly into two main categories : ontology-based DM workflow planning systems and meta-learning. We discuss how meta-learning formalism can be used to help resolving the algorithms selection and the parametrization problem. Finally, we present the limitations of the current approaches and tools from which we defined the starting points of the thesis.

The **second part** shows our contributions. The **Chapter 3** describes the proposed meta-learning based AutoML system design, architecture, components, and characteristics. It also defines all methods used in this thesis and presents

a detailed discussion of all pre-design experimentations including the setup, algorithms considered, problems identified, and knowledge gained during the experiments. Finally, it provides a comparative study to some of the current state of the art AutoML related works and tools. The identified problems and knowledge gained in this chapter, served as the basis for the design and modelling in the next chapters. In **Chapter 4**, we review the existing works in characterizing datasets; assess the currently available approaches and methods with respect to meta-features used as input to quantify the tasks similarity in the meta-learning process, and identify the important aspects that was overlooked by the current works and tools. Furthermore, we establish the set of intrinsic meta-features that will be used to describe the datasets, and to construct the meta-learning problems. The idea of the Autoencoder-kNN based meta-model with built-in latent features extraction is presented, and various issues are discussed concerning the quality of the characteristics and the problems that they set. Finally, we assessed thoroughly the effectiveness of the proposed meta-model based data characterization approach and show that the approach significantly outperforms the state-of-the-art methods in data characterization. In **Chapter 5**, we describe the self-explainable AutoML framework, in which we used the meta-learning for defining the algorithms selection and parametrization task, and which we extend to overcome the black-box nature of such powerful support system. We will explore the combination of AMLBID with, AMLExplainer, a developed toolbox platform that makes it possible to provide features of explainability of the AutoML resulting algorithms and models. Finally, in **Chapter 6**, we present the open source AMLBID software package. A python based decision support system for automated selection and tuning of implied hyperparameters for machine learning algorithms to cope with the prominent challenges posed by the evolution of industrial big data. Furthermore, the tool is equipped with the explainer module that makes the outcomes rather transparent and interpretable for well-performing ML systems. Being based on meta-learning, the tool is able to simulate the role of the machine-learning expert as a decision support system.

Chapter 7 summarizes and concludes the work and proposes directions for further work.

Part I

State of the art

Chapter 1

ML algorithms selection and hyperparameters tunning

Outline of the current chapter

Machine learning has made significant advances with the rise of deep learning and large-scale models development evidenced by successes in robotics [1], healthcare [2], and autonomous driving [3]. However, developing such solutions is a resource intensive endeavour both in terms of computation and human expertise. Applying machine learning to real world problems is a multi-stage process

requiring significant human effort from data collection to model deployment.

In this chapter, we present the ML algorithms selection and parametrization problem, and the different approaches to solve it. In this thesis, we decided to focus on the classification problem as it is one of the most widely studied in ML due to the large amount of situations that can be modeled as such a problem. Specifically, we focus on binary and multiclass classification problems.

1.1 The algorithms selection problem

The machine learning field has been evolving for a long time and has provided us a variety of models and algorithms to solve supervised, semi-supervised and unsupervised tasks. Designing and solving a classification problem is a time intensive task consisting of many phases, that require a considerable technical knowledge generally held by experts analysts.

Once a learning problem is defined, the practitioner needs to find adequate learning tools to solve it. These tools can target different parts of the ML pipeline, i.e., *data preparation*, *features engineering*, *model selection*, and *hyperparameters tuning or optimization*. The quality of the available data is one of the most crucial factors to achieve high performance solution. However, we will not go into the details of data collection and the quality of the available data. The analyst's main task will be to select the most appropriate learning method (supervised learning/classifier or unsupervised learning/clusterer), according to some performance measures and within the constraint imposed by the application. The analyst has to select the ones that better match the morphology and the specific characteristics of the problem at hand. This selection is one of the most difficult problems since there is no model or algorithm that performs better than all others and independently of the particular problem characteristics, as it has been observed in various empirical comparisons [16].

In figure 1.1, we give an overview of the analyst process. The analyst has at his disposal a pool of learning tools, from which (s)he initially selects some ones for the evaluation on the specific problem. The initial selection can be based on knowledge of the problem, that means to select the algorithms whose characteristics better match the characteristics of the dataset, or even on the

analyst's preferences for specific learning algorithms. The evaluation usually requires an extensive experimentation, which consists in repetitive executions of the selected algorithm(s).

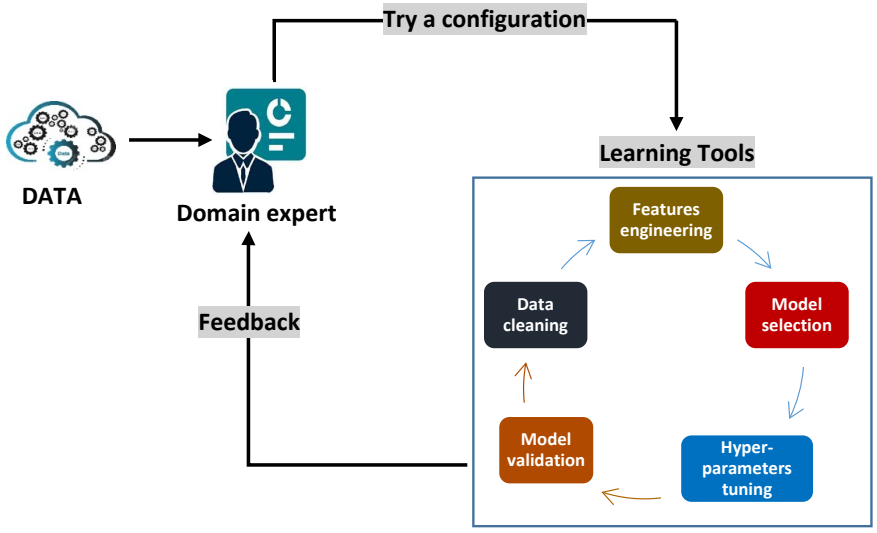

Figure 1.1: The process of ML models selection for knowledge discovery.

To obtain a good learning performance, (s)he will try to set a configuration using personal experience or intuition about the data and tools underneath. Then, based on the feedback about how the learning tools performed, the practitioner will adjust the configuration hoping that the performance shall be improved. Such a trial-and-error process terminates once a desired performance is achieved or the computational budget runs out.

1.2 Hyperparameters tuning

The selection of an algorithm or a family of algorithms that are more likely to perform better on a given combination of datasets and evaluation measures is an important task [17]. The machine learning algorithms generally have two kinds of parameters : (1) The *ordinary parameters* that the model learns and optimizes automatically based on its regular behavior during the learning phase; (2) The *hyperparameters* (categorical and continuous) which are usually manually set before beginning the training of the model (as shown in Table 1.1).

ML Algorithm	Nb. of ordinary parameters	Nb. of hyperparameters
SVM	2	5
Decision Tree	1	3
Random Forest	2	4
Logistic Regression	4	6

Table 1.1: Configuration space of some classification algorithms.

In contexts of manufacturing industry, it poses a major research challenge to select the feasible ML algorithms and tuning of hyperparameters in the context of a fresh problem. The algorithms selection and parametrization is a complex process as generally the ML algorithms are used as "black boxes". Their performance is affected by multiple characteristics of the datasets and algorithms hyperparameters [18]. Thus, the complexity of the selection and configuration of appropriate algorithm(s) is an error prone and time-consuming process due to the prevailing flaws while establishing the multiple configurations. Figure 1.2 illustrates the general hyperparameters tuning schema.

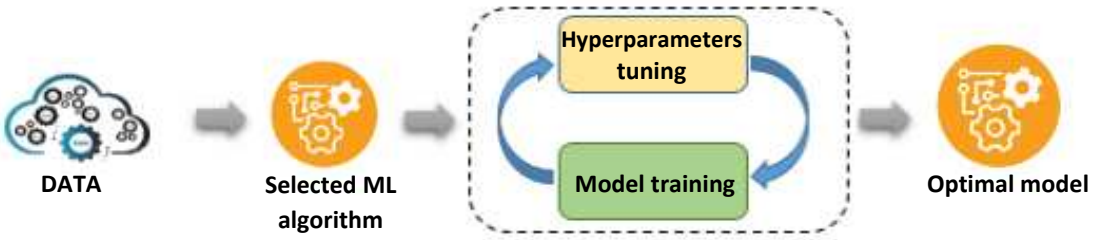

Figure 1.2: General view of the hyperparameters tuning schema.

In many situations, the HPs tuning is carried out manually by the expert, progressively *refining* a grid of values over the desired space. From a theoretical point of view, selecting the ideal HPs values requires an exhaustive search over all possible subsets of HPs. Depending on the number and types of the HPs, this task becomes impractical. Therefore, for the ML community, it is often accepted to search reduced HPs space instead of the complete space [19].

1.2.1 Definitions

The HPs tuning process is usually treated as a black-box optimization problem whose objective function is associated with the predictive performance of the model induced by a ML algorithm. Formally this can be defined as follows :

Definition 1.2.1. *Let $\mathcal{H} = \{H_1, \ldots, H_n\}$ be the HPs space of an algorithm $A^{(i)} \in \mathcal{A}$, where $\mathcal{A}$ is the set of learning algorithms space. Each $A^{(i)*} \in \mathcal{A}$ represents a tuned version of $A^{(i)}$ and can be usually defined by a set of constraints.*

Definition 1.2.2. *Let $\mathcal{D}$ be a dataset divided into disjoint training D_{train}, and validation $D_{validation}$ sets. The function $\mathcal{L} : A^{(i)} \times H_n \times D \rightarrow \mathbb{R}$ measures the predictive performance of the model induced by the algorithm $A^{(i)}$ with an hyperparameters configuration $H_n \in \mathcal{H}$ on the dataset D. Without loss of generality, higher values of $\mathcal{L}(A^{(i)}, H_n, D_{train}, D_{validation})$ mean higher predictive performance.*

Definition 1.2.3. *Given $\mathcal{A}$, $\mathcal{H}$ and $\mathcal{D}$, together with the previous definitions, the goal of the algorithm selection problem (ASP) is to find the $A^{(i)*}$ that minimizes or maximizes the $\mathcal{L}$ on $\mathcal{D}$ such that :*

$$A^{(i)*} \in \underset{A^{(i)} \in \mathcal{A}, H_n \in \mathcal{H}}{argmin} \; \mathcal{L}(A^{(i)}, H_n, D_{train}, D_{validation})$$

The combined algorithms selection and hyperparameters optimization (CASH) involves identifying the most adequat algorithm $A^{(i)} \in \mathcal{A}$ along with related hyperparameters configuration H_n from a set of possible configurations $\mathcal{H}$. The search space defines this set of configurations, and can include *continuous* or *discrete* hyperparameters in a structured or unstructured manner [19, 20]. Examples of continuous hyperparameters include learning rate and momentum for stochastic gradient descent, degree of regularization to apply to the training objective, and the scale of a kernel similarity function for kernel classification. Examples of discrete hyperparameters include choices of activation function, number of layers in a neural network, and number of trees in a

random forest. Finally, structured search spaces include those with conditional hyperparameters (e.g., the relevant hyperparameters can depend on the choice of supervised learning method) and other tree type search spaces [21].

The optimization of the HPs values can be based on any performance measure (e.g. Accuracy, AUC, Recall), which can even be defined by multi-objective criteria. Further aspects can make the tuning more difficult, like :

- HPs configurations that lead to a model with high predictive performance for a given dataset may not lead to high predictive performance for other datasets;
- HPs values often depend on each other[1]. Hence, independent tune of HPs may not lead to a good set of HPs values;
- The exhaustive evaluation of several HPs configuration can be very time-consuming.

Given a search space, there are various search methods to select putative configurations to evaluate. The most simple, and often used, are *Grid Search* (GS) and *Random Search* (RS) [22]. The former is more suitable for low dimensional problems, i.e., when there are few HPs to set. For more complex scenarios, GS is unable to explore finer promising regions due to the large hyperspace. The latter is able to explore any possible solution of the hyperspace, but also does not perform an informed search, which may lead to a high computational cost [23]. Meta-heuristics have also been used for HPs tuning, having the advantage of performing informed searches. Population-based methods, such as *Genetic Algorithms* (GAs) [24], *Estimation of Distribution Algorithms* (EDAs) [25], and *Sequential Model-based Optimization* (SMBO) [26], have been largely explored in the literature due to their probabilistic nature and faster convergence. However, SMBO itself has many HPs and does not eliminate the shortcoming of having to iteratively evaluate the function to be optimized. All these techniques are valuable alternatives to GS and RS, but they might have a high computational cost, since a large number of candidate solutions usually needs to be evaluated.

[1] This is the case of SVMs

1.2.2 Hyperparameters tuning techniques

Over the last decades, different hyperparameters tuning techniques have been applied to ML algorithms [22–25]. Some of these techniques iteratively build a population $\mathcal{H}$ of HPs settings, where $\mathcal{L}(A^{(i)}, H_n, D_{train}, D_{validation})$ is computed for each $H_n \in \mathcal{H}$. By doing so, they can simultaneously explore different regions of a search space. There are various population-based HPs tuning strategies, which differ in how they update the $\mathcal{H}$ at each iteration. Some of them are briefly described next.

Grid Search

Grid Search is the most straightforward approach to HPs tuning. It is an exhaustive search through a subset of the HPs space to select the best of a family of models, parametrized by a grid of values. Its simple execution is illustrated in Algorithm 1.

Algorithm 1 GS pseudocode.

$Best_{global} \leftarrow$ NULL
for each $H_n \in \mathcal{H}$ **do**
    Sample a set of values $V = v_{i1}, v_{i2}, \ldots, v_{in}$ from H_n
for each $\lambda \in (H_1, \ldots, H_k) = (V_1, \ldots, V_k)$ **do**
    $Best_{local} \leftarrow f(A, D, \lambda)$
$Best_{global} \leftarrow \max(Best_{local})$
    return $Best_{global}$

GS may be a good choice for spaces with few HPs. However, it suffers from the dimensionality of the problem : the higher the number of HPs evaluated, the higher the computational cost required to solve the problem. Even so, the manual selection of the grid values that precedes the search may provide some tips on how the HPs space surface behaves [19].

Random Search

Random Search is a simple technique that performs random trials in a search space. Its use can reduce the computational cost when there is a large number

of possible settings being investigated [22]. Usually, RS performs its search iteratively on a population P in a predefined number of iterations. $P(i)$ is extended (updated) by a randomly generated HPs setting $h \in \mathcal{H}$ in each (ith) iteration of the HPs tuning process. RS has obtained efficient results in optimization for Deep Learning (DL) algorithms [19]. The RS simple workflow is described in Algorithm 2.

Algorithm 2 RS pseudocode.

$t \leftarrow 1$
$Best_{global} \leftarrow$ NULL
while Stoping criteria not satisfied **do**
  Generate a population $P(t)$ randomly
  for each $p_i \in P(t)$ **do**
    $f_{p_i} \leftarrow f(A, D, p_i)$
  $Best_{local} \leftarrow \max(f_p)$
  if $Best_{local} \geq Best_{global}$ **then**
    $Best_{local} \leftarrow Best_{global}$
  $t \leftarrow t + 1$
  return $Best_{global}$

Bayesian Optimization

Both, grid and random search, are stateless optimization techniques which do not take previous evaluations into account. Bayesian optimization can be used to overcome this lack. Considering the choice of algorithm configurations as a black-box global optimization problem, BO can be used to automatically find optimal configurations.

Bayesian optimization is an adaptive hyperparametric search method that predicts the next combination that is likely to bring the most benefit based on the currently tested hyper-parametric combinations [27]. Assuming that the function $f(x)$ of hyperparameter optimization obeys the Gaussian process, then $p(f(x)|x)$ is a normal distribution. The Bayesian optimization process is modeled as a Gaussian process based on the results of existing N group experiments, $H = \{x_n, y_n\}_{n=1}^{N}$, and calculates the posterior distribution $p(f(x)|x, H)$ of $f(x)$.

After obtaining the posterior distribution of the objective function, an acquisition function $a(x, H)$ is defined to trade off in sampling where the model predicts a high objective and sampling at locations where the prediction uncertainty is high. The goal is left to maximize the acquisition function to determine the next sampling point. The Bayesian optimization process is summarized by the pseudo code in Algorithm 3.

Algorithm 3 Bayesian Optimization pseudocode.

$H \leftarrow \emptyset$
for $titteration \in \mathbb{N}$ **do**
  $s' \leftarrow argmax_x a(x, H)$
  evaluate $y' = f(x')$
  $H \leftarrow H \cup (x', y')$
  Remodeling Gaussian processes according to H, calculate $p(f(x)|x, H)$
return H

Genetic Algorithms

Bio-inspired techniques, such as Genetic Algorithms (GA), based on natural processes, have also been largely used for HPs tuning [24]. In these techniques, the initial population P generated randomly or according to the background knowledge, is updated in each iteration according to operators based on natural selection and evolution. The GA general pseudocode is presented in the Algorithm 4.

Due to the underlying assumptions made by different search methods, the choice of an appropriate search method can depends on the search space. Bayesian approaches based on Gaussian processes [18, 28] and gradient-based approaches [19, 29] are generally only applicable to continuous search spaces. In contrast, tree-based Bayesian [30, 31], evolutionary strategies [21], and random search are more flexible and can be applied to any search space. The application of reinforcement learning to general hyperparameter optimization problems is limited due to the difficulty of learning a policy over large continuous action spaces.

Algorithm 4 Genetic Algorithm pseudocode.

$t \leftarrow 0$
Generate initial population $P(0)$
Evaluate the current population $P(t)$
while Stopping criteria not satisfied **do**
    $t \leftarrow t+1$
    Select population $P(t)$ from $P(t-1)$
    Apply crossover operators in $P(t)$
    Apply mutation opperatos in $P(t)$
    for each new individual i in the current population $P(t)$ **do**
        Evaluate individual i fitness
$Best_{global} \leftarrow$ best individual i from $P(t)$ **return** $Best_{global}$

1.3 Machine learning for industrial big data analysis

Taking advantage of the Industrial Internet of Things and artificial intelligence, the fourth industrial revolution is relying more and more on machine learning methodologies. Nowadays, the ML algorithms are often used to satisfy the commercial plans, meet the profitability requirements, productivity and delivery time objectives [32]. The smart industrial monitoring systems integrate ubiquitous sensors and processors like any other classical application in industry 4.0 processes for equipment monitoring and timely fault detection. The ML algorithms, in this regard support the automatic defect inspection. They also deal with the control of production processes, enabling real-time synchronization of resources and product customization. Recently, Schmitt *et al.* [33] proposed a machine learning predictive model-based quality inspection. The model is trained on historic datasets in the cloud. The proposed system seeks to allow the operators to make informed decisions regarding expected product quality as well as maintenance operations. These results, among others, indicate a lot of interest in ML research and development for manufacturing applications.

This section contains an overlapped overview of two research areas : (1) the application areas of data analytics in manufacturing, and (2) the challenges of applyingML algorithms as well as knowledge engineering in Industry 4.0 area.

1.3.1 Application areas of machine learning in manufacturing

In manufacturing industry, ML techniques have been applied across three different fundamental stages composing the manufacturing projects that can be briefly described by the terms : *Plan* (set of tasks that deal with demand forecasting and jobs scheduling activities), *Make* (aims to optimize the manufacturing processes with respect to quality, time, and cost criteria), and *Maintain* (concerns mainly the activities like diagnostic & predictive maintenance). To understand machine anomalies and states, these terms map the fundamental levels for manufacturing projects, whose details are outlined below to analyze the internal mechanisms of advanced analytics techniques.

Advanced analytics practices at the process level

The focus of modern manufacturing departments, with the progress in Industry 4.0, has shifted from reactive to proactive methods in the recent decades. There has been significant development in defect prevention methodologies via process improvements [34]. Overall, we can observe that monitoring and predicting product quality is a crucial part of improving manufacturing processes [35].

In this context, data-driven machine learning methods provide an efficient way to control products quality. Consequently, the necessary process information can be learned directly from large amounts of production data [36]. Researchers have successfully demonstrated the use of ML techniques to improve the production quality by timely predicting faults and defects in industrial processes well ahead of risks of failure [37]. Chen *et al.* [38] proposed a data-driven method that enable automatic detection and localization of wire bonding defects in order to inspect wire bonding defects in integrated circuits (ICs). The proposed method principally involves three steps : (1) data pre-processing to locate and separate IC chip image patches from the raw image, (2) features engineering to extract geometric information from captured wire segments, and (3) machine learning algorithms (i.e., CNN and SVM) for detection and classification. On a set of X-ray images collected from a semiconductor factory, the authors, demonstrate the effectiveness of their developed method.

Similarly, for the detection and identification of faults in rotating machinery

parts, Dimitrios *et al.* [39] proposed a diagnostic system architecture based on multilayer perceptron with automatic relevance determination for the anticipation and mitigation of breakdowns in rotating machinery. The multilayer perceptron (MLP) with Bayesian automatic relevance determination has been used for the classification of accelerometer data. The obtained results showed that using kurtosis and the integral of the acceleration signal together is a promising technique for detecting bearing fault locations. The proposed architecture has increased the levels of accuracy in fault detection up to 99% for different fault types.

Advanced analytics practices at the machine level

In some research works, the machine learning applications are deployed to monitor and understand machine behaviors. Tool conditions, for example, have been monitored using machine learning techniques. Monitoring tools condition involves tracking the evolution of the tool state and detects a fault or breakage [40–42].

Existing literature features the applications of Support Vector Machines (SVM) for the condition monitoring of tools. Medina *et al* [43] proposed a machine learning based approach for fault classification in two mechanical equipments (i.e. gearbox and roller bearings). The fault classification is obtained by using a multi-class SVM. The proposed approach has been tested using a 10-Fold cross-validation strategy on the vibration signals of these equipments. Their final results show that the proposed approach could achieve a classification accuracy of 99.3% for the gearbox dataset and 100% for the roller bearings. Similarlly, an adapted deep neural network strategy [40] has been proposed for condition monitoring of an in-wheel motor (classification of the wear of inner race and outer race). The classification results of the approach achieved a classification accuracy of 99.8%.

Advanced analytics practices at the shop floor and supply chain levels

The advanced analytics are also applied in the manufacturing industry at a higher level. It is aimed at the correct production planning and control which can lead

towards the global improvement in manufacturing production systems [44].

The utility of applied analytics has also proven to solve the supply chain problems which are often complex *np-hard* combinatorial optimisation problems. Carbonneau *et al.* [45] used Neural Networks (NNs) to forecast demands in a supply chain to optimize the polynomial time costs and resource usage to fulfill the orders. They compared this technique with regular regression and SVM, concluding that SVMs and NNs are faster, but not more accurate than regular regression models. Likewise, Wu [46] proposed a hybrid intelligent system combining the wavelet support vector machine and particle swarm optimization for forecasting car sales. The obtained simulation results demonstrate the proposed approach as an effective solution in dealing with uncertain data and finite samples.

1.3.2 Challenges in building ML models with industrial big data

The *Predictive modeling* is crucial to transform large manufacturing datasets, or "industrial big data" into actionable knowledge for various industrial applications. The machine learning is widely used in many industrial applications across different levels, including processes, machines, shop floors, and supply chain levels. For instance, machine learning models can be used to control product quality [47], to monitor the condition of tools by tracking the evolution of their state [48], or to monitor the health of machines by predicting the time of occurrences of machine failures and also to estimate the criticality of these failures [49]. However, despite its countless benefits and advances, building a machine learning pipeline is still a challenging task, partly because of the difficulty in manually selecting an effective combination of an algorithm and hyperparameters values for a given task or problem. Both of these challenges concern the features engineering (dealing with the inputs of the ML algorithms) and the automatic selection and parametrization of the adequate model, as detailed in the following :

Challenge 1 : efficiently performing features engineering

Features engineering is the process of generating and selecting features from a given dataset for the subsequent modeling step. As the overall model performance highly depends on the available features. Feature engineering is a crucial process in the life cycle of a ML pipeline construction. The performance of a ML pipeline can be increased many times over by building good features. In many cases, the original features from the data may not be good enough, e.g., their dimensionality may be too high or samples may not be discriminable in the feature space [50]. Consequently, it is necessary to perform some pre-processing on these features to improve the learning performance.

Features engineering involves the application of some transformation functions such as arithmetic and aggregate operators on given features to build new features and remove data errors such as missing values in an input data entry, invalid values or broken links between entries of multiple data sets.

Given a predictive modeling problem, an analyst manually examines the quality of the available data, performs adequate transformations and then builds the model which is evaluated later on. If the model accuracy is insufficient, the analyst changes the applied transformation functions and operators for some attributes and re-builds the model. This labor-intensive process requires interactions between industrial professionals and computer scientists. Moreover, it is often repeated many times before converging, causing a time and human resource bottleneck especially in fields that do not tolerate delays such as the manufacturing industry.

Challenge 2 : efficiently selecting algorithms and hyperparameters values

Owing to the development of open source ML packages and the active research in the ML field, there are dozens of machine learning algorithms, where each machine learning algorithm has two types of model parameters : (1) *ordinary* parameters that are automatically optimized or learned during the model training phase; (2) and *hyperparameters* (categorical and continuous) that are typically set by the user manually before the training of the model (cf. Table 1.1).

Given a modeling problem like, predicting whether an equipment failure

will occur, an analyst builds a model manually and iteratively. Initially, the analyst selects an algorithm among the many other applicable algorithms like Logistic Regression, SVM, Random Forest or Naïve-Bayes. Subsequently, (s)he sets the hyperparameters values for the selected algorithm. Later on, (s)he trains the model to automatically optimize the ordinary parameters. If the model accuracy is insufficient, the analyst changes the hyperparameters values and/or the algorithm and re-builds the model. This process is iterated until (s)he obtains a model with sufficient accuracy,or (s)he no longer has time to optimize it or the model accuracy cannot be improved anymore (cf. Figure 1.1).

Numerous combinations of algorithms and hyperparameters values result in hundreds or thousands of labor-intensive manual iterations to build a model, which can be difficult even for experts in machine learning [51]. It is largely observed in the available literature and empirically proved that the algorithms and the used hyperparameters values, affect the model accuracy. Thornton *et al.* [52] have shown in their study that for the 39 ML algorithms in Weka, the effect of HPs tuning on the models accuracy, in average is equal to 46% on 21 datasets and it is of 94% on one dataset. Even when considering only a few common algorithms such as support vector machine and random forest; the effect is still greater than 20% on 14 out of 21 datasets.

Furthermore, the effective combination of an algorithm and the hyperparameters values varies with respect to the problem we attempt to model. In the literature, some authors explore the automatic search of algorithms and hyperparameters values [51]. Evidently, it shows that automatic search methods can obtain equivalent or even better results than those resulting from the manual tuning done by machine learning experts [26].

1.3.3 Common practices to apply advanced analytics for manufacturing related problems

Predictive analytics have gained significant interest among the industry 4.0 community. Machine learning based data analytics techniques are widely applied across different levels of the manufacturing industry. Wuest *et al.* [53] summarized the ability of machine learning techniques to meet manufacturing

requirements. According to their work, expressively not every machine learning technique is applicable to every manufacturing problem. As manufacturing stakeholders do not possess the necessary expertise to achieve these tasks, they often collaborate with data scientists who may provide guidelines for applying machine learning techniques. In most cases, these collaborations are complex and causes excessive consumption of time and effort [36]. Capabilities to perform advanced analytics without strong data science knowledge are highly desired to facilitate the application of advanced analytics in manufacturing.

1.4 Conclusion

In this chapter, the formal definition of the algorithms selection and related HPs tuning problem is presented along with their applications and challenges for industrial big data. Subsequent sections also described the main techniques often used to solve it, ranging from the most straightforward techniques GS and RS, meta-heuristic like GA; to more complex approaches like BO. All these techniques can improve the predictive performance of the final induced models, but they can also be very time consuming to find suitable HPs settings. Moreover, it is not guaranteed that the tuning process will lead neither to improved ML models nor significant improvements. As stated in the thesis' hypothesis, Automated machine learning can be useful to make ML algorithms selection and related HPs tuning more efficient and less costly.

Chapter 2

Automated machine learning

Outline of the current chapter

Machine learning became a vital part in many aspects of our daily life. To give just a few examples, ML can suggest active user which books or newspapers to read [54], what movies to watch [55], what music to listen to [56], etc. ML approaches has been employed to develop systems able to recommend which places (e.g., cultural and artistic attractions [57]) to visit [58] and the best itinerary to get there [59], and have been proven to be very efficient in

self-driving cars [60] and predictive maintenance in Industry 4.0 [61]. However, along with these global applications, there is also widespread awareness that, given a specific problem, the process of designing and implementing a truly effective and efficient ML systems requires considerable knowledge and effort by highly specialized data scientists and domain experts. Each algorithm is intrinsically optimized and its performance on a particular task depends on how well its embedded fixed bias match the problem. Hence, there is no single algorithm that can learn all the tasks efficiently and every algorithm can perform better only on limited number of tasks. This phenomenon is alsocalled performance complementarity [62], and is also been confirmed by the well known *No Free Lunch* theorem [63].

Since no single algorithm can best learn all the tasks effectively, the question that which algorithm should be used from the large number of available algorithms for a given task has gain tremendous importance and attention. The complete or partial automation of roles that today require human skills would therefore be welcomed with great interest. Based on this motivation, Automated Machine Learning [14] has now become one of the most relevant research topics not only in the academic field but in the industrial one too [64, 65]. The main purpose of AutoML is to provide seamless integration of ML in various industries, which will reduce the demand for data scientists by enabling domain experts to build ML applications automatically without extensive knowledge of statistics and machine learning.

2.1 Automated machine learning

Although ML algorithms do not require human interference while learning, preparing data that is going to be consumed by these algorithms, finding the right algorithm, and tweaking it to get the best results require skilled data scientists. Data scientists try different techniques for preprocessing and multiple ML algorithms to come up with the combination that is the most efficient (cf. Figure 1.1). These processes are human dependent and require special skills in computer science, programming, mathematics, and statistics, in addition to business knowledge in the area of the processed data [53].

A novel research area has then emerged. Its main goal is to enable non-ML experts to effectively generate and configure data analytics solutions, without special assistance or intervention, on the real-world problems [14]. Furthermore, it leverages human expertise by allowing users to define the conditions that restrict the algorithms as well as the performance metrics to be used while evaluating candidate algorithms and thus save the time and effort for knowledgeable practitioners. The core problem considered by AutoML can be formulated as follows : given a dataset, a machine learning task and a performance criterion; solve the task with respect to the dataset while optimizing the performance [66]. The goals of AutoML can be summarized as listed below :

- democratizing the application of ML to non-experts of data analysis by providing them with "off the shelf" solutions,
- enabling the knowledge practitioners (e.g. engineers, researchers) to save time and effort,
- increasing the productivity by playing the role of a shield against methodological errors and over/ under-estimations of performance.

Multiple approaches have been proposed to support the machine learning automation. These approaches range from automatic data pre-processing [67] to automatic model selection [68, 69]. Some approaches [70, 71] attempt to automatically and simultaneously select the right learning algorithm and find the optimal configuration of its hyperparameters. These approaches are also referred as combined algorithm selection and hyperparameters optimization problem (CASH) [71, 72]. A solver for the CASH problem aims to pick an algorithm from a list of options and then tune it to give the highest validation performance amongst all the possible combinations of algorithms and hyperparameters.

Owing to the immense potential of AutoML, different learning paradigms have been applied to this task, and tools are available to the research community such as Auto-sklearn [73], AutoWEKA [70], TPOT [71] as well as commercially ones such as RapidMiner [74], H2O [75], Big ML [76], and Data Robot [77]. An ongoing competition [78] around this goal has been running since 2015 focusing on various budget-limited tasks for supervised learning.

Automation of analytics workflows or their parts have been studied and attempted actively over the past years. As a result, there are various approaches to support data scientists and neophyte ML domain experts. Hereafter, we show two main approaches respectively based on the meta-learning and the use of ontologies.

2.1.1 Meta-learning based approach

When one learn new skills, (s)he rarely, if ever, starts from scratch. (S)He starts from skills learned earlier in related tasks, reuses approaches that worked well before, and focuses on what is likely worth trying based on experience [79]. With every learned skill, learning new skills becomes easier, requiring fewer examples and less trial-and-error. In short, *(s)he learns how to learn across tasks*. Likewise, when building machine learning models for a specific task, we often build on experience with related tasks, or use our (often-implicit) understanding of the behavior of ML techniques to help make the right choices.

Several ML algorithms have been proposed for prediction tasks. However, since each algorithm has its inductive bias, some of them can be more appropriate for a particular dataset. When applying an ML algorithm to a dataset, a higher predictive performance can be obtained if an algorithm whose bias is most appropriate to the datasets is used. Thus, non-experienced users become overwhelmed and require support (e.g., to be recommended the algorithm and related HPs configuration to use). The recommendation of the most adequate ML algorithm configuration for a new dataset is investigated in a research area known as Meta-Learning (MtL).

Meta-learning or learning to learn is the process about learning how machine learning algorithms perform across a range of tasks. It aims to learn which algorithm will work well for a dataset with certain characteristics, or which hyperparameters will give a good performance. Thus, the MtL investigates the relation between tasks/problem domains and learning strategies. The goal is to select the most promising algorithm for a given task and to understand when a particular learning strategy is more suitable than others. This task is also commonly referred to as "Algorithms Selection" [80].

The idea of meta-learning for the algorithm selection and HPs tuning problems is based on the following assumption : "Algorithms show similar performance for the same configuration for similar problems". It consists of generating a meta-model that maps the characteristics of problems to the performance of algorithms that can be used to solve these problems.

The challenge in meta-learning is to learn from prior experiences in a systematic and data-driven way. As depicted in Figure 2.1, meta-learning consists of 3 main phases : first, a meta-learning space is established using *meta-data* that describe prior learning tasks and previously learned models. It consists of datasets characteristics (*meta-features*) and a performance measure (*meta-responses*) for data mining algorithms on these particular datasets. Then, there comes the meta-learning phase. Here, a predictive *meta-model* is generated out of the meta-dataset constructed in the first phase, to extract and transfer knowledge that guides the search for optimal models for new tasks. Finally, in the third stage when a new dataset occurs, its characteristics are extracted and the predictive meta-model is used to recommend the most promising ML algorithms with related HPs configuration.

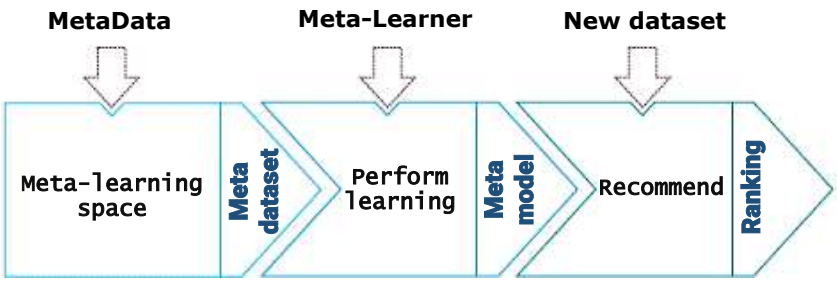

Figure 2.1: The meta-learning process.

The workflow to build a *meta-model* is shown in Figure 2.2. In order to train a meta-model, a *meta-dataset* from which to learn is required. A meta-dataset contains informations about machine learning experiments. For instance, it captures which ML algorithm is used, with which hyperparameters configuration, and how well the resulting model performed. Additionally, for each such experiment it also describes the dataset on which the experiment was performed. The description of the dataset is done by *meta-features* that capture information

about the data, such as the number of attributes, classes, kurtosis, skewness, etc.

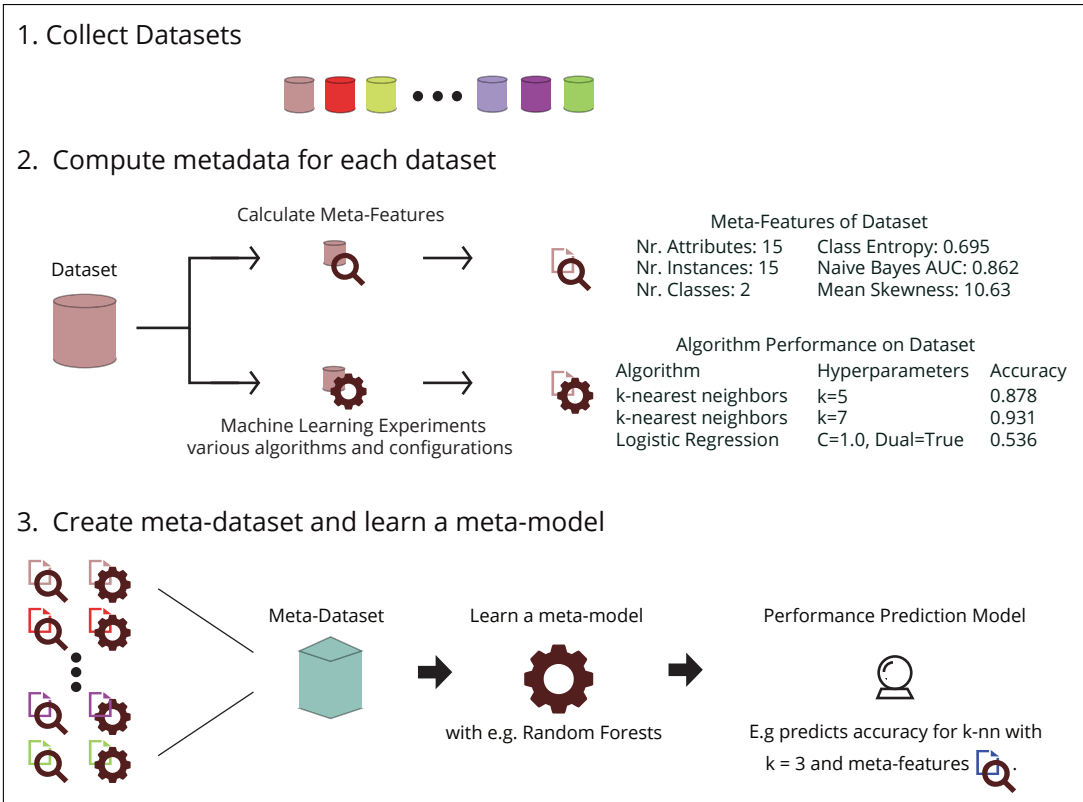

Figure 2.2: An overview of the steps to creat a meta-model.

The first step to create a meta-dataset is to select a set of datasets to perform ML experiments on. This is illustrated as the first step in Figure 2.2. In the second step, two actions have to be performed on each dataset. On one hand, the meta-features (characteristics) of the dataset need to be calculated. Meta-features describe the dataset in various ways. Examples include the number of classes in the dataset, the mean skewness of numerical features or the accuracy of simple classifiers. More information about the various meta-features follows in the next section. On the other hand, ML experiments have to be run on the datasets. The type of ML experiments that are run should be the same type you want the meta-model to make predictions about. In the final step, the results of step two are combined into a single meta-dataset. This dataset contains a row for each machine learning experiment, describing the used algorithms

and related hyperparameters configurations, the achieved performance, and the meta-features of the dataset on which they were achieved.

Following these requirements, a suitable meta-dataset is generated, and as result of the MtL process, a meta-model is induced. This meta-model represents a mapping between the meta-features describing the datasets, and the predictive performance obtained by the group of learning algorithms when applied to these datasets. Therefore, the quality of the meta-features is essential for the predictive performance of the meta-models. Hence, it can be used to recommend the best algorithms for a new unseen problem.

The two main concepts of meta-learning are the *meta-data* and the *meta-model*. In the following, we briefly discuss these two concepts.

Meta-data

To create an effective meta-model, the meta-data from which to learn is required. Meta-data are the necessary information required to establish the meta-dataset. In our definition, they consist of: i) *meta-features*, and ii) performance measures of the considered algorithms — *meta-responses*. In statistics, the former are called *predictors* and the latter are called *responses*.

Meta-features A fundamental MtL question is : *how to extract suitable information to characterize specific tasks?*. Researchers have been trying to answer this question by looking for dataset properties that can affect learning algorithms performance, measuring this performance outright [81, 82], investigating alternatives [83] and adapting/creating new measures based on existing ones [84, 85]. In all cases, different types of meta-features have been developed, ranging from simple features such as the number of samples in a dataset, to more complex ones. Next, we provide a concise overview of the most commonly used meta-features, together with a short rationale for why they are indicative of model performance. Where possible, we also show the formulas to compute them. More complete surveys can be found in the literature [13, 86–88].

1. **Simple meta-features**
   The simple measures are directly extracted from the data and they represent basic informations about the dataset. They are the simplest set of measures in terms of definition and computational cost [89–91]. Table 2.1 presents some of these measures. They are directly computed, free of hyperparameters and deterministic. Semantically, these measures represent concepts related to the number of predictive attributes, instances, target classes and missing values. These measures are relevant to characterize the main aspects of a dataset, providing information that can support the choice of a learning algorithm for a particular task.

Name	Formula	Rationale	Variants
Nr instances	n	Speed, Scalability	$p/n, log(n), log(n/p)$
Nr attributes	p	Dimensionality	$log(p), \%categorical$
Nr classes	c	Complexity, imbalance	$ratio, min, maxclass$
Nr missing values	m	Imputation effect	$\%missing$
Nr outliers	o	Data noisiness	o/n
attrToInst		Dimensionality	
instToAttr		Sparsity	

Table 2.1: Simple measures and their characteristics.

The number of instances and the number of classes by themselves do not provide much information since they indicate the dataset size and its label diversity. However, when combined with the number of attributes, different simple concepts can be captured. The measures attrToInst and instToAttr represent the dimensionality and sparsity of the data, respectively. The latter is a potential indicator for overfitting when its value is too small, a learning model may take into account irrelevant details in the training data, resulting in poor generalization [92]. Finally, some measures assess dataset quality, such as the number of missing values in the dataset attributes and instances, as well as the total number. Since some ML algorithms can deal with missing values, these measures can provide important information for the algorithms selection.

2. **Statistical meta-features**
   Statistical measures are used to extract informations about the performance of statistical algorithms or about data distribution, for instance, central tendency and dispersion [89]. They are the largest and the most diversified group of meta-features, as shown in Table 2.2. Statistical measures are deterministic and support only numerical attributes.

Name	Formula	Rationale	Variants
Skewness		Feature normality	$min, max, \mu, \sigma, q_1, q_3$
Kurtosis		Feature normality	$min, max, \mu, \sigma, q_1, q_3$
Correlation		Feature interdependence	$min, max, \mu, \sigma, \rho_{XY}$
Covariance		Feature interdependence	$min, max, \mu, \sigma, Cov_{XY}$
Sparsity		Degree of discreteness	min, max, μ, σ
Gravity		Inter-class dispersion	
ANOVA p-value		Feature redundancy	$P_{val_{XY}}$

Table 2.2: Statistical measures and their characteristics.

   Correlation and covariance capture the interdependence of the predictive attributes [89]. They are computed for each pair of attributes in the dataset. The former is a normalized version of the latter, and the absolute value of both measures are frequently used, which changes the range from $[-1;1]$ and $]-\infty;+\infty[$, respectively. High values indicate a strong correlation between the attributes, which can be interpreted as a level of redundancy in the data [93].

3. **Information-Theoretic meta-features**
   Information-theoretic meta-features capture the amount of information in the data. Table 2.3 shows the information-theoretic measures, which require categorical attributes and most of them are restricted to represent classification problems. Moreover, they are directly computed, free of hyperparameters, deterministic and robust. Semantically, they describe the variability and redundancy of the predictive attributes that represent the classes.

Name	Formula	Rationale	Variants
Class entropy	$H(C)$	Class imbalance	
Norm. entropy	$\frac{H(X)}{log_2 n}$	Feature informativeness	min, max, μ, σ
Mutual inform.	$MI(C,X)$	Feature importance	min, max, μ, σ
Uncertainty coeff.	$\frac{MI(C,X)}{H(C)}$	Feature importance	min, max, μ, σ
Equiv. nr. feats	$\frac{H(C)}{MI(C,X)}$	Intrinsic dimensionality	
Noise-signal ratio	$\frac{H(X)-MI(C,X)}{MI(C,X)}$	Noisiness of data	

Table 2.3: Information-theoretic meta-features and their characteristics.

The entropy of the predictive attributes and the target values capture the average uncertainty present in the predictive and class attributes [86], respectively. In the former, all predictive attributes are assessed, thus its summarization can provide an overview of the attributes capacity for class discrimination. In the latter, it represents how much information is necessary to specify one class. In a learning perspective, a predictive attribute with a low entropy contains a low discriminatory power [89], whereas a target attribute with low entropy contains a high level of purity.

Since the problem to be solved is usually a prediction problem, and, a variable (or more) is defined to be the response, further meta-features measuring the association between the predictors and the response have been used. These measures are grouped into the Landmarking and Model-based classes [94]. Yet, when performed on bigger datasets, these measures may introduce significant computational costs.

4. **Landmarking**
   Landmarking is an approach to characterize datasets using the performance of a set of fast and simple learners, which extract information from the learning models. Examples include the performance of a decision stump, naive bayes classifier or linear discriminant analysis [94]. Table 2.4 lists the most common landmarking measures. Good landmarkers should probably have a runtime complexity of at most $O(nlog(n))$. Moreover, when using multiple landmarkers, they should have different biases [94].

If two distinct landmarkers have similar performance across all datasets, then it probably suffisant to only use one of them [95]. These measures were explored in studies such as [18, 73, 91].

Name	Formula	Rationale	Variants
Landmarker(1NN)	$P(\theta_{1NN}, t_j)$	Data sparsity	Elite 1NN [96]
Landmarker(Tree)	$P(\theta_{Tree}, t_j)$	Data separability	Stump,RandomTree
Landmarker(Lin)	$P(\theta_{Lin}, t_j)$	Linear separability	Lin.Disciminant
Landmarker(NB)	$P(\theta_{NB}, t_j)$	Feature independence	More models [97]
Relative LM	$P_{a,j} - P_{b,j}$	Probing performance	
Subsample LM	$P(\theta_i, t_j, s_t)$	Probing performance	

Table 2.4: Landmarking meta-features and their characteristics.

5. **Data complexity**
   Complexity measures are a set of measures which analyze the complexity of a problem considering the overlap in the attributes values, the separability of the classes, and geometry/topological properties. They were introduced in [98] to capture the underlying difficulty of classification tasks. Table 2.5 summarizes the main characteristics of these measures. While a complete survey of the complexity measures can be found in [99, 100].

Name	Formula	Rationale	Variants
Fisher's discrimin	$\frac{(\mu c1 - \mu c2)^2}{\sigma_{c1}^2 - \sigma_{c2}^2}$	Separability classes c_1, c_2	See [100]
Volume of overlap		Class distribution overlap	See [100]
Concept variation		Task complexity	See [101]
Data consistency		Data quality	See [13]

Table 2.5: Data complexity meta-features and their characteristics.

6. **Model structure-based meta-features**
   Contrarily to the previous data characterization methods that were calculated from the data distribution, the model based is an indirect characterization method and are calculated by inducing a decision tree model on a dataset to get information about the hidden structures of the data [62,

91]. The properties of the tree are used as meta-features. Its advantage is that it does not only rely on the distribution of the data but consider the representation of the data set in a special structure for getting information about the learning complexity. However its drawback is that it has relatively high computational cost associated with it. Table 2.6 shows the DT model meta-features.

Name	Formula	Rationale	Variants
Nr nodes, leaves, and branches	$\|\eta\|, \|\psi\|$	Concept complexity	$Tree\ depth$
Nodes per feature	$\|\mu X\|$	Concept complexity	min, max, μ, σ
Leaves per class	$\frac{\|\psi_c\|}{\|\psi\|}$	Feature importance	min, max, μ, σ
Leaves agreement	$\frac{n_{\psi_i}}{n}$	Class complexity	min, max, μ, σ
Information gain		Class separability	min, max, μ, σ

Table 2.6: Model-based meta-features and their characteristics.

Many other non-traditional characterization measures have been reported in the literature. Despite the fact they are not broadly used in MtL studies, due to a high computational complexity or domain bias, they can be useful for a particular learning scenario and MtL problem. Besides, some works show good results when using those characterization measures [87, 98].

Meta-reponses Performance measures are different outputs that can be obtained after the evaluation of data mining algorithms. Since we are dealing with classification problems, then, the algorithms we consider are of classification type. The performance is usually measured in terms of predictive accuracy, precision, recall or the area under the roc curve (AUC). In Table 2.7, formulas for calculating these measures are given.

Meta-model

After having generated a meta-dataset with all the necessary metadata, the goal is to build a predictive meta-model that can learn the complex relationship between the tasks characteristics (meta-features) and the utility of specific ML pipeline to recommend the most useful ML algorithm configuration $A^{(i)*}$ given

Measure	Formula
Accuracy	$\frac{(TP+TN)}{(TP+FP+FN+TN)}$
Precision	$\frac{TP}{(TP+TN)}$
Recall	$\frac{TP}{(TP+FN)}$
AUC	$P(X_2 > X_1)$

Table 2.7: Performance evaluation measures for classification algorithms. `TN` - True Negatives; `TP` - True Positives; `FN` – False Negatives; `FP` - False Positives; `X1`, `X2` - Score functions of the classes.

the meta-features $\mathcal{M}$ of the new task t_{new}. Formally, each task $t_j \in \mathcal{T}$ is described by a vector $m(t_j) = (m_{j,1}...m_{j,K})$ of K meta-features. This can be used to define the task similarity measure based on, for instance, the Euclidean distance between $m(t_i)$ and $m(t_j)$, so that we can transfer information from the most similar tasks to the new task t_{new}.

Different meta-learners have been used in the literature, such as k-nearest neighbors (KNN), decision trees, and XGBoost [4]. The various approaches investigating the use of MtL in providing recommendation of the ML pipeline to use give suggestion in one of the following forms, depending on the aspect to which MtL is applied :

a) List of applicable algorithms

In this category, we classify all the approaches where the suggestion consists of a single algorithm or alogirthms, that is (are) expected to perform best on the dataset, according to the performance criterion that is used. In these approaches, the HPs settings are predicted without actually evaluating the model on the new dataset [80]. In this category, we found the work done by [102], where the autors employed MtL to recommend the optimal algorithm for a given task. The authors performed experiments with C4.5, kNN and SVM classifiers covering few of their HPs by a grid design. Performances were evaluated over 100 UCI [1] datasets regarding AUC. As a meta-model, the kNN algorithm was applied to recommend the best HPs setting for new unseen datasets. Similarly, [91] evaluated

[1] `http://archive.ics.uci.edu/ml/index.php`

different ML algorithms over 54 datasets and used the performance predictions to develop a MtL system for automatic algorithms selection. In [103], the authors conducted a similar study with SVMs but using a Genetic Algorithm (GA) to optimize HPs and perform features selection of six meta-learners (kNN, SVM, J48, JRip, NB, and Bagging). Experiments were carried out over 78 classification datasets assessing HPs settings using a 5-fold cross validation strategy and the Mean Absolute Deviation (MAD) evaluation measure.

Meta-models can also generate a ranking of the top-K most promising configurations. [104] proposed the "autoBagging" tool, an AutoML system that automatically ranks Bagging work-flows considering four different Bagging HPs by exploring past performances and datasets characterization. Experiments were carried out on 140 OpenML [2] datasets and 146 meta-features (extracted with post-processing aggregation functions). They used an XGBoost as a meta-learner to predict ranking of workflows, and evaluated results at the meta-level using a Mean Average Precision (MAP) measure in a Leave-One-Out Cross-Validation (LOO-CV) strategy. The advantage of a ranking is that there is an ordered set of suggestions to try. Additionally, a user may have preferences about the algorithm used, based on e.g. computational efficiency or model interpretability.

b) Predict HPs tuning necessity and training runtime

A different approach is to use MtL to check the HPs tunnig necessity. [105] employed MtL to predict the improvement obtained in a DT algorithm varying its HPs. They selected the C4.5 algorithm and defined a grid of values for the hyperparameters **C** and **M**. A total of 14 educational datasets were characterized by means of 5 simple meta-features. Thus, the MtL was used to predict whether the use of different HPs settings increase, decrease or maintain the predictive performance of the induced DTs. Similarly, [106] used MtL to identify when HPs tuning would lead to significant increase in accuracy. They carried out experiments using a PSO [107] technique to search the hyperspace of several ML algorithms in 326 binary

[2] https://www.openml.org/

classification datasets. The analysis performed by the authors considered different thresholds to determine when an improvement was obtained or not in the data collection, but no statistical analysis was performed.

Instead of predicting predictive performance, a meta-regressor can also be trained to predict algorithms runtime (training/prediction time) when induced by different HPs settings. For instance, Reif, *et al.* [108] predicted the training runtime of several classifiers : kNN, SVM, Multilayer Perceptron (MLP), and DT. They defined a discrete grid of HPs settings, evaluating these settings on 123 classification datasets. The performance measures used for HPs assessment were the Pearson Product Moment Correlation Coefficient (PMCC) and Normalized Absolute Error (NAE). Similarly, [109] predict both the predictive performance and runtime using polynomial regression, based only on the number of instances and features.

c) Estimate predictive performance for a given HPs setting

Meta-models can also directly predict the performance, e.g. accuracy or training time, of a configuration on a given task, given its meta-features. The final meta-target is to predict an estimation of the performance. This makes the task a regression problem allowing to estimate whether a configuration will be interesting enough to evaluate in an optimization procedure. Early works [110, 111] used linear regression and rule-based regressors to predict the performance of a discrete set of configurations and then rank them accordingly.

To predict the performance of a set of machine learning algorithms, [112] trained an SVM meta-regressor per classification algorithm to predict its accuracy, under default settings, on a new task t_{new} given its meta-features. Similarly, [91] trained a meta-regressor on more meta-data to predict its optimized performance. More recently, [113] adapted the acquisition function of surrogate models by one optimized meta-model. They evaluated several HPs in a holdout fashion procedure over 105 datasets and used the meta-knowledge to predict the performance of new HPs settings for new datasets. The base-level algorithms explored were the AdaBoost and SVM.

Research papers from the detailed literature survey that either embedded or used MtL to cope with these tasks are summarized in Table 2.8.

Reference	Task	N. of learner(s)	Meta-features				Meta-learner(s)	N. of datasets	Evaluation mearsure(s)
			SI	MS	LM	DC			
[104]	Recommends HP settings	63 bagging workflows	✓	✓	✓	-	Xgboost	140	Kappa score
[102]	Recommends HP settings	3	✓	✓	✓	✓	KNN	84	AUC
[114]	Recommends HP settings of SVM	1	✓	-	-	-	KNN	40	Acc
[99]	Recommends HP settings of SVRs	1	-	-	-	✓	kNN	39	NMSE
[115]	Predicts training runtime	5	✓	-	✓	-	PMCC	123	NAE
[103]	Predicts training runtime	6	✓	-	-	-	SVM	78	MAD
[114]	Recommends initial values for HP optimization of SVM	1	✓	-	-	-	kNN	40	Acc
[20]	Recommends initial values for HP optimization	3	✓	-	✓	-	kNN	57	Acc
[106]	Predicts HP tuning necessity	-	✓	✓	✓	-	J48, SVM, RF	42	Acc
[113]	Estimates predictive performance for a HP setting	2	✓	-	-	-	AdaBoost	25	-
[116]	Ranked list of ML pipelines	22	✓	✓	✓	✓	KNN	115	Acc + Runtime
[117]	Ranked list of ML pipelines	6	✓	✓	-	-	C4.5 , genetic fuzzy	40	Acc + Runtime
[118]	Ranked list of ML pipelines	17	✓	✓	✓	✓	EM	84	Acc + Runtime
[119]	Ranked list of ML pipelines	21	✓	✓	✓	✓	link prediction	131	Acc + Runtime

Table 2.8: Summary of related studies applied to MtL to support the automation of ML.

2.1.2 Summary of literature overview

The literature review on MtL related works, leads to identify some interesting aspects. Overall, the following aspects were observed: :

- most of the studies created the meta-data using GS to tune the algorithms' HPs;
- most of them evaluated the resultant models with a holdout or single CV resampling procedure and the simple Accuracy evaluation measure;
- the majority of the studies used less than 100 datasets with few exceptions that used more than 100 datasets [104, 115, 116, 119], but all of them are binary classification problems. There is no clear indication in literature regarding the adequate number of datasets at the meta-level, however, reasonable number of datasets must be considered that can appropriately map the features space into the performance one [62, 120];
- different algorithms were tried at the meta-learning level but mostly concentrated by the same study and consider just one evaluation measure;
- all investigated a small number of categories (simple and statistical) meta-features to characterize the datasets. Therefore, different approaches to generate meta-features are not well explored in the literature;
- few of them provide the complete resources for the reproducibility of experiments;
- an end-to-end MtL resolver for the CASH problem has not been proposed nor ivestigated;
- none of the studies found by the author combined all these previous issues or limitations.

2.1.3 Ontology based approach

The ontology-based approach for semantic data mining attempts to make use of formal ontologies in the data mining process. A well-designed ontology can assist data analysts and neophyte ML domain experts to select appropriate modeling techniques and build specific models as well as the rationality for the techniques and models selected in a number of ways. By expressing the domain expertise in a formal structure, one can use logical reasoning to reduce the search space and hence, find the most predictive model for a given problem.

In contrast to the conventional data-driven MtL approach, semantic data mining is extensively co-driven by the knowledge of the data-mining process as well as its components expressed in a data mining ontology and knowledge base.

Bernstein *et al.*[121] describe an ontology-based Intelligent Discovery Assistant (IDA). After analyzing an input dataset (to extract meta-data), the system generates all possible workflows from the ontology and which are valid within the characteristics of the input. The recommendations are then sorted based on some criteria specified by the user (e.g., simplicity, performance metric, etc.). Similarly, Nural *et al.*[122] provided a comparison of the meta-learning approach as described in [123] with the ScalaTion ontology-based suggestion approach [124] on a set of 114 datasets. Using ScalaTion, each dataset is provided as an input to the suggestion engine and the suggested modeling technique is recorded. When predicting the top-1 performing technique, the ontology-based approach achieved an accuracy of 51% compared to 75% with meta-learning.

2.1.4 Background on AutoML systems and their components

AutoML systems have recently gained traction in the research community and there exists a multitude of approaches, often accompanied by open-source software. Bayesian optimization is a technique that optimizes hyperparameters for ML algorithms based on a well-known theory in probabilities called Bayes' theorem [18, 52, 64]. Other simpler techniques are also used such as grid search and random search. Meta-Learning is another method for hyperparameter optimization, where the AutoML system learns from its own experience of applying machine learning. The literature reveals a variety of AutoML tools and platforms. Some of theme are open-source while others are commercial. Table 2.9 shows a comparison among some of the most popular AutoML platforms, in terms of cost, coding requirements, processing location, input data requirements, and supported Operating Systems.

Auto-WEKA [70] is an AutoML framework with ongoing improvements [72] for building machine learning pipelines based on the Weka [125] ML library. Auto-Weka addresses the CASH problem applying the Bayesian optimization.

Auto-Sklearn [73] is an AutoML toolkit implemented on top of the Scikit-Learn[3] data-mining library. Auto-sklearn extends the idea of configuring a general machine learning framework with global optimization which was introduced with Auto-WEKA. To improve generalization, auto-sklearn builds an ensemble of all models tested during the global optimization process. It uses the ensemble construction and Bayesian optimization search procedures to address

[3] https://scikit-learn.org

System	Cost	Coding need	Data type	Operating system		
				Linux	Mac	Windows
Google AutoML [126]	Billable	No	Img, Txt Tabular	Cloud computing		
H2O.ai [75]	Billable	No	Img, Txt Tabular	Cloud computing		
Rapidminer [74]	Billable	No	Img, Txt Tabular	Yes	Yes	Yes
Auto Keras [127]	Free	No	Image	Yes	Yes	Yes
Auto-Sklearn [73]	Free	Yes	Tabular	Yes	No	No
ATM [128]	Free	Yes	Tabular	Yes	-	-
TPOT [21]	Free	Yes	Tabular	Yes	Yes	Yes
Auto-WEKA [72]	Free	Yes	Tabular	Yes	Yes	Yes

Table 2.9: Summary of related AutoML systems.

the CASH problem. Auto-sklearn wraps a total of 15 classification algorithms, 14 feature preprocessing algorithms.

The Tree-based Pipeline Optimization Tool (TPOT) [21] employs the genetic programming algorithms to optimize classification and regression ML pipelines by exploring many different possible pipelines. Each pipeline consists of a machine learning model and their hyperparameters configuration. TPOT can only handle categorical parameters; similar to grid search all continuous hyperparameters have to be discretized. While their evolutionary strategy can cope with this irregular search space, many of the randomly assembled candidate pipelines evaluated by TPOT end up as invalid, thus, wasting valuable time that could have been spent training the valid models.

Among the big market actors, Google Cloud Platform recently released the AutoML Tables [126], a supervised learning service that handles end-to-end AutoML, but it is only available on Google Cloud as a managed service of the commercial framework.

While all of these tools provide partial or complete ML process automation, each one works differently, and targets different dataset structure, platform, algorithm, or end user, posing unique advantages and disadvantages at the same time. For instance, Auto-Sklearn is embedded in Python, however, it only operates on structured data using *Linux Operating System*. Auto-WEKA supports Weka ML algorithms with the advantage of a graphical user interface (GUI), but it is limited to statistical algorithms. RapidMiner provides features engineering capability but requires expert guidance. While Google AutoML supports most

datasets and algorithms, the service is only cloud-based, and mostly commercial for dedicated data processing.

In this thesis, we investigate a way of recommending ML algorithms with their related HPs configurations. In the literature, there are some studies with proposals close to our objectives. Unlike these related studies, the MtL experiments described in this thesis explored a large number of heterogeneous and real world datasets, an unbiased tuning methodology, and induced meta-models for different target algorithms and predictive measures (cf. Chapter 3). These meta-models generate promising configurations by the use of a novel and efficient methodology to automatically extract more informational meta-features from the traditional ones (cf. Chapter 4).

2.2 AutoML in the manufacturing industry

Although AutoML has been applied to a range of purposes and applications such as crash prediction [129], clinical big data [130], disease diagnoses [131], sentiment analysis [132] and educational data mining [133], few attempts have been made to apply these techniques in the manufacturing field. It is largely observed that the industrial needs are yet to be satisfied as the industrial actors mostly use traditional ML processes rather than AutoML.

In manufacturing industry, there have been a few studies focused on implementing AutoML systems that are specialized for manufacturing services. Considering the lack of funds for industrial coding [129] and high data scientist salaries [134], it is essential to find a cost-saving method that allows manufacturing organizations to benefit from machine learning capabilities without huge costs. More importantly, such a methods may improve production outcomes, which is of paramount importance in developing any industrial tool. As an emerging technology, AutoML can help achieve these goals for manufacturing organizations, especially for extracting diagnoses from production data, which is the focus of this thesis. This will save not only manufacturing workers' time, but it will also improve production outcomes by accelerating data treatment planning and improving the accuracy of diagnoses.

Overall, we found two general approaches that have been studied to use AutoML in the manufacturing industry. The first approach is to *use already existing AutoML tools and platforms* to perform predictive modeling or classification on manufacturing related problems, while the second approach is to *build new AutoML tools for industrial big datasets*. Below, we discuss these approaches in more details.

2.2.1 Using existing AutoML tools for manufacturing datasets

AutoML systems are tools that propose to automate the ML pipeline : integration, preparation, modeling and model deployment. Although all AutoML systems aim to facilitate the usage of ML in production, they may differ on how to accomplish this objective, approaching the ML pipeline in different levels. The purpose of this section is to, using currently available AutoML systems, evaluate how each system approaches the ML pipeline and help a user to choose which ML pipeline configuration to pick.

In order to asses the effectiveness of state-of-the-art AutoML systems for manufacturing datasets, these were tested on a highly varied selection of 15 datasets that cover both binary and multiclass classification problems from the perspectives of different industry 4.0 levels. These data are gathered broadly from two major sources :

- **OpenML AutoML benchmark**: Datasets curated to serve as representative benchmark for AutoML frameworks [135]. These datasets span various binary and multi-class classification problems and exhibit substantial heterogeneity in sample size and dimensionality.

- **State of the art papers**: Datasets collected from research papers dealing mainly with industry 4.0 related problems using machine learning solutions.

Benchmarked datasets characteristics are shown in table 2.10. The datasets was pre-processed before being tested by each system. The evaluation results in terms of predictive preformance and rutime are presented in the tables 2.11 and 2.12 respectively.

Dataset	Num Classes	Num Instances	Task
[136]	4	959	Failure risk analysis
[137]	3	2000	Chatter prediction
[138]	2	61000	RUL prediction
[139]	2	7586	CNC Mill Tool Wear
APSFailure	2	60000	APS system failure prediction
Vehicle	2	846	Silhouette classification
CustSat	2	76020	Customer Satisfaction

Table 2.10: List (sample) of datasets used in the evaluation.

Dataset	AutoML tools Accuracy			Original paper result
	TPOT	Auto-sklearn	AutoWeka	
[136]	**0.9120**	0.8215	0.8353	0.85
[137]	0.9517	**0.9632**	0.9594	0.95
[138]	**0.9907**	0.9782	0.8398	0.9895
[139]	**0.9991**	0.9357	0.8868	0.9984
[140]	0.6711	0.908	**0.9689**	0.9677
[141]	0.7767	0.678	0.8334	**0.9278**
[142]	**0.8899**	0.6783	0.8477	0.884
[143]	0.7826	0.6702	**0.8942**	0.8659
vehicle	0.8415	**0.9027**	0.8415	-
Gas_Sens	**0.9843**	0.9256	0.9102	-
shuttle	0.9905	0.8429	**0,9953**	-
APS Failure	**0.9933**	0.9716	0.9559	-
CustSat	0.8276	0.8072	**0.8571**	-
car	**0.9999**	0.8549	0.9197	-
airlines	0.6758	**0.7094**	0.6493	-
Best performance	7	3	4	

Table 2.11: Performances of the selected AutoML frameworks on the benchmark datasets. The best performances among all AutoML frameworks are highlighted in bold.

As shown in table 2.11, all systems presented a more than acceptable model as a solution for particular manufacturing problems. It is obvious, that the results of some machine learning solutions, oriented from manufacturing industry, can be improved simply through the use of better ML models and related hyperparameters configuration.

With state of the art results, a distinction between better and worse system is hard to establish here. Performance also depends on the runtime, computational complexity and running budget, since running bayesian optimization or genetic algorithms based systems for more time can output better results.

In most of the state-of-the-art AutoML systems, one of the shortcoming is their computational complexity [14]. Oftenly, they require huge time and resources budget on non-conventional datasets. The available literature witness that the majority of state-of-the-art tools evaluate a set of pipelines by actually executing them on a given dataset prior to the recommendation [14]. It is observed in table 2.12 that such executions may require considerable computing time while consuming precious resources as per their availability [144].

Dataset	Dataset size	AutoML tools runtime		
		TPOT	Auto-sklearn	AutoWeka
[136]	959	00:08:14	01:23:47	00:32:14
[137]	2000	00:13:57	01:49:21	01:31:30
[138]	61000	03:42:09	04:19:05	04:02:27
[139]	274627	06:09:51	08:19:37	07:13:12
[140]	5000	01:38:36	02:31:07	03:20:27
[141]	1567	00:19:47	01:33:45	00:58:33
[142]	5388	00:55:51	01:56:50	01:57:44
[143]	1567	00:21:12	00:58:50	00:52:07
vehicle	8463	01:45:40	02:12:40	02:08:47
Gas_Sens	4188	00:42:36	02:47:20	01:16:14
shuttle	57999	04:26:03	05:15:45	04:02:27
APS Failure	60000	05:23:35	03:58:39	04:32:14
CustSat	76020	04:09:36	05:07:03	05:26:35
car	1728	00:40:07	01:38:30	01:04:10
airlines	5473	00:57:52	02:18:27	01:13:12

Table 2.12: Runtime of selected AutoML frameworks on the benchmark datasets.

From a manufacturing practitioner's (e.g. engineer, researcher) perspective, an adapted AutoML decision support system can be very useful as it does not require any machine learning or coding experience. Therefore, manufacturing professionals can use AutoML to identify different insights based on available datasets of imaging, sensors data, production history, etc.

2.2.2 Building AutoML for manufacturing datasets

Despite the advances achieved in the field of AutoML, few works have been conducted to apply these techniques in the manufacturing field and hence, the industrial needs are yet to be fulfilled. As we mentioned before, there are several challenges that tackle the application of machine learning in the manufacturing space. One of the main challenges is the construction of a high quality and representative dataset. Ideally, an ML model should be trained with data that reflects as close as possible the original one. This is not easy as in a manufacturing unit, data are heterogeneous and each individual process generates different amounts of data with various formats (text, images, sensors data, etc.) and with different quality levels.

Besides the difficulty of constructing a high-quality dataset, a much bigger

issue arises, consisting of a lack of transparency regarding the decisions made by AutoML systems making them as black boxes [145]. The lack of transparency leads machine learning experts and novices alike to question the results that were automatically obtained. If users cannot interpret the obtained results, they will not trust the AutoML system they are attempting to use and hence, they will hesitate to implement the model in critical applications, especially in manufacturing fields where interpretation and transparency of algorithms are a must for a system to be adopted into a workflow [146]. Another reason that justifies the low adoption rate of AutoML solutions in the industrial space is that the current methods for the ML pipeline optimization are inefficient on the large datasets originating from the manufacturing environment.

In order to overcome some of these issues, Lechevalier *et al.* [34] proposed a framework for semi-automatic generation of analytical models in manufacturing and a proof-of-concept prototype that allows practitioners to generate artificial neural networks for prediction tasks through a user interface. Similarly, Zacarias *et al.* [147] show a framework that automatically recommends suitable analytics techniques with respect to a domain-specific problem at hand. Both frameworks have represented promising approaches to tackle the problem of automated analytics techniques configuration in the manufacturing domain. However, these frameworks do not achieve the required goal of identifying the promising combinations of analytics and the application areas in the first place. Therefore, they cannot be used as decision-making tools at the managerial level [148].

All of these findings point to one take-away message : intelligent systems are able to automatically design the whole or parts of machine learning pipelines, which can save practitioners considerable amounts of time by *automating* one of the most laborious parts of machine learning pipelines.

2.3 Towards AutoML for industrial big data

By addressing the aforementioned challenges and difficulties, remarkable benefits for rechearchers, engineers and the industrial manufacturing system will be achieved. This will significantly help with the lack of funds for industrial coding in the manufacturing systems [149]. More importantly, significant improvement in production outcomes can be achieved by faster and more accurate diagnoses and prognoses. Although each manufacturing organization has different structure and size of IBD (Industrial Big Data), a potential AutoML tool will find the best algorithm and settings that provide best accuracy without the need for human interference, which will save time and costs by reducing the necessity of highly skilled coders and data analyts in the industry.

In addition, such a tool will assist manufacturing practitioners to better manage their production systems and use their valuable time to deliver better outcomes for processes. It can also contribute to improve industrial resources management. Moreover, it will help practitioners increase manufacturing coding team efficiency, reduce coding errors, improve coding quantity and quality, and assist domain rechearchers through significantly cutting the time needed to trial-and-error process for processing IBD.

2.4 Conclusion

The process of ML algorithms selection and parametrization is a complex and time intensive task as it was already exhibited in Chapter 1. Motivated by the academic dream and industrial needs, the automated machine learning has recently became a hot topic in order to relieve the analysts (either experts or novices) from the ML pipeline building difficulties. This chapter presented a systematic review of existing AutoML approaches and tools with a highlight on MtL as a promising paradigm for the algorithms selection and configuration problem. We first defined what the AutoML problem is, and then introduced the fundamental concept of AutoML and the related tools and techniques. We also provided taxonomies of existing works based on "what" and "how" to automate, which acts as a guidance to design new and use old AutoML approaches. Related studies on MtL were split into three general categories according to the final goal of the MtL system. The main aspects of these studies are presented and discussed. We further review state-of-art AutoML tools and platforms. Finally, we survey the potential of these tools on industrial big data.

Although the number of related works in the last years has been increasing, there are still several aspects that need further investigations. Literature presents some patterns : most of the studies produced meta-knowledge through GS executions with a Holdout or single CV resampling, characterizing datasets using simple and statistical meta-features, and recommending HPs with a kNN meta-model. The reproducibility of the experiments and the sharing of results are two key aspects that have not been explored yet. Additionally, an end-to-end MtL resolver for the CASH problem has not been proposed nor ivestigated. These aspects would benefit the research community with valuable meta-knowledge for further works. Thus, exploring different experimental setups, meta-features, and procedures applied to the learning tasks may open up new horizons in the MtL research area.

Our work falls within a framework that directly uses informations drawn from a dataset, without having to perform extensive experimentations, recom-

mends which ML algorithms and related HPs configuration to use. We took special care in the construction of a modular meta-learning space and the definition of the meta-learning problems that populate it. The dataset characteristics were chosen carefully in an effort to provide a set that can best discriminate among the performance of different inducers; furthermore we proceeded to a systematic experimentation to characterize their discrimination power. We also undertook a systematic experimentation in order to determine the most appropriate inducer for meta-level learning, and compare our set of features with various different approaches to dataset characterization. Finally, we also explore the aspect of explainability of such decision support systems to address the trust-in AutoML. Where our work differs from and where it resembles existing approaches will be clarified in the forthcoming part.

Part II

Contributions

Chapter 3

Towards the automation of industrial data science : A MtL based approach

Outline of the current chapter

Advanced analytics are fundamental to transform large manufacturing data into resourceful knowledge for various purposes. In its very nature, such "industrial big data" can relay its usefulness to reach further utilitarian applications. In this context, machine learning is among the major predictive modeling approaches

that can enable manufacturing researchers and practitioners to improve the product quality and achieve resources efficiency by exploiting large amounts of data (which is collected during manufacturing process). However, disposing ML algorithms is a challenging task for manufacturing industrial actors due to the prior specification of one or more algorithms hyperparameters and their values (cf. Chapter 1). Moreover, manufacturing industrial actors often lack the technical expertise to apply advanced analytics. Consequently, it necessitates frequent consultations with data scientists; but such collaborations tends to cost the delays, which can generate the risks such as human-resource bottlenecks. As the complexity of these tasks increases, so does the demand for support solutions.

In this regard, existing AutoML solutions include evolutionary algorithms, bayesian optimization, and reinforcement learning. These approaches mainly focus on providing the user assistance by automating the partial or entire data analysis process, but they provide very limited details concerning their impact on the analysis. The major goal of these conventional approaches has been generally focused on the performance factors, while the other important and even crucial aspects such as computational complexity are rather omitted (cf. Chapter 2). Therefore, in this chapter, we show that the combined algorithms selction and parametrization problem can be addressed with the help of meta-learning, overcoming the majority of the challenges stated by the related AutoML solutions. We present a novel meta-learning based approach to automate ML predictive models build over the industrial big data. The approach is leveraged with development of, AMLBID, an Automated ML tool for Big Industrial Data. It attempts to support the manufacturing engineers and researchers who presumably have meager skills to carry out the advanced data analytics in order to conduct the ML techniques for manufacturing problems.

3.1 Introduction

The fourth industrial revolution or Industry 4.0 is increasingly relying on machine learning based solutions [144]. This is particularly stimulated by the availability of large datasets concerning various real-world features [146] and also through the increase of the computational gains which are generally attributed to the powerful GPU cards [150]. The availability of such data combined with the knowledge of manufacturing experts may be an opportunity to build AI based processes and models providing high value insights and assets for decision makers [32]. For example, ML algorithms have been applied with great success at the process, machine, shop floor and supply chain levels, and have proven

effectiveness in the maintenance field by predicting the occurrence and severity of machinery failures [151–153]. Recently, a predictive model [61] based on machine learning has been used to estimate and predict the gradual degradation of such machinery, allowing the operators to make informed decisions regarding maintenance operations. These results, among others, show a heavy interest in ML development and analysis for manufacturing applications.

As machine learning has proven its benefits and efficiency in many fields, its successful implementation in the context of manufacturing industry requires a large effort from human experts and practitioners since there is no *one size fits all* algorithm that can perform well on all possible problems [63]. A data-mining algorithm may perform differently on datasets with different characteristics, e.g., it might perform better on a dataset with continuous attributes rather than with categorical ones, or the other way around. Typically, a learning algorithm needs to be tuned before being mined, taking into account all the possible hyperparameters types, dependencies and values (cf. Chapter 1). However, building such processes and models requires AI and data science skills and expertise that are not always available in the manufacturing area workbenches and laboratories.

Expert users have the required knowledge to find the right data-mining pipeline. However, when it comes to non-experts, even though being familiar with manufacturing data, they are overwhelmed by the amount of ML algorithms and it is challenging for them to find the HPs configurations that would positively impact their analysis[36]. Existing support solutions either assume that users have expert knowledge, or they recommend ML pipelines that are only "syntactically" applicable to a dataset, without taking into account their impact on the final analysis. To overcome such a lack, they often cooperates with data science experts. Nevertheless, for this interactive process to converge, a lot of effort and time is required from both sides. This is due to the fact that devising and deploying ML solutions often needs to be started from scratch. This long journey has to start from a lengthy data provisioning process. It continues with finding the right collaborators which requires a continuous back-and-forth exchange between ML experts and industrial actors. Hence, automating activities often require human expertise that would allow smart factories actors and researchers to rapidly build, validate, and deploy ML solutions [34, 53].

Motivated by this goal, Meta-Learning opportunities present themselves in many different ways, and can be embraced using a wide spectrum of learning techniques. Every time we try to learn a certain task, whether successful or not, we gain useful experience that we can leverage to learn new ones. We should never have to start entirely from scratch. Instead, we should systematically collect our "learning exhaust" and learn from it to build automated ML systems that continuously improve over time, helping us tackle new learning problems

ever more efficiently. The more new tasks we encounter, and the more similar those new tasks are, the more we can tap into prior experience, to the point that most of the required learning has already been done beforehand. The ability of computer systems to store virtually infinite amounts of prior learning experiences (in the form of meta-data) opens up a wide range of opportunities to smart factories stakeholders to use that experience in completely new ways. We are then only starting to learn how to learn from prior experience effectively.

In this chapter, we aim at providing assistance to non-expert users by recommending ML pipelines that are ranked according to their impact on the final analysis. In order to do that, we make use of the concepts of *meta-learning* to automate ML predictive models build over the industrial big data. The approach is leveraged with development of, AMLBID, an Automated ML tool for Big Industrial Data. Given a dataset, and an evaluation metric (e.g., predictive accuracy, recall, F1 score), AMLBID produces a ranked list of all candidate pipelines based on their expected performance with respect to the desired metrics. This list is produced based on a meta-knowledge base gained from previously analyzed manufacturing datasets and combinations of pipelines, without executing individual candidate pipelines. As a result, AMLBID has a computational complexity near O(1). Therefore, the proposed solution may improve their quality of service, productivity, and more importantly, reduce the need for ML human experts.

Contributions. The main contributions of this chapter can be summarized as follows :

- We leverage ideas from meta-learning to present a technique for recommanding the optimal or near optimal ML pipelines depending on their impact on the final result of data analysis and the desired evaluation metric.
- We show the benefits of our approach by implementing a prototype that is capable of automatically recommending ML algorithms along with related hperparameters configurations to the user.
- We perform an empirical study comparing the ability of state-of-the-art solutions against our approach, on finding the well performing ML algorithms(s) for a given dataset using a desired performance measure.

The rest of this chapter is organized as follows : an overview on MtL for the automatic algorithms selection and configuration is given in Section 3.2. Our proposed solution is formally defined in Section 3.3. A brief look at the materialization of our proposed approach in terms of a prototype solution is given in Section 3.4. The results of the experimental evaluations are reported in Section 3.5. Finally, Section 3.6 summarizes the work shown in this chapter.

3.2 Meta-learning for combined algorithms selection and configuration

The AutoML problem we consider is to generate the optimal or near optimal pipeline $\mathcal{P}(A_H) : x \mapsto y$, induced by an algorithm $A \in \mathcal{A}$ and parametrized by $H \in \mathcal{H}$ that automatically produces predictions for samples from the distribution of a given task, while minimizing or maximizing the generalization metric $\mathcal{L}$:

$$A_H \in \underset{A \in \mathcal{A}, H \in \mathcal{H}}{argmin}\, \mathcal{L}(A_H(x_i), y_i) \tag{3.1}$$

In practice we have access to two disjoint, finite samples which we denote as D_{train} and $D_{validation}$. For searching the best ML pipeline, we only have access to D_{train}, however, the end generalization performance is estimated on $D_{validation}$,e.g., by a K-fold cross-validation:

$$\mathcal{L}_{CV}(A_H, \mathcal{D}) = \frac{1}{K} \sum_{k=1}^{k} \mathcal{L}(A_H^{\mathcal{D}_{train}^{(train,k)}}, \mathcal{D}_{valid}^{(valid,k)}) \tag{3.2}$$

where $\mathcal{D}_{train}^{(train,k)}$ denotes that A_H was trained on the training splits of k-th fold $\mathcal{D}_{train}^{(train,k)} \subset \mathcal{D}_{train}$, and it is then evaluated on the validation splits $\mathcal{D}_{valid}^{(valid,k)}$.

Having set up the problem statement, we can use this to further formalize the goals. Instead of using random populations of pipelines and tune related hyperparameters, e.g., by GA or BO, we will introduce an optimization policy that aims at learning relationships between datasets characteristics and data mining algorithms [154]. Given the characteristics of a dataset, a predictive meta-model can be used to forcast the performance of a given combination of A and H on a dataset D.

First, a *meta-learning space* from which to learn is established using meta-data. The meta-data consist of datasets characteristics along with performance measures of data mining algorithms on those particular datasets. Then, the *meta-learning phase* generates predictive meta-model that defines the area of competence of the data mining algorithms [4]. Finally, when a dataset arrives, the dataset characteristics are extracted and fed to the predictive meta-model, which predicts the potential well performing ML pipelines on the considered dataset. At this point, by comparing the obtained predictions for the different pipelines on simillar tasks, we are able to rank the pipelines depending on their predicted impact on the given dataset. This concludes the *recommending* phase.

3.3 Conceptual description

The global architecture of the proposed framework is depicted in Figure 3.1. The algorithms recommendation and parametrization is provided by the *Suggestion Engine* through the use of a knowledge base (KB) which is an inherent part of the system. The knowledge base is simply a collection of inductive meta-features that describe the datasets, the pipelines and their interdependency. When a new dataset is presented to the system, the suggestion engine provides a recommendation of the most appropriate classifiers. This is achieved by combining the pipelines of the knowledge base with the morphological characteristics originating from the meta-model.

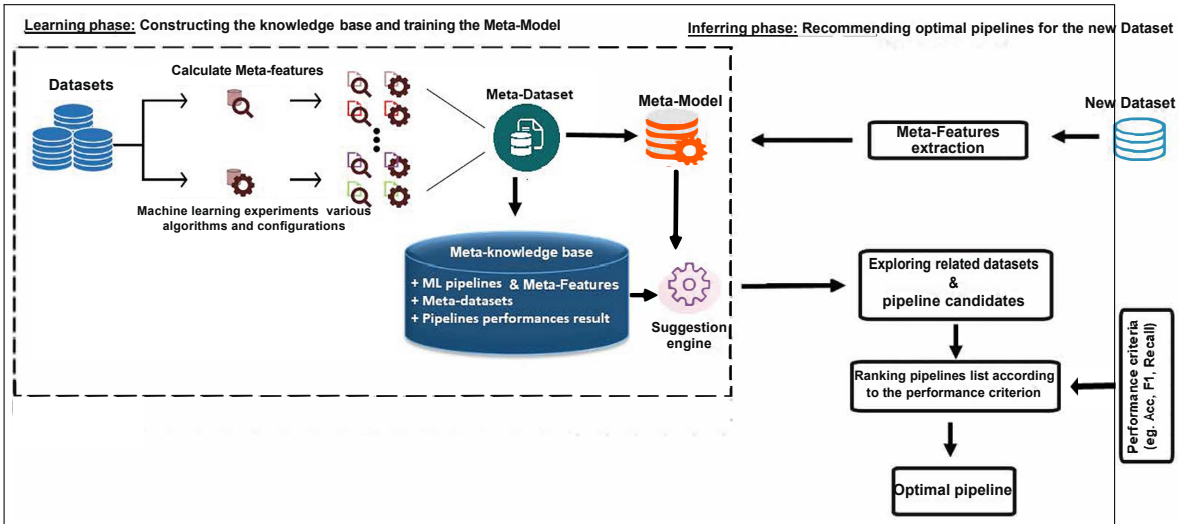

Figure 3.1: The functional architecture and process flowchart of AMLBID.

AMLBID has been developed on the meta-learning concept, consequently, it consists of two main phases which are the *learning* phase and the *inferring* one. During the learning phase, we evaluate different classification algorithms with multiple hyperparameters configurations on a large collection of various datasets, analyze the learning datasets (to extract meta-features), and train a ranking meta-model. During the inference phase, the meta-model generated in the training phase is used to produce a ranked list of promising ML pipelines for a new dataset and a classification performance metric.

3.3.1 Learning phase

Two important activities are performed in the learning phase. First, a meta-knowledge base (i.e., set of meta-datasets) is generated for all the considered performance measures (cf. Algorithm 5), and then on top of it, a learning algorithm is applied (cf. Algorithm 6). As a result, a statistical model (meta-model) is generated for every considered perfomance measure. The inputs required to construct the meta-knowledge base are datasets, classification algorithms and the hyperparameters values that are likely to improve the performance of the considered classification algorithms.

For the sake of simplicity, let us consider that we want to create the meta-dataset for a predictive metric (e.g., accuracy). In line 7 of Algorithm 5, we first extract the datasets characteristics (i.e., meta-features). Next, we apply the classification algorithms with all possible and resonable HPs configurations and then take the corresponding performance measures (e.g., predictive accuracy) — line 10. The latter is the meta-response, which together with the meta-features of the datasets compile the complete set of meta-data — see line 11.

Algorithm 5 Establish the meta-knowledge base.

```
1:  Input: ClassificationAlgs[..],          ▷ available classification algorithms
    HpSpace[..],                            ▷ set of HPs configurations to be applied
    PerfMeasures[..]                        ▷ set of performance measures to acquire
2:  Output: meta_KB[#measure][#metadata]    ▷ meta-knowledge base
3:  function CreateMetaKB(datasets[ ])
4:      metadata[ ] = ∅
5:      for each measure in PerfMeasures do
6:          for each dataset DS in datasets do
7:              ds_mf = ComputeMetaFeatures(DS);          ▷ See Table 3.3
8:              for each algorithm Alg in ClassAlgs do
9:                  for each hyperparameters_configuration Hp in HpSpace do
10:                     ds_pm = GetPerformanceWith5FoldCV(Alg, Hp, DS);
11:                     metadata[ ] ← ds_mf ∪ ds_pm;
12:         meta_ds[measure] ← metadata[ ];
13:     return meta_KB
```

Once a meta-dataset for each performance measure is obtained, next, a meta-learner algorithm is applied on top — line 5 of Algorithm 6, and as a result, a meta-model for each of the performance measures is obtained. We used the KNN and Random Forest algorithms as meta-learners. Better results were obtained using the KNN meta-model (the comparative study is shown in next section).

Algorithm 6 Create meta-models.

```
1: Input: meta_KB[..][..],                                  ▷ See Algorithm 5
   PerfMeasures[..]                          ▷ set of available performance measures
2: Output: meta_models[..]          ▷ meta-model for each performance measure
3: function PerformMetaLearning( )
4:     meta_models[ ] = ∅
5:     meta_model = KNN();                                  ▷ a meta-model of choice
6:     for each measure in PerfMeasures do
7:         meta_models[measure] ← ApplyMetaModel(meta_KB[measure][ ]);
8:     return meta_models
```

3.3.2 Recommendation phase

The recommendation phase is initiated when a new dataset to be analyzed arrives. At this point, the user selects a performance measure to be used for the analysis and the system automatically recommends the ML algorithms along with related HPs configurations to be applied, such that the final result is optimal. This phase is described in Algorithm 7, where, first, the meta-features are extracted from the dataset in lines 5. Next, the extracted features are then fed to the predictor (meta-model) in line 6. The predictor in line 6 apply an already existing meta-model to the extracted features, to find the the optimal or near optimal pipelines for the given dataset. After, the ranked list of the potential well performing pipelines is obtained for the given dataset and desired performance measure in line 7.

Algorithm 7 Recommend ML pipelines.

```
1: Input: Dataset[..],                                  ▷ new dataset chosen by the user
   meta_models[..]                          ▷ meta-model for each performance measure
2: Output: MLpipelines[..]                  ▷ ML pipelines ranked according to the PM
3: function RecommandPipelines(Dataset[ ])
4:     recommendations[ ] = ∅
5:     ds_mf = ComputeMetaFeatures(Dataset);
6:     PotentialPipelines[ ] ← ApplyMetaModel(ds_mf, meta_models[measure])
7:     recommendations[ ] ← Rank(PotentialPipelines, desc = True)
8:     return recommendations
```

For the sake of concreteness, let us assume that, the user wants to perform predictive analytics to a dataset using the *Recall* performance measure, to deal with a classification problem at hand. Our system, first, extracts the necessary meta-features from the dataset and uses them as input to the predictive meta-

model which is specifically built for the Recall performance measure. The meta-model is built by training a meta-learner (e.g., KNN) on historical meta-data consisting of dataset characteristics and a performance measure (i.e., Recall) of multiple ML pipelines on the datasets. This meta-model is used to produce a prediction for provided dataset.

3.4 Prototypical implementation

In this section, we discuss the materialization of the proposed approach into a prototype solution. In the previous sections we mentioned that in order to build a predictive meta-model, we must firstly establish the meta-space — denoted as *learning phase* in Figure 3.1. In our context, the meta-space needs to be constructed out of meta-data that can be extracted from datasets and from the executions of classification algorithms on these datasets. As a matter of fact, we needed to fetch hundreds of datasets, extract their characteristics, run different algorithms on them and get different evaluation measures. Finally, we use all of these to feed the meta-knowledge base. In the following sections, we discuss in detail the used datasets and their meta-features along with the performance evaluation and the meta-knowledge base construction.

3.4.1 Datasets

In our study, we used 400 real-world manufacturing classification datasets that have been collected from the popular Kaggle[1], KEEL[2], UCI[3], and OpenML [155] platforms. These datasets represent a mix of binary (71%) and multiclass (29%) classification tasks, which are highly diverse in terms of dimensionality, and class imbalance. The datasets characteristics are indicated in the Appendix C, and summarized in the following Tables 3.1 and 3.2.

	Classes	Attributes	Instances
Min	2	3	185
Max	18	71	494051

Table 3.1: Statistics about the used datasets according to the number of classes, predictive attributes and instances.

[1] https://www.kaggle.com/
[2] https://sci2s.ugr.es/keel/datasets.php
[3] https://archive.ics.uci.edu/

It is worth noting that the used datasets cover a broad range of application areas, including process level studies, machine related problems and the supply chain level, among others (see Table 3.2). In order to ensure the fairness in our performance comparison, we have not performed any preprocessing operation on the datasets to avoid any potential bias or impact on the classifiers performances.

Task	Number of datasets	Average of		
		Attributes	Instances	Classes
Process level	78	29	30529	3
Machine level	248	53	13942	2
Supply chain level	74	17	21726	2

Table 3.2: Statistics about the used datasets according to related tasks.

3.4.2 Meta-features

A meta-feature, also considered as a characterization measure, is a function that extracts relevant characteristics from a dataset to characterize its complexity. The description of a dataset by a set of meta-features produces a numerical values vector. During this thesis, we considered 41 meta-features extracted from the training datasets using the PyMFE[4] tool [156] for the *Simple*, *Statistical*, *Information-theoretic*, *Model-based*, *Landmarking*, and *Data complexity* measures. The meta-features we specifically consider are highlighted in Table 3.3 and detailed in Table B.1 in the Appendix B.

3.4.3 Meta-knowledge base

The pipelines generation

We used 08 classifiers from the popular Python-based machine learning library, Scikit-learn[5] in order to build the meta-knowledge base. These classifiers are *Support Vector Machines*, *Logistic Regression*, *Decision Tree*, *Random Forest*, *Extra Trees*, *Gradient Boosting*, *AdaBoost*, *and Stochastic Gradient Descent* classifier. Detailed description of the algorithms and their tuned hyperparameters are described on the Tables A.1-A.7 in the Appendix A.

[4] https://pypi.org/project/pymfe/
[5] https://scikit-learn.org

Type	Dataset characterization measures (Meta-features)
	1. Simple, Statistical & Information Theoretic
Simple	Number of instances, Number of Attributes, Number of target concept values, Proportion of minority target, Proportion of majority target, Proportion of binary attributes, Proportion of nominal attributes, Proportion of numeric attributes Proportion of instances with missing values, Proportion of missing values, Geometric mean, Harmonic mean,
Statistical	Kurtosis of data based on numerical attributes, Maximum eigenvalue Skewness of data based on numerical attributes, Covariance.
Info theo	Class entropy, Uncertainty coeff.
	2. Model Based Measures
Model Based	Height of tree, width of tree, Number of nodes in tree,number of leaves in tree, maximum number of nodes at one level, mean of the number of nodes on levels, length of the longest branch, Model Based length of the shortest branch, Mean of the branch lengths, Standard deviation of the branch lengths, Minimum occurrence of attributes, Maximum occurrence of attributes, Mean of the number of occurrences of attributes, Standard deviation of the number of occurrences of attributes
	3. Landmarking Based Measures
Landmarking	Naive Bayes, ii) 1-NN (Nearest Neighbor), iii) Elite 1-NN, iv) decision Tree learner and v) a random chosen node learner.
	4. Complexity Based Measures
Dimentionality	Average number of points per dimension, Ratio of the PCA dimension to the original dimension
Class balance	Entropy of classes proportions, Imbalance ratio

Table 3.3: A sample list of meta-features used in current thesis.

The knowledge base consists of the accumulated experience of previous optimizations of the classifiers on the datasets. It consists of their meta-features $m_1, ..., m_{400}$ and the optimal hyperparameters values of the classifier found for each dataset : $\mathcal{K}_B = \{(m_1, A_{H^1}^{(1)}), \ldots, ()m_{400}, A_{H^n}^{(n)})\}$.

To better understand the construction of the knowledge base, let us consider the execution scenario of the proposed framework. We actually generate at least 1000 different combinations of the hyperparameters configurations for every single execution of an algorithm $\boldsymbol{A}$ over each dataset $\boldsymbol{D}$. This execution process results in an average of 8000 pipelines for each dataset. It might be useful to note that during the construction of the meta-datasets, we performed a 10×5-fold stratified cross-validation strategy for estimating the pipelines performance, in order to get stable performance. i.e., for each candidate pipeline applied on every

problem (dataset), the 05 fold stratified cross-validation is repeated 10 times by randomizing the order of instances. This process controls the variation imputed by different choices of training and test instances [62]. As a result, the knowledge base consists of more than *4 millions* evaluated classification pipelines. It can be observed the number of configurations/evaluations of any considered algorithm is not the same due to the different variations of algorithms hyperparameters. Finally, for each classification algorithm and for each performance measure, we obtained a meta-dataset that was fed to the Meta-knowledge base.

Additionally, since the presented algorithm 7 calculates the meta-features of a dataset and returns the optimized hyperparameters values H_i of the most adequat algorithm to use, this experience is added to the collaborative knowledge base $\mathcal{K}_{B_{new}} = \mathcal{K}_B \cup (m_{new}, A^{(i)}_{H^i})$. This makes the propsed system smarter by attaining more experience, based on the growing knowledge base that is continuously improved over time by running more tasks.

The measures

As part of our core idea, we aim to recommend high-performing ML pipelines for a given combination of datasets and evaluation measure. The point that most of state-of-the-art systems do not take into account, the proposed system supports various classification performance measures to evaluate the performance of the ML pipelines (ML algorithms and related hyperparameters configuration). Table 3.4 shows supported measures details.

Measure	Description	Importance
Precision	Precision considered as a measure of exactness or quality.	Precision is used to retrieve fraction of instances that are relevant.
Recall	Recall is a measure of completeness or quantity.	Recall is used to retrieve fraction of relevant instances that are retrieved.
Accuracy	The accuracy is the proportion of the total number of predictions that were correct. Accuracy is related to the degree of bias in the measurements	Accuracy is used to represent the correct answer or percentage of accurate classification.
F1 score	F1 score or F-measure is defined as the harmonic mean of precision and recall. Commonly used as a single metric to evaluate the classifier performance.	A value closer to one implies that a better combined precision and recall is achieved by the classifier

Table 3.4: Supported classification measures.

The meta-knowledge base schema

The meta-knowledge base ERD schema is illustrated in Figure 3.2. The Knowledge base is used to store the results of the experiments and allows easy access to any operation and data during the inferring phase. As the KB is going to be used mainly for storing and extracting data with no need for any high complexity queries and because by definition, each dataset, pipeline, and experiments can be different from one another, we choose to implement a NoSQL database structure. Mainly we used MongoDB[6]. The schema contains the following entities, and it is used as a base for our KB :

- **Datasets.** The datasets are associated with the metadata of their meta-features and a learning job.
- **Learning jobs.** learning job is defined as the combination of a dataset and a metric. It can be treated as the input for a given ML pipeline.
- **Pipelines.** storing pipelines that were generated beforehand while conducting the experiments.
- **Hyperparameters.** The hyperparameters values are determined only when associated with an algorithm, by the Pipeline entity.
- **Experiments.** results of running a pipeline on a learning job. The results are the pipeline scores on the defined learning job's metrics, the runtime, and the RAM usage.

After obtaining the meta-data, hence constructing the meta-space, the meta-knowledge base is first transformed into a learning dataset. Therefore a meta-model can be trained on it to model the relationship between datasets characteristics and performance information of candidate algorithms.

3.4.4 The Meta-model

Having stored the algorithms performances (see Table 3.5) and a set of datasets characteristics (see Table 3.6), the goal is to build a mapping *meta-model*. The mapping meta-model is intended to learn the complex relationship between a task meta-features and the utility of specific ML pipelines, to recommend the most useful ML algorithm(s) configuration according to the meta-features M of a new task t_{new}.

[6] `https://www.mongodb.com/`

Figure 3.2: The ERD schema of the knowledge base.

	$Classifier_1$	$Classifier_2$	$\cdots$	$Classifier_{8000}$
$Dataset_1$	0.85	0.91	$\cdots$	0.48
$Dataset_2$	0.91	0.68	$\cdots$	0.94
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$Dataset_{400}$	0.94	0.86	$\cdots$	0.75

Table 3.5: Performance of classification algorithms on various datasets.

	$No.instances$	$No.attributes$	$\cdots$	$Meta-feature_{41}$
$Dataset_1$	2000	16	$\cdots$	0.73
$Dataset_2$	1340	11	$\cdots$	0.89
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$Dataset_{400}$	61598	17	$\cdots$	0.81

Table 3.6: Meta-features of the datasets.

Formally, each task (dataset) $t_j \in \mathcal{T}$ is described by a vector $F(t_j) = (m_{j,1},\ldots,m_{j,K})$ of K meta-features $m_{j,K} \in F$, the set of all known meta-features.

This can be used to define the task similarity measure based on, for instance, the *Euclidean distance* between $m(t_{new})$ and $m(t_j)$, so that we can transfer information from the most similar tasks to the new task t_{new}. The distance measured between meta-features of t_{new} and t_j is given by Eq. (3.3) :

$$d\left(m(t_{new}), m(t_j)\right) = \sqrt{\sum_{i=1}^{k}\left(m_{t_{new,i}} - m_{t_{j,i}}\right)^2} \tag{3.3}$$

One of the aims of our work is to produce an enriched meta-model able to recommend the top-performing classification configuration(s) for a combination of an unseen dataset and a classification evaluation measure. For this purpose, a few basic criteria were followed for selecting the meta-learner to use. First, the problem in the meta-learning space is of classification type. A class needs to be predicted (a ML pipeline has the potential be be among the well performing pipelines or not). The second criterion is that the meta-model needs to be more *sensitive*. By this, we mean that the meta-model needs to be able to capture even the slight meta-features that characterize the datasets. This is because we need to predict the impact of the HPs values on the data mining results and we need to be able to learn the correlation between the datasets meta-features, the types and configurations of differents ML alogrithms. The third criterion is that the meta-learner should handle missing values. Recall that some dataset characteristics (meta-features) can be calculated on datasets that necessarily contain either continuous or categorical attributes. As a matter of fact, two state of the art learning algorithms were chosen to produce meta-models able to predict the most appropriate pipelines for the dataset at hand : Random Forest (RF) and k-Nearest Neighbor (kNN).

The KNN algorithm has proven to be generally effective, often referred to as "*nearest neighbor imputation*" as well for the RF learner that complies with all the above mentioned criteria. It suffers far less from the discreteness of the leaves, because internally, a lot of trees (i.e., 500 trees) are built at random and at the end, averages are taken to be used as predictions. Finally, it performs well when missing values are present. Thus, we use the KNN and RF to build meta-models for each data mining performance measure we consider.

Ranking using the KNN classifier is a commonly used strategy to obtain the top-K rankings. When a new dataset is presented to the meta-learning system, the KNN identify the *k-nearest datasets* (KND) of the candidate dataset in the meta-knowledge base, using the *Euclidean distance* measure (Eq. (3.3)). Based on this measure, a vector $d = [d_1, d_2, \ldots, d_k]$ containing the dissimilarity among all characteristics (meta-features) of datasets is built and a weighted average of each individual neighbor is used for forecasting the optimal pipeline configuration

based on the relevant measure. The k-nearest datasets selection approach is shown as pseudo code in Algorithm 8.

Algorithm 8 K-nearest datasets selection.

```
1: Input: Ds[..],                          ▷ new dataset chosen by the user
   meta_KB[..]                             ▷ the constructed knowledge base
2: Output: KND[..]                         ▷ K-nearest datasets distance vector
3: function KND_Selection(Ds[ ])
4:     KND = ∅
5:     ds_mf = ComputeMetaFeatures(Ds);
6:     for each dataset d in meta_KB do
7:         d ← sqrt(Σ_{i=1}^{k} (m_DS − m_d)^2)
8:         KND ← d
9:     return K-nearest datasets distance vector
```

While for the Random forest meta-model, we produce for each supported classification evaluation measure a large labeled training set using the following process :

1. For each combination of $d \in \mathcal{D}$ and $A^{(i)} \in \mathcal{A}$, where $\mathcal{D}$ is the 400-learning datasets, $A^{(i)}$ a learning algorithm configuration from the 4 millions evaluated configurations, we retrieve the set of all best predictive results $R(d, E_i)$ for each evaluation metric E_i (e.g., accuracy, F1-score, recall and precision).

2. For each $d \in \mathcal{D}$ we designate the learner algorithm configuration $A^{(i)}$ as *Class 1* (top performer algorithm configuration for the dataset) if its best predictive results for the dataset are greater than or equal to the highest performance achieved by all other configurations. Otherwise we label the $A^{(i)}$ for the dataset as *Class 0* (low performer algorithm configuration).

3. For each combination of $d \in \mathcal{D}$ and $A^{(i)} \in \mathcal{A}$ we generate a joint set $F = \{F_d \cup F_{A^{(i)}}\}$, where :

   - F_d : the dataset's meta-features generated in the learning step.
   - $F_{A^{(i)}}$: a discrete feature describing the learning configuration $A^{(i)}$.

4. The joined meta-features vectors F are used to fit the RF meta-model for the top performing algorithms configurations, using the meta-features variables as predictors and the learner's labels as targets of the meta-model. For our Meta-Model, we have been mainly interested in optimizing the prediction *recall* of Class 1 (the classifier has the potential to be among

the best performing classifiers). Therefore, we had to consider different levels of the decision tree model hyperparameters configuration where the configuration: $\{class\_weight : \{1 : 1, 0 : 0.7\}, criterion : gini, max\_features : None\}$ provided the best meta-model result.

The main functionality of the meta-model can be formally defined as follows : given a set of learning algorithms space $\mathcal{A} = \{A_1^{(1)}, \ldots, A_n^{(i)}\}$ where i is the i-th hyperparameters configuration of A, a dataset D divided into disjoint training D_{train}, and validation $D_{validation}$ sets, and an evaluation measure E, the goal is to identify the ML algorithm(s) $A^{(i)*}$, where $A^{(i)*} \in \mathcal{A}$ and $A^{(i)*}$ is a tuned version of $A^{(i)}$ that minimizes or maximizes the E on D.

We used the Python programming language to construct a model for each one of the considered performance measures. After that, the models were exported to PKL files, and were next fed to the *Suggestion engine* in the inferring phase.

3.5 Empirical study

In the following sections, we describe the empirical study of the performance achieved by the experiments with AMLBID on various manufacturing datasets. Following the eventual experimental configuration, we demonstrate the ability of AMLBID to effectively search the generated enormous hyperparameters space to find the optimal algorithms and hyperparameters with a low computational complexity. Finally, this leads us to a comparative evaluation of the performance of AMLBID tool and the currently available state-of-the-art (i.e., the TPOT [157] and Autosklearn [73]) ML pipelines generation tools.

3.5.1 The experimental configuration

To ensure meaningful comparison, we benchmark on a highly varied selection of 30 datasets that cover both binary and multiclass classification problems from the perspectives of different industry 4.0 levels. These data are gathered broadly from two major sources :

- **OpenML AutoML benchmark** : Datasets curated to serve as representative benchmark for AutoML frameworks [135]. These datasets span various binary and multi-class classification problems and exhibit substantial heterogeneity in sample size and dimensionality.
- **State-of-the-art-papers**: Datasets collected from research papers dealing mainly with industry 4.0 related problems using ML based solutions.

These 30 fresh datasets were not previously exploited by any learning method during the offline phase in our framework. These are introduced to AMLBID to evaluate the pipeline recommendations.

Dataset	Num Class.	Num Inst.	Task
[136]	4	959	Failure risk analysis
[137]	3	2000	Chatter prediction
[138]	2	61000	RUL prediction
APSFailure	2	60000	APS system failure prediction
Higgs	2	110000	predictive maintenance
CustSat	2	76020	Customer Satisfaction

Table 3.7: List (sample) of datasets used in the evaluation.

The evaluation method

The recommended ML pipelines were trained on the benchmark datasets. Subsequently, the performances of AMLBID were compared to those of the TPOT and Auto-sklearn frameworks and also to the results of the related research papers (that served as the source of original data). For TPOT, we used the default settings (i.e. generation and evaluation of 100 pipelines for each dataset). While for Auto-sklearn, we compared AMLBID with two versions, as the Auto-sklearn has a "Vanilla" version that produces a single optimal pipline (Auto-sklearn(V)) and the other version that creates a set of 50 best pipelines (Auto-sklearn(E)).

To avoid hardware-dependent performance differences, we ran all AutoML systems on our local hardware (Intel(R) Core(TM) i9-10900KF CPU @ 3.70GHz - 32Go RAM). We used the pre-defined setting, which divides each dataset into 5 stratified folds and runs each tool on 10 CPU cores to produce a final pipeline.

3.5.2 Experimental results

We refer to the table 3.8 to consult the comparative evaluation of the results obtained by exposing the 30 datasets to the AMLBID, TPOT, and both versions of Auto-sklearn. It can be observed that AMLBID performances are comparatively better than those of the baseline even though any pipeline on the dataset was not executed, prior to the recommendation. We can also observe and position the results obtained by AMLBID as more accurate than the results obtained by the TPOT, Auto-sklearn, and the related research papers on the same datasets in table 3.9.

System	>	<
AMLBID	**19**	2
TPOT	6	5
Auto-sklearn	5	23

Table 3.8: Performance of AutoML systems on the 30-benchmark datasets. We count how many times the system performs better (>) or worse (<).

Dataset	AMLBID	TPOT	Auto-sklearn(V)	Auto-sklearn(E)	Original paper result
[136]	**0.9374**	0.9120	0.8215	0.9283	0.85
[137]	**0.9706**	0.9517	0.9632	0.9356	0.95
[138]	**0.9941**	0.9907	0.9782	0.99	0.9895
[139]	0.9205	**0.9991**	0.9357	0.6863	0.9984
[140]	0.8971	0.6711	0.908	**0.9723**	0.9677
[141]	**0.9706**	0.7767	0.678	0.9843	0.9278
[142]	**0.8967**	0.8899	0.6783	0.7952	0.884
[143]	**0.8748**	0.7826	0.6702	0.7727	0.8659
Wafer-ds	**0.8571**	0.7312	0.8033	0.8953	-
HTRU	0.7841	0.7923	0.8038	**0.8134**	-
vehicle	0.8880	0.8415	**0.9027**	0.6591	-
Cnae-9	**0.9671**	0.8803	0.7922	0.8365	-
Gas_Sens	0.9739	**0.9843**	0.9256	0.9468	-
Covertype	**0.8344**	0.7307	0.7890	0.6521	-
Kc1	**0.8793**	0.7097	0.7697	0.8552	-
jannis	0.6719	**0.7229**	0.6171	0.6845	-
MiniBooNE	**0.9645**	0.9423	0.8343	0.8903	-
KDDCup	**0.9740**	0.8934	0.9331	0.95	-
segment	**0.9735**	0.9681	0.9337	0.9542	-
Higgs	0.713	0.726	0.7135	**0.729**	-
Credi-g	**0.7921**	0.7188	0.5739	0.6121	-
shuttle	0.9649	**0.9905**	0.8429	0.9362	-
APS Failure	0.9910	**0.9933**	0.9716	0.984	-
nomao	**0.9708**	0.9570	0.6995	0.7987	-
CustSat	**85.59**	0.8276	0.8072	0.8290	-
kr-vs-kp	**0.9976**	0.9209	0.6532	0.7593	-
car	0.9754	**0.9999**	0.8549	0.9462	-
albert	**0.8759**	0.8005	0.8288	0.7981	-
airlines	0.6982	0.6758	**0.7094**	0.5927	-
Numerai28.6	**0.5207**	0.4229	0.4836	0.4433	-

Table 3.9: Comparative performance analysis of AMLBID and the baseline AutoML tools on the benchmark datasets.

As shown in table 3.9 and illustrated in Figure 3.3, the results of some machine learning solutions, oriented from manufacturing industry, can be improved simply through the use of better ML models and related hyperparameters configuration. It can be useful to note that evaluating only the Top-1 recommended pipeline makes AMLBID more efficient than Auto-sklearn and TPOT. As all state of the art solutions support only the accuracy measure, a comparative study on other important performance measures such Recall and F1-score could not be done.

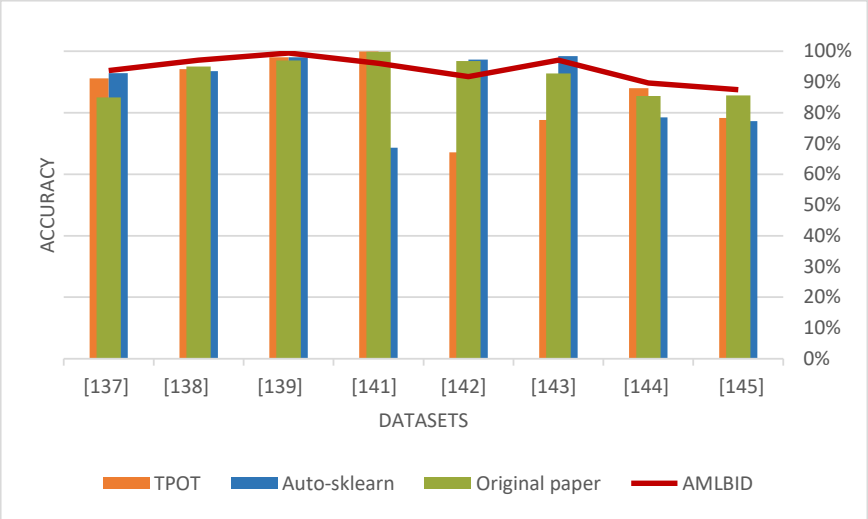

Figure 3.3: The cumulative gains chart of AMLBID and the baseline AutoML tools over the state of the art datasets.

In most of the state-of-the-art AutoML systems, one of the shortcoming is their computational complexity. Oftenly, they require huge time and resources budget on non-conventional datasets. On the contrary, AMLBID has the advantage of the O(1) computational complexity, generating the recommendation in significantly a negligible amount of time. This argument can be further testified by the proven results shown in Table 3.10, which presents the performance of AMLBID, TPOT and Autosklearn in terms of execution time on the same machine for the benchmarked datasets.

The landscaped performance difference of AMLBID is accomplished because the other AutoML systems consume massive time to train the multiple algorithms with various configurations on the same dataset to produce the recommendation.These require to train the ML model from scratch for the fresh datasets prior to generate the list of recommendation configurations. Whilst, as discussed earlier in section 3.4.3, the AMLBID meta-knowledge base is equipped

Dataset	Dataset size	AMLBID	Autosklearn	TPOT
[136]	959	**00:00:05**	01:23:47	00:08:14
[137]	2000	**00:00:12**	01:49:21	00:13:57
[138]	61000	**00:05:29**	04:19:05	03:42:09
[139]	274627	**00:11:43**	08:19:37	06:09:51
[140]	5000	**00:01:27**	02:31:07	01:38:36
[141]	1567	**00:00:53**	01:33:45	00:19:47
[142]	5388	**00:00:57**	01:56:50	00:55:51
[143]	1567	**00:00:33**	00:58:50	00:21:12
Wafer-ds	7306	**00:02:17**	03:44:26	01:42:21
HTRU	54641	**00:06:59**	03:42:09	02:57:11
vehicle	8463	**00:02:28**	02:12:40	01:45:40
Cnae-9	63260	**00:05:47**	04:07:39	03:24:52
Gas_Sens	4188	**00:01:14**	02:47:20	00:42:36
Covertype	25524	**00:03:04**	01:28:31	01:36:14
Kc1	2108	**00:00:38**	04:19:26	04:51:02
jannis	8641	**00:01:41**	02:31:07	01:41:51
MiniBooNE	52147	**00:04:23**	03:59:56	02:11:01
KDDCup	49402	**00:05:06**	03:47:20	02:37:38
segment	2310	**00:00:25**	01:15:45	00:33:02
Higgs	110000	**00:06:16**	07:37:55	05:43:24
Credi-g	30000	**00:04:39**	02:03:34	05:33:03
shuttle	57999	**00:05:48**	05:15:45	04:26:03
APS Failure	60000	**00:05:39**	03:58:39	05:23:35
nomao	31772	**00:04:08**	03:01:15	02:49:36
CustSat	76020	**00:06:06**	05:07:03	04:09:36
kr-vs-kp	3196	**00:00:54**	01:17:19	00:22:44
car	1728	**00:00:38**	01:38:30	00:40:07
albert	43824	**00:06:27**	04:09:17	03:01:03
airlines	5473	**00:01:40**	02:18:27	00:57:52
Numerai28.6	6574	**00:03:22**	02:07:39	01:16:17

Table 3.10: The run-time (in HH:MM:SS format) of the AMLBID, Autosklearn and TPOT tools on the benchmark datasets.

with more than 4 millions of evaluated pipelines hence, it is capable to generate the recommendation by comparative search of meta-features with most similar existing datasets. Furthermore, with each iteration of fresh datasets, the meta-knowledge base of AMLBID is further enriched with evolutionary training. The confidentiality of the fresh dataset is respected by the fact that the knowledge base of AMLBID consists of the meta-features of the datasets and not the data.

We can evidently conclude, from any of these obtained results, that AMLBID is significantly more accurate than all the baseline AutoML frameworks.

We have used the k-nearest neighbor meta-model for the sake of comparison to the decision tree meta-model because the rankings using KNN classifier is among the most commonly used algorithms for obtaining top-k rankings in meta-learning [158]. After identifying the closest neighbors of the dataset, using the Euclidean Distance metric, the optimal pipeline configuration is forecast using a weighted average of each individual neighbor's ranking. The choice of KNN meta-model over the random forest can be justified by the results shown in Figure 3.4. It presents the performance of the RF and KNN meta-models on suggesting the optimal predictive pipeline configuration. The KNN based meta-model performance can be perceived as better than the random forest classifier based meta-learner according to the accuracy metric.

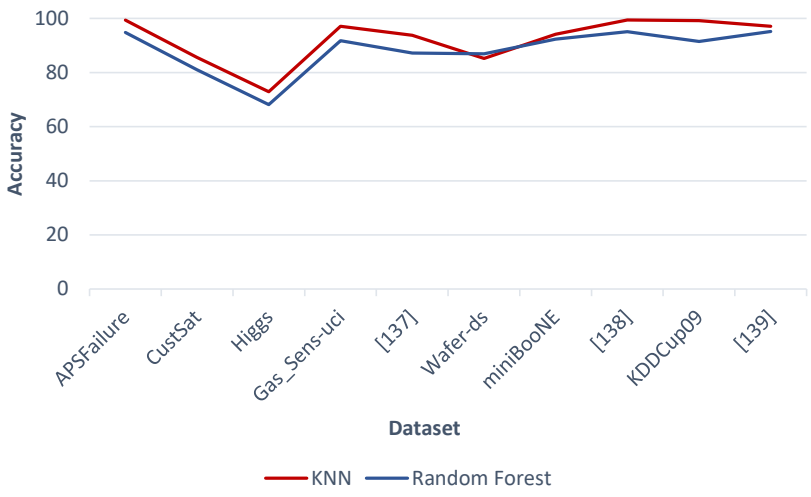

Figure 3.4: Predictive performance of the KNN and RF meta-models.

3.6 Conclusion

To enhance the production quality and improve the manufacturing industry, new techniques are being developed consistently. The machine learning techniques have been promising for the interests of the fourth industrial revolution. However, the customised knowledge required to use ML during the training process has been among the major obstacles to incorporate and advance ML models in manufacturing industrial domains. The dependence on manual crafting of ML models to produce desired performance makes it a difficult task despite the proven potential of ML models to improve the production. One approach that may help in reducing human's interventions in ML is automating the process of ML algorithms design and learning.

In this chapter, we addressed the problem of assisting non-expert users to perform algorithm selection and configuration with the goal of improving the final results of their classification tasks. To provide assistance, we presented the design, implementation, and evaluation of an automated ML system for manufacturing industry. AMLBID, in this context, is a novel meta-learning-based tool that address the problematic of automated selection and configuration of ML algorithms. The proposed system uses a recommendation engine that incorporates a meta-knowledge base maintained by the previous and ongoing recommendation results in manufacturing domain. The AMLBID explicitly train meta-models which are capable of identifying effective pipelines by exploring the interactions between datasets and pipelines topology without performing expensive computational analysis. In this regard, we presumably prevail the major limitations of AutoML-based systems which have been the computational complexity and excessive run-time performance losses.

Meta-learning systems use a set of data characteristics to represent and characterize data mining tasks, and search to identify the correlations between these attributes and the performance of learning algorithms. The proper identification of data properties is essential to map tasks to learning mechanisms. As a data-driven approach, the effectiveness of meta-learning is largely dependent on the description of tasks (i.e., meta-features). Meta-learning requires meta-features that represent the primary learning tasks or datasets to transfer knowledge across them. In the next chapter, we attempt a novel approach to extract intrisic data characteristics after reviewing the existing works in characterizing datasets, assess the currently available approaches and methods with respect to meta-features used as input to quantify the tasks similarity in the meta-learning process.

Chapter 4

Learning Abstract Tasks Representation

Outline of the current chapter

Meta-learning, or learning to learn, is an AutoML approach that uses prior learning experiences to expedite the learning process on unseen tasks. As a data-driven approach, appropriate data characterization is crucial for the meta-learning. A proper form of data characterization can guide the process of learning algorithms selection and configuration. The recent literature witness a variety of data characterization techniques including simple, statistical and information theory based measures. However, these estimated traditionally as

engineered dataset statistics that require expert domain knowledge tailored for every meta-task. Therefore, their quality still needs to be improved. This chapter presents new measures, based on an induced Autoencoder-kNN network architecture baptized as AeKNN, to characterize datasets for meta-learning in order to select appropriate learning algorithms. The main idea is to induce new intrinsic meta-features with lower dimensionality but more significant and meaningful features as latent characteristics of datasets from the traditional ones. Their effectiveness is illustrated through extensive experiments in an application on a large-scale hyperparameters optimization task for 400 real world datasets with varying schemas as a meta-learning task. We show that AeKNN offers considerable improvements of the classical kNN as well as traditional meta-models in terms of performance.

4.1 Introduction

Meta-learning refers to any learning approach that systematically makes use of prior learning experiences to accelerate training on unseen tasks [159]. One major goal is to build self-adaptive systems that adjust their learning mechanisms automatically with new tasks. Automatic adaptation can be described in a plethora of ways. It can be as simple as selecting an algorithm or a family of learning algorithms, tuning hyperparameters, or simply warm-starting a model [83]. Meta-learning relies on past experiences stored in the form of meta-knowledge. One type of meta-knowledge encompasses families of meta-features used as a form of data (or task) characterization. Meta-features capture various types of data properties such as number of numerical attributes, degree of class separation, Fisher's Linear Discriminant [160], or level of concept complexity [161].

As a data-driven approach, the effectiveness of meta-learning is largely dependent on the description of tasks (i.e., meta-features). In the current context, meta-learning requires meta-features that represent the primary learning tasks or datasets to transfer knowledge across them. We observe, in the available literature that several approaches in meta-learning use families of meta-features as input to quantify task similarity. It is common to compute tasks similarity as the Euclidean distance between two meta-features vectors. While these approaches have shown to be effective in simple scenarios, they exhibit clear limitations [162]. The foremost non-trivial task among the exhibited limitations is the identification and selection of relevant meta-features. Several research questions can emerge to better address these limitations such that *What criteria should we invoke to include or discard a family of meta-features?* For instance, sta-

tistical meta-features are not always intuitive and lack expressiveness. In [163], the authors have shown how different datasets may share identical statistical properties but noticeably they have different data distributions. Ultimately, the selection of meta-features is an *ad hoc* process based on domain knowledge. It is highly desirable to develop the more predictive meta-features and select the more informative ones in order to improve the effectiveness of meta-learning [84, 164, 165].

We believe that traditional meta-features are not always able to capture crucial characteristics of a given task, even though some of them are very task specific [83]. This can be attributed to the fact that they only model the general characteristics of the dataset (e.g. number of instances, classes imbalance, etc.). Learning relevant meta-features can be useful to better identify the hidden relationships across tasks, to necessarily build the accurate meta-models.

A different approach that has achieved popularity in recent years invokes Artificial Neural Networks (ANNs). The strength behind ANNs is their capacity to learn data characteristics from the diverse and large amount of data [166]. ANNs have had a strong impact in application areas such as image understanding and speech recognition [83, 167]. However, their use in meta-learning is still incipient and requires further investigations. The development of deep learning for features generation has been largely studied in the literature. It represents different datasets and tasks as embeddings generated by trained deep networks. In [168], the authors solve different automatic speech recognition tasks through a two-step learning process. In the first step, the algorithm performs a classification with ANNs, which is followed by the extraction of intrinsic features from the DNN output. In the second step, extracted features are used to improve model predictions.

Our hypothesis is that ANNs provide the means to extract intrinsic meta-features from data. In particular, Autoencoders are a type of artificial neural networks offering good results due to their architectures and operations [169–172]. In this chapter, we propose an instance-based algorithm, that learns latent meta-features from families of traditional ones. Its objective is to obtain meaningful and more informational meta-features. Specifically, the present chapter introduces AeKNN, a kNN-based algorithm with built-in latent features extraction strategy. AeKNN projects the training patterns into a lower-dimensional space, with the help of an Autoencoder (Ae). The goal is to produce new meta-features of higher quality from the initial data characteristics. In short, AeKNN combines two reference methods, k-Nearest Neighbors (KNN) and Autoencoder, in order to take advantages of Autoenconder in learning higher-level features. Thus, it supports kNN in performing pipelines recommendation in meta-learning paradigm.

The main contribution of this chapter is the design of a novel meta-model, called AeKNN, which combines an efficient latent features extraction mechanism (autoencoder) with a popular classification model (KNN). For the experimentation purposes, a collection of 400 real world problems and 8 ML algorithms have been used to assess the competitiveness of the proposed meta-model.

The rest of the chapter is organized as follows : In section 4.2, a brief review of the closely related works is introduced, including meta-learning for algorithm selection and data characterization techniques. In section 4.3, the proposed AeKNN meta-model is described. Finally, section 4.4 describes the experiments illustrating the effectiveness of the proposed approach.

4.2 Theoretical background and related works

Meta-learning involves two basic aspects : the characterization of the learning problems (datasets), and the identification of the correlation between the optimal learning algorithms and the problems characteristics. The first aspect relates to the techniques for characterizing datasets with meta-features, which constitutes the meta-data for meta-learning, whilst the second one is the learning stage at the meta-level, which develops meta-models for the selection of appropriate algorithms and related hyperparameters configuration in respect of previously unseen datasets.

4.2.1 The problem statement

Given a classification task on a dataset $\mathcal{D}$ with n instances, our goal is to compute a meta-feature $\mathcal{F}$ on $\mathcal{D}$. A meta-feature is usually a hand-crafted characterization function that captures a specific property of interest on a given task. Meta-features are regarded as a form of meta-knowledge collected over a distribution of tasks to learn how to learn. Not all meta-features are informative, and some of them are very task specific [83]. Learning relevant meta-features can prove useful in identifying hidden relationships across tasks, and is necessary to build accurate meta-models.

Knowledge extracted across tasks, a.k.a. meta-knowledge, is a key feature to the success of meta-learning by obviating learning from scratch on new tasks [83]. By exploiting meta-knowledge, the meta-model can effectively construct an optimal solution based on past experiences [162, 173] . For example, a meta-model can identify that a new task is similar to previous ones and warm-start a similar model with optimal hyperparameters. This avoids the –sometimes painstakingly– slow processes of error and trial in building a new

model. Meta-knowledge can be understood as meta-features, model hyperparameters, performance measures, etc. In our work, meta-knowledge consists of meta-features and performance measures gathered from previous tasks.

The process of meta-features extraction is formalized by [156] as a function $\mathcal{F} : \mathcal{D} \rightarrow \mathbb{R}^k$ that receives a dataset $\mathcal{D}$ as input, and returns a features vector of k values characterizing the dataset, and that are predictive of algorithms performance when applied to the dataset. Formally, it can be detailed as follows :

$$\mathcal{F}(\mathcal{D}) = \sigma(m(\mathcal{D})) \tag{4.1}$$

Where $\mathcal{D} = \{(x_i, y_i)|i \in \{1, \ldots, n\}\}$ is a dataset with n instances; x_i and y_i indicate the i-th training data and label respectively. The measure $m : \mathcal{D} \rightarrow \mathbb{R}^{k'}$ can extract more than one value from each data set, i.e., k' can vary according to $\mathcal{D}$, which can be mapped to a vector of fixed length k using a summarization function σ. In meta-learning, where a fixed cardinality is needed, the summarization functions can be, e.g., *mean*, *min*, *max*, *skewness* and *kurtosis*. Thus, a meta-feature can therefore be seen as a combination of a measure and a summary function [156].

4.2.2 Data characterization

The task of characterizing datasets for meta-learning is to capture the information about learning complexity on the given dataset and identify structural similarities and differences among datasets [84]. The most early attempts to characterize datasets in order to predict the performance of classification algorithms were made by Rendell *et al.* [174]. We observe in the literature that broadly two main strategies are proposed subject to characterize a dataset for suggesting which algorithm is more appropriate for a specific task or dataset. Among them are the methodologies using statistical measures and a set of simplified learners. The former attempt to describe the properties of datasets using statistical and informational measures. In the later, a dataset is characterized using the training performance (e.g. accuracy) of a set of simplified learners, which became later *Landmarking* [96].

The intuitive idea behind *Landmarking* is that the performance of classifiers is related to the intrinsic features of the problem. Thus, classifiers with similar accuracy may indicate problems with similar characteristics. Characterization with the use of *Landmarkers* is known as *indirect* characterization because it is not directly related to the attributes of the problem (cf. section 2.1.1 Chapter 2).

The characterization of datasets using statistical and informational measures properties appeared for the first time within the framework of the STATLOG project [175]. The authors used a set of 15 characteristics, spanning from simple ones, like the number of instances and the number of attributes, to more complex

ones, such as the canonical correlation between the attributes and the classes. This set of characteristics has been later applied in various studies for solving the algorithms selection problem [17, 145, 176]. This characterization approach has been later extended. It is currently known as *direct* data characterization [177] and consists of extracting simple, statistical, and information-theoretic task properties that can be straightforwardly extracted from datasets by capturing information concerning data dimensionality, distribution, and the amount of information present in the data.

Another characterization method is based on informations extracted by models built out on the problems [84]. For instance, from a decision tree model constructed over a dataset, it is possible to extract structural informations about the tree itself, such as the number of leaves, nodes, and the tree depth [178]. Similarly, in [17], the authors proposed AutoGRD, a meta-learning approach for algorithms recommendation through graphical dataset representation. First, they applied the Random Forest algorithm to create a hierarchical representation of the datasets where the vertices represent the dataset instances and the edges indicate the existence of a sufficiently high co-occurrence score among them. Then, the Grid-Cross Downsampling method [179] has been used to generate the embedding representation of the obtained graph that is fed to train an XGBoost meta-model to predict the ranking of algorithms based on their performances. However, this approach suffers from a computational complexity of $O(V4)$ where V is the number of vertices in the analyzed graph. It is further observed that this approach is not practical for large datasets of real world problems.

Meta-features or data characteristics can be transformed to summarize the data, e.g., by reducing data dimensionality. For instance, in [180], the authors performed Principal Component Analysis (PCA) [181] to select relevant components, subsequently, a filter is used to extract the discriminating features and eliminate the redundant ones.

A different approach that has achieved popularity in recent years in learning most relevant features from data involves deep autoencoder neural networks. We blieve that Autoencoders provide the means to extract intrinsic meta-features from traditional ones. In this process, traditional meta-features are used by the Ae to learn relevant features, then, the knowledge captured in the hidden layers of autoencoder is used to extract latent meta-features. Once identified and extracted, they can be used by any meta-learning algorithm. Meta-features extractors should therefore satisfy the following constraints in order to be useful :

1. **Schema Agnosticism & Expressivity.** The meta-features extractor should be able to extract meta-features for a population of meta-tasks with varying schemas and complexity, e.g., datasets containing different predictors and target variables, also having a different number of predictors and targets.

2. **Scalability.** The meta-features extractor should be able to extract meta-features fast, e.g., it should not require itself some sort of training on new meta-tasks.

3. **Correlation.** The meta-features extracted by the meta-feature extractor should correlate well with the meta-targets, i.e., improve the performance on meta-tasks such as hyperparameters optimization.

4.3 The AekNN data characterization approach

AekNN, the meta-model proposed in this chapter, is a kNN-based meta-model with bluit-in latent features extraction method designed to deal with the algorithms selection problem. This section outlines the essential concepts AEkNN is founded on, such as the k-nearest neighbors classifier and Ae networks.

4.3.1 AekNN foundations

The KNN algorithm

kNN is a non-parametric algorithm developed to deal with classification and regression tasks. In classification, kNN predicts the class for new instances using the information provided by the k nearest neighbors, so that the assigned class will be the most common among them. Fig. 4.1 shows an example on how kNN works with different k values. As can be seen, the prediction obtained with k = 3 would be B, with k = 7 would be A and with k = 15 would be B.

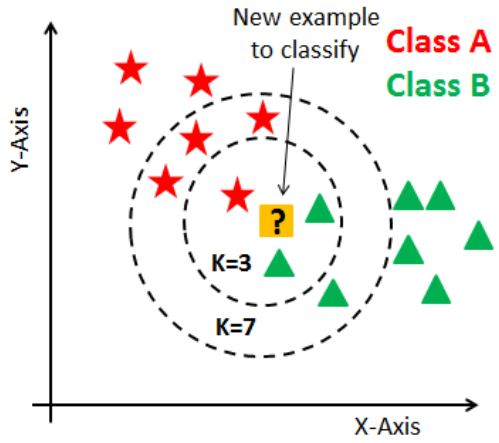

Figure 4.1: k-Nearest neighbors algorithm in a bi-dimensional space.

An important feature of this algorithm is that it does not build a model for accomplishing the prediction task. Usually, no work is done until an unlabeled data pattern arrives, thus the denomination of *lazy* approach [182]. Once the instance to be classified is given, the information provided by its k nearest neighbors is used as explained above.

One of kNN's main issues is its behavior with datasets having a high-dimensional input space, due to the loss of significance of traditional distances as the dimensionality of the data increases [183]. In such a high-dimensional space distances between individuals tend to be the same. As a consequence similarity / distance based algorithms, such as kNN, usually do not offer adequate results. The point that most of the state of the art studies, that use the KNN as a meta-model, overcome by reducing the number of meta-features. Thus, losing a set of important features that characterize the datasets.

In our work, kNN has been selected to perform meta-learning tasks. kNN is very popular since it has a good performance, uses few resources and it is relatively simple [182]. The objective of this proposal is to present a meta-model that combines the advantages of kNN to classify with Autoencoder networks to extract latent meta-features with lower dimensionality but more significant and meaningful characteristics.

One of the primary concerns in the selection of meta-model is the extensibility of the system, because a meta-learning system accumulate knowledge and evolves with experience as more meta-examples are added to the knowledge base. Hence, the addition of new meta-examples to the meta-knowledge base without the requirement of remapping the relationship of datasets and performance measures of candidate algorithms makes KNN a good choice for meta-model.

Autoencoders

Autoencoders are a type of artificial neural networks designed to learn efficient data representations (encoding) in an unsupervised manner [184]. An autoencoder is composed of *two* networks concatenated together : an **encoder** network and a **decoder** one. It has a similar structure to the feedforward neural network Multi-Layer Perceptron (MLP) [185]. However, the primary difference is that the number of neurons in the output layer is *equal* to the number of inputs, whereas the autoencoder tries to generate the inputs from the learnt representation (encoding) as close as possible to its original input. Consequently, in its simplest form, an autoencoder uses hidden layers to try to recreate the inputs. We can describe this algorithm in two parts :

1. an encoding function $Z = E(X)$ that encodes the high-dimensional input data $X = \{x_1, x_2, \ldots, x_n\}$ into a low-dimensional hidden representation $Z = \{z_1, z_2, \ldots, z_m\}$ by an activation function $f(x) = S_f(Wx + b)$, and

2. a decoding function $X' = D(Z)$ that produces a reconstruction of the inputs $X' = \{x'_1, x'_2, \ldots, x'_n\}$.

The goal is to create a reduced set of codings that adequately represents X by minimizing the reconstruction error $L(X, X')$, which measures the differences between the original input data X and the consequent reconstruction X'. There are two ways of formulating the reconstruction error: *square error* and *cross-entropy*. Their formulas are shown below:

Square error:

$$L(X, X') = \frac{1}{2} \sum_{i=1}^{n} \| x_i - x'_i \|_2^2 \, | \, i \in \{1, \ldots, n\} \tag{4.2}$$

Cross-entropy:

$$L(X, X') = -\sum_{i=1}^{n} (x_i \log x'_i) + (1 - x_i)(x_i \log (1 - x'_i)) \tag{4.3}$$

The general architecture of an autoencoder is described by the number of hidden layers l_i^n and by the number of neurons per layer, where i is the index for the hidden layer and n is the total number of neurons in that layer. Each layer contains a learnt latent representation of the input data. The encoded hidden layer in the middle of the autoencoder, often called the *bottleneck* layer, comprises the final learnt latent features, where each latent variable is a representation of the original input in an abstract space. The number of latent variables is user defined by controlling the number of neurons in that layer. By training an autoencoder on the traditional meta-features space, we can learn a new representation (latent meta-features). The resulting deep neural network serves as a features extractor where the learnt latent space Z is extracted from the middle hidden layer. This process is highlighted in Figure 4.2.

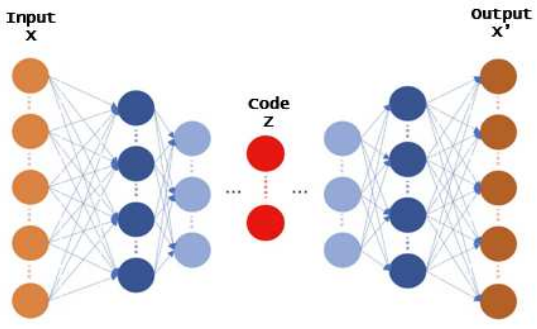

Figure 4.2: Schematic structure of an Autoencoder.

4.3.2 The AeKNN meta-model

AeKNN consists of two main phases : the *learning* phase and the *inferring* one. The former phase is carried out using the meta-dataset from the previous chapter to train the autoencoder model. It allows the extraction of latent meta-features of data. Later, the recommendation (inferring) phase is performed that principally uses the feed forward autoencoder model which has been generated in the learning phase to extract the latent meta-features of the test data and, later on, the recommendation and ranking of the optimal pipeline (s) are estimated based on nearest neighbors in the meta-knowledge base. Figure 4.3 elaborates this process, while the Algorithm 9 shows the pseudo-code of AeKNN that is thoroughly discussed in the following.

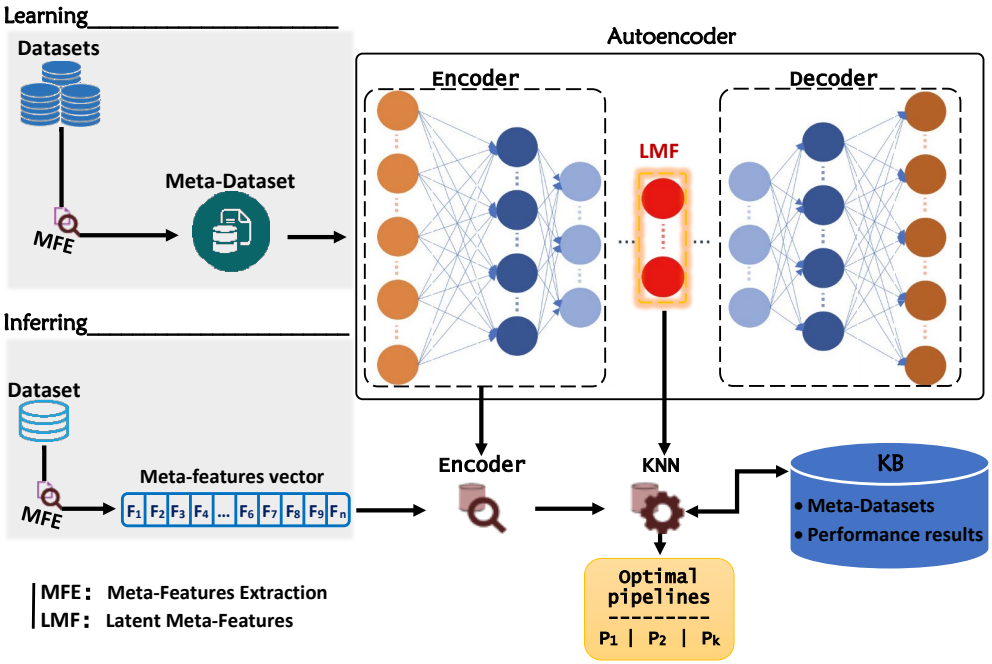

Figure 4.3: Overview of proposed AeKNN-based meta-model.

The proposed methodology constructs an autoencoder which can be used as a latent features extractor. After providing traditional meta-features as input, we train the autoencoder to learn a meaningful latent representation of the meta-dataset. Once the autoencoder is trained, the meta-dataset is forward propagated to extract the latent variables from the middle hidden layer to induce the AeKNN meta-model. The process consists of two phases. The first phase corresponds to the learning of AeKNN (lines 1-5) while the second phase (lines 6-8) refers to the inferring phase. During learning, AeKNN focuses on learning a new

Algorithm 9 AeKNN algorithm's pseudo-code.

Input : Train Data, Test Data, KB ▷ KB is the constructed knowledge base
Output : P< $P_1, P_2, P_3, \ldots, P_n$ > ▷ Suggested pipelines
Learning phase :
1: *MetaData* ← MetaFeaturesExtractor(*TrainData*)
2: *AE* ← Autoencoder(*MetaData*)
3: *EncoderModel* ← FeedForwardAEModel(*AE*)
4: *LatentMetaFeatures* ← EncoderModel(*TrainData*)
5: *AeKNN* ← KNN(*LatentMetaFeatures*, KB)
Inferring phase :
6: *MetaFeatures* ← MetaFeaturesExtractor(*TestData*)
7: *LatentMetaFeatures* ← EncoderModel(*MetaFeatures*)
8: *OptimalPiplines* ← AeKNN(*LatentMetaFeatures*, *KB*)

representation of the data to extract latent meta-features from traditional ones (cf. Section 3.4.2—Chapter 3). This is done through the feed forward autoencoder model, using the training meta-data to learn the weights linking the units of autoencoder. During the inferring phase, the optimal pipelines are generated. The process, performed internally in this phase, transforms the extracted meta-features using the autoencoder model which is generated in the training phase. It produces a new dataset characterization (latent meta-features), which is more compact representative (line 7) of data. In fact, this new set of features is used by the AeKNN meta-model to recommend the optimal pipeline (s) for the given problem (line 8). An illustrative example of this process is shown in Figure 4.4.

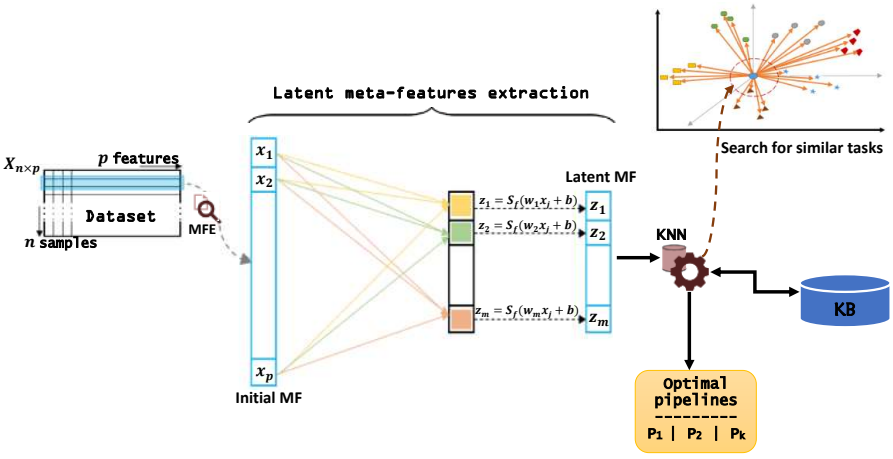

Figure 4.4: An illustrative example of the AeKNN inferring process.

4.4 Experimental study

This section describes the experimental design to induce latent meta-features and the evaluation of the proposed approach. In this respect we made use of the constructed meta-knowledge base from the last chapter. Subsequently, the experimental results are presented and discussed in substantial detail.

4.4.1 AeKNN architectures analysis

AeKNN is characterized by the aforementioned $l_i{}^n$ parameter that establishes the architecture of the network. This parameter allows the selection of different architectures in terms of depth (number of layers) and number of neurons per layer. Table 4.1 shows the considered architectures. For each model the number of hidden layers, as well as the number of neurons in each layer are shown.

Model	Number of hidden layers	Number of neurons per layer					Architecture $l_i{}^n$
		L 1	L 2	Latent layer	L 4	L 5	
AeKNN 1	1	-	-	**32**	-	-	(**32**)
AeKNN 2	1	-	-	**16**	-	-	(**16**)
AeKNN 3	1	-	-	**8**	-	-	(**8**)
AeKNN 4	3	32	-	**16**	-	32	(32,**16**,32)
AeKNN 5	5	32	16	**8**	16	32	(32,16,**8**,16,32)

Table 4.1: Experimental configurations of AeKNN.

The results produced by the considered architectures using a benchmark of 20 real world datasets with different characteristics (as shown on the table D.1 in the Appendix D) are presented as grouped by datasets. Table 4.2 summarizes the evaluation results of each recommended pipeline for each configuration.

The results presented in table 4.2 shows the evaluation results of the recommended pipelines by AeKNN with different architectures. The obtained results indicate that the architectures with single hidden layer get better results in 17 out of 20 datasets, whereas the architectures with three hidden layers gets 2 out of 20 datasets and the five hidden layers architecture obtained best results in 1 out of 20 datasets. In the rankings obtained, it can be seen that single hidden layer architectures are the ones winning more times (17). The $l_i{}^n = (32)$ works best for most cases (14 win), while $l_i{}^n = (16)$ shows disparate results, the best values for some cases and bad results for other cases.

Dataset	AeKNN				
	(32)	(**16**)	(**8**)	(32,**16**,32)	(32,16,**8**,16,32)
APSFailure	**0.9921**	0.9734	0.86475	0.9033	0.8325
Higgs	**0.7283**	0.6911	0.4872	0.6398	0.5316
CustSat	0.8155	0.7826	0.5318	**0.8559**	0.6943
car	**0.9999**	0.9808	0.7049	0.9203	0.8277
kr-vs-kp	**0.9985**	0.8130	0.6532	0.7330	0.7291
airlines	0.7021	0.6833	0.5627	**0.7167**	0.4334
vehicle	0.8952	**0.8964**	0.3591	0.8004	0.4098
MiniBooNE	**0.9730**	0.9217	0.8143	0.85	0.7436
jannis	**0.7229**	0.6843	0.6371	0.6911	0.6608
nomao	0.9884	**0.9919**	0.5395	0.6994	0.4659
Credi-g	**0.8037**	0.6502	0.5121	0.3871	0.4768
Kc1	**0.8905**	0.8754	0.3597	0.7488	0.5691
Cnae-9	**0.9800**	0.8923	0.5622	0.5208	0.6049
albert	0.8790	0.8131	0.6981	0.8439	**0.9053**
Numerai28.6	**0.5591**	0.4530	0.3029	0.4760	0.2810
segment	**0.9867**	0.9622	0.8837	0.9508	0.5791
Covertype	**0.8637**	0.7189	0.6521	0.6305	0.4620
KDDCup	**0.9781**	0.8514	0.8034	0.8821	0.8572
shuttle	0.9362	**0.9997**	0.6429	0.8576	0.6744
Gas_Sens-uci	**0.9843**	0.9755	0.7256	0.9667	0.7032

Table 4.2: Performances of considered AeKNN architectures on the test datasets. The best performances among all architectures are highlighted in bold.

Therefore, it is considered that ${l_i}^n = (32)$ is the best among them with a reconstruction error standard deviation of 0.020025 (Figure 4.5). Thus, in the following, the results of AeKNN using the presented architecture are compared against the classical kNN as well as other state-of-the-art meta-models.

4.4.2 Results of latent meta-features extraction

We extracted latent features of datasets form traditional ones by using the ${l_i}^n = (32)$ Autoencoder architecture. To comprehend the features extracted by AeKNN more intuitively, the obtained 32-dimensional meta-features were reduced to a 3-D space $\{X, Y, Z\}$. This was accomplished by the t-Distributed Stochastic Neighbor Embedding (tSNE) [186] which is used in the conversion of

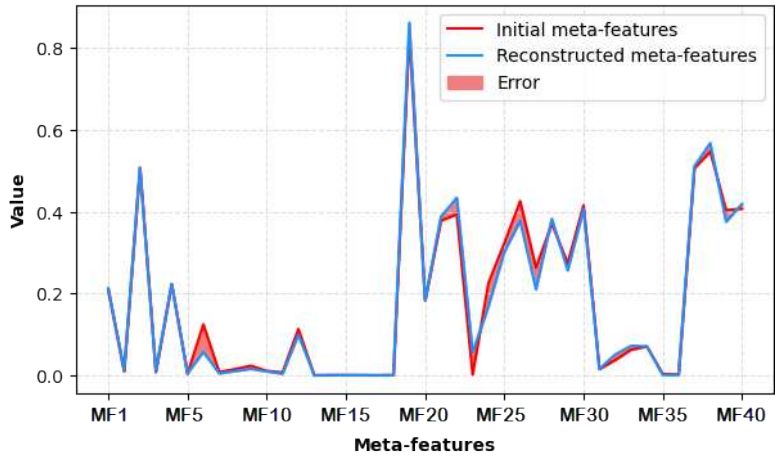

Figure 4.5: The reconstruction error of an instance from the meta-features set after it's encoded and decoded by the ${l_i}^n = (32)$ architecture.

high-dimensional data visualization into low dimensional embeddings. It converts high-dimensional Euclidean distance between data points into conditional probabilities for mapping low-dimension space and adopts kullback-leibler divergence [187] to minimize mismatch on the low-dimensional data representation. Using this technique, we can acquire more interpretable data visualization on high dimensional representations. In this experiment, we used the tSNE technique for mapping data into a three-dimensional plane for the Autoencoder based latent features selection and extraction. Figure 4.6 shows that the AeKNN features could separate different datasets reasonably well, as the different datasets are located in a sequence of regions. The same data are portrayed in Figure 4.7 in the space of different views 3-D scatter plots of the learned latent features. Marker shapes are the superposed datasets (each circle represents a dataset).

Among the visualizations of the extracted latent meta-features over the traditional ones, it is obvious that the datasets in the AeKNN features space are more sharply delineated, and the features associated with each class are mostly separated from those of others, although a small number of overlaps can be seen.

The results of latent features visualization show that our method can effectively extract meaningful characteristics (meta-features) of datasets for the meta-learing process. This method avoids the overlaps of separation and combination caused by the complex background of tasks (datasets) when using the traditional meta-features measures.

To validate and assess the competitiveness provided by the deep autoencoder-KNN based meta-model, we perform a comparative study to other state-of-

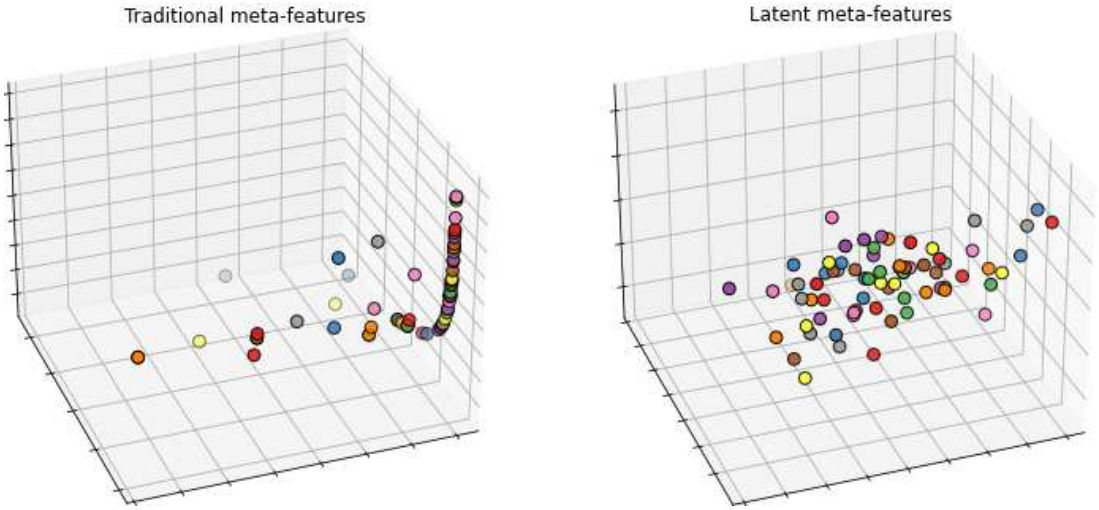

Figure 4.6: 3-D scatter plot of traditional meta-features of the datasets and the latent ones extracted by our model.

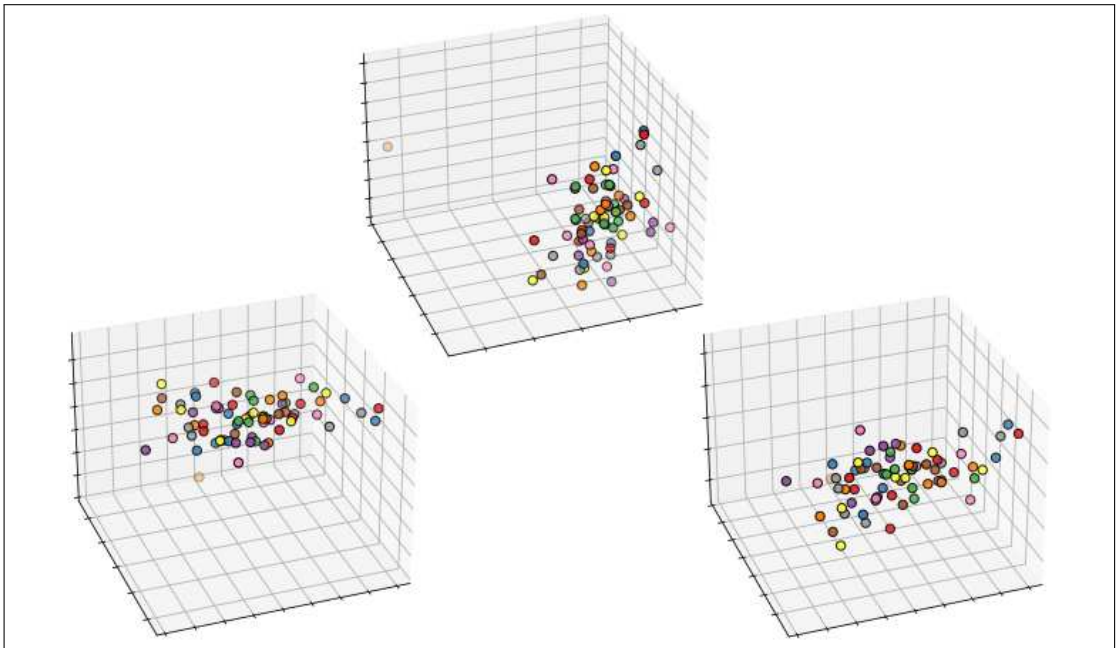

Figure 4.7: Three different views of the same extracted latent meta-features. The plots have been rotated to illustrative angles.

the-art meta-models with an oversampling approach using the 20-benchmark datasets. We compared AeKNN to three widely used meta-models including random forest, k-nearest neighbor, and XGBoost [17, 145, 176, 188].

4.4.3 Results of the algorithms selection process

This section is intended to assess the competitiveness of the proposed meta-model. In this regard, a comparison has been made between the results obtained by AeKNN, using the ${l_i}^n = 32$ architecture as selected in the previous section, and the results obtained with the baseline meta-models on the same datasets. Table 4.3 provides a pairwise one-to-one comparison of the baseline meta-models against AeKNN for recommending optimal pipelines for the test data. The results of all meta-models are presented jointly in Table 4.4, and the best ones are highlighted in bold. More peculiar and detailed results are presented in the Appendix D.

Meta-model	Wins	Losses	Champion
AeKNN	-	-	16
KNN	1	19	1
RF	0	20	0
XGB	3	17	3

Table 4.3: Comparing each baseline meta-model against AeKNN on the 20-benchmark datasets. Listed are the number of datasets where each meta-model produced better predictions than AeKNN *(Wins)*, worse predictions *(Losses)*, or more accurate predictions than all of the other 3 meta-models *(Champion)*.

The results shown in Table 4.4 indicate that AeKNN performs better than most of the traditional meta-models especially the classical kNN and RF for most datasets. The AeKNN improves kNN in 18 out of 20 cases, obtaining the best overall results in 16 of them and obtains better results than the Random Forest meta-model in all cases. In the presented ranking, the results of AekNN with the ${l_i}^n = 32$ architecture appear first, clearly highlighted with respect to the other obtained values. Therefore, it is considered that AekNN obtains better predictive performance, since the reduction of dimensionality generates more significant features. Table 4.3 confirms this trend.

Summarizing, it can be observed that the results obtained through AeKNN improve those obtained with the RF, XGB and KNN algorithms for most of the datasets. The quality of the results with AeKNN in terms of algorithms selection performance are better than those of baseline meta-models in most cases. This means that the high-level meta-features obtained by the AeKNN algorithm provide more relevant information than those obtained by the state-of-the-art algorithms.

Dataset	Accuracy			
	AeKNN	KNN	XGB	RF
APSFailure	**0.9921** (0.11) ▲	0.9910	0.9673	0.8950
Higgs	**0.7283** (1.53) ▲	0.7130	0.6801	0.6072
CustSat	0.8155 (4.04)▼	0.8559	**0.8715**	0.7382
car	**0.9999** (2.45) ▲	0.9754	0.9462	0.8549
kr-vs-kp	**0.9985** (0.09) ▲	0.9976	0.7593	0.6532
airlines	0.7021 (0.39)▲	0.6982	**0.7094**	0.5927
vehicle	0.8952 (0.72)▲	0.8880	**0.9027**	0.6591
MiniBooNE	**0.9730** (0.85) ▲	0.9645	0.8903	0.8343
jannis	**0.7229** (5.10) ▲	0.6719	0.6845	0.6171
nomao	**0.9884** (1.76) ▲	0.9708	0.7987	0.6995
Credi-g	**0.8037** (1.16) ▲	0.7921	0.5739	0.6121
Kc1	**0.8905** (1.12) ▲	0.8793	0.7697	0.7097
Cnae-9	**0.9800** (1.29) ▲	0.9671	0.8365	0.7922
albert	**0.8790** (0.31) ▲	0.8759	0.8288	0.7981
Numerai28.6	**0.5591** (3.84) ▲	0.5207	0.4836	0.4229
segment	**0.9867** (1.32) ▲	0.9735	0.9542	0.9337
Covertype	**0.8637** (2.93) ▲	0.8344	0.7890	0.6521
KDDCup	**0.9781** (0.41) ▲	0.9740	0.9331	0.8934
shuttle	0.9362 (2.87) ▼	**0.9649**	0.9649	0.8429
Gas_Sens-uci	**0.9843** (1.04) ▲	0.9739	0.9468	0.9256

Table 4.4: Results of RF, XGB, KNN, and AeKNN meta-models for recommending optimal pipelines for test data. The triangles (▲,▼) denote the gain/ loss obtained with AeKNN compared to the traditional KNN meta-model.

4.5 Conclusion

In this chapter, a novel meta-model based latent features extraction method, namely AeKNN, is proposed. This model is based on kNN to recommend the optimal pipelines while its major objective is to mitigate the high-dimensional data characterization limitations. In this regard, AeKNN internally incorporates a model-building phase which is aimed at an extraction of latent meta-features, using a feed forward autoencoder. The main reasons that has led to the design of AeKNN are the good results that have been obtained by the Autoencoders networks when they are used to generate higher-level features and those of KNN

for performing pipelines recommendation in meta-learning systems. AeKNN relies on a feed forward Autoencoder to extract latent representations of a higher level that replaces the original meta-features.

In order to assess the competitiveness of the proposed approach, a series of experiments are carried out. Initially, the analysis of the results have allowed to determine the architecture of AEKNN that works better. Furthermore, in the later parts of the conducted experiments, the results of the adopted architecture have been compared with the results produced by the state-of-the-art meta-models. It is observed that AeKNN offers a considerable improvement of the results obtained by all baseline meta-models. These results show that the use of autoencoders can be helpful to extract relevant meta-features which are more significant and informative. It thus improve the effectiveness of meta-learning, and broadens the directions of future works. They can be applied to support the solution of similar problems in a better manner than the traditional meta-models.

Chapter 5

Towards Interactive Explainable Automated machine learning

Outline of the current chapter

To relieve the pain of manually selecting machine learning algorithms and tuning related hyperparameters, automated machine learning methods have been remarkably successful for a wide range of application areas to automatically search for the best models. However, such a highly positive impact of theses powerfull black-boxes solutions is coupled with a significant challenge : *how do we understand the decisions suggested by these systems in order that we can trust them?*. Users tend to distrust automatic results and increase the search budget as much as they can, thereby undermining the efficiency of AutoML.

In an effort to identify open challenges and address these issues, we design and implement, a framework for interactive and explainable AutoML that enables users to (1) *understand* the reasoning behind a recommendation; (2) understand the provided results and *diagnose* model limitations using different explainable AI methods; as well as (3) explore the possibilities of performance *refinement*. To operationalize the framework, we present AMLExplainer, a visual analytics system for interactive and explainable AutoML that instantiates all phases of the suggested pipeline(s) within the commonly used Bootstrap Dash environment.

5.1 Introduction

The Rapid evolution of artificial intelligence technologies in the last decade has brought us many novel use cases and futuristic applications never seen before. The performance of ML techniques is becoming more than satisfiable, due to the large amount of available data. To ease the difficulty of developing ML models, automated machine learning methods have been proposed. Instead of searching algorithms and tuning hyperparameters manually, AutoML automatically iterates through various machine learning algorithms and optimizes hyperparameters in a predefined search space (i.e., a set of feasible ML pipelines).

The interest of building complex AI models that are able to achieve unprecedented performance levels has been gradually replaced by a growing concern for alternative design factors leading to an improved usability of the resulting tools. Indeed, in a manifold of application areas, complex AI models become of limited practical utility [189]. The major reason lies on the fact that AI models are often designed to focus the performance factors, thus leaving aside other important and even sometimes crucial aspects such as confidence, transparency, fairness or accountability. The absence of explanation for predicted performing factors makes the AI models usually black boxes, which only allows the prominent exhibition of input and output parameters but conceals the visibility of inherent associations among them. It is more preferably desired to avoid such a lack of transparency in real-life applications such that in industrial manufacturing processes. Since, these applications may imply critical decision choices, it is favorable to have some justifications of individual predictions which are perceived trough an AI algorithm, more particularly, in an automated environment. Therefore, the acceptance of, and the trust in, an AutoML system is highly dependent on the transparency of the recommendations.

Because of the lack of transparency in AutoML systems as Decision Support Systems (DSS), users tend to question the validity of automatic results, such

that : *did the AutoML run long enough? Did the AutoML miss some suitable models? Did the AutoML sufficiently explore the search space? Did the recommended configuration over or under fit? Etc.* Such queries may cause reluctance for users to apply the results of AutoML in more critical situations [190]. Meanwhile, when AutoML provides unsatisfactory results, users are unable to reason and thus cannot improve the obtained results. They may only increase the computational budget (e.g., the run-time) as much as possible, which can result as barriers of the AutoML effectiveness.

It is therefore a preliminary objective of this chapter to make the outcome from such well-performing AutoML systems transparent, interpretable and self-explainable. This shall make AutoML support systems more reliable and operational through a set of different visual summary levels of the provided models and configurations. It may render the AutoML systems more transparent and controllable, hence increasing their acceptance.

5.2 The need for transparency to trust in AI and in AutoML

Black-box AI systems have been used in various areas. Their implication in critical domains, like in power consumption forecasting or supply chain management usually have less focus to consider the quality features such as transparency and explainability rather considering more importantly the system's overall performance. However, even if these systems fail, e.g., the Quality Control System is mostly not able to detect the failure, the Equipment Failure Prevention system are less expected to identify the exact cause of failure and generally produces false or inaccurate predictions. The consequences are rather underwhelming. In industrial critical applications, the situations are different where the lack of transparency of ML techniques can be a disqualifying factor, if not limited. Specifically, a single wrong decision can be highly risked to put in danger the entire production line (e.g., failure of a critical unit) and can cause significant financial deprivations (e.g., product conformity). It is therefore, relying on an incomprehensible black-box data-driven system would not be the best option. The lack of transparency is among the most relevant reasons to question the adoption of AI models in manufacturing industry. The stakeholders are more cautious than doing so in the consumer entertainment, or e-commerce industries.

Predictive accuracy metrics, e.g. precision and recall, may not be reliable enough to assess the usefulness of a ML model [191, 192]. For many tasks, in order to trust a ML model and use it for making real-world decisions, it is

needed to understand what relationships the model has learned, how the model produces its outcomes, how the model's decision logic differs in different parts of the features space, possible biases in the data and model, and the collective influence of features on the model output (Figure 5.1).

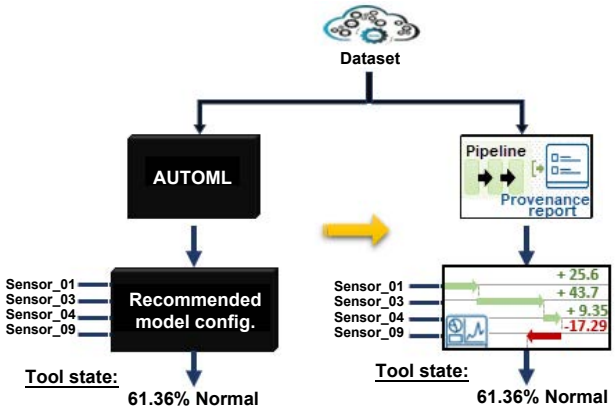

Figure 5.1: From "Black-box" model recommendation and prediction to "White box" model with explanations.

Explaining the reasoning behind one's decisions or actions is an important part of human interactions in the social dimension [193]. As the explanations help to build trust in human-to-human relationships, similarly, these should also be part of human-to-machine interactions [146]. In this chapter, we investigate the contributions and feasibility of a process designed to make such powerful DSS transparent, interpretable and self-explainable to foster trust, both in situations where the AI system has a supportive role (e.g., production planning) and in those where it provides directions and decision-making (e.g., Quality Control, predictive maintenance or autonomous driving). In the former cases, explanations provide extra information, which help the human in the loop to gain an overall view of the situation or the problem at hand in order to take decisions. It is similar to an expert who has to provide a detailed report explaining his/her findings, a supportive AI system should explain the decisions in detail instead of providing only a prediction or a decision.

5.3 Explainable Artificial Intelligence

Explainable AI (XAI) [189] refers to artificial intelligence technologies that can provide human-understandable explanations for their outputs or actions [194].

End users, by nature, may wonder about the reasoning behind how and why algorithms make or arrive to decisions [195]. As the complexity of the AI algorithms and systems grows, they are viewed as "black-boxes" [196, 197]. Increasing complexity can result in the lack of transparency that hampers understanding the reasoning of these systems, which negatively affects the users trustiness.

Model explainability can be divided into two categories : *global* explainability and *local* explainability. Global explainability means the users can understand the model directly from its overall structure. Local explainability just consider a specific input and tries to find out why the model makes a certain decision.

The development of methods for explaining, visualizing and interpreting machine learning models has recently gained increasing attention under the XAI area [189, 190, 194–196]. In the recent years, the advancements in XAI are grown rapidly but there are still broader gaps to generalize XAI approaches. The current major XAI methodologies are only applicable to specific type of data and models. Such specificities mostly require the pre-configuration of input parameters that are not easily coded by non-experts [190]. A variety of XAI methods characteristics in terms of data explanations level, data and model dependency, and pre-configuration requirements are highlighted in Table 5.1.

XAI method	Level		Dependency		Require pre-configuration
	Local	Global	Data	Model	
LIME[190]	●	○	●	○	●
ANCHORS[198]	●	○	●	○	●
Node-Link Vis[199]	●	●	○	●	●
SHAP [200]	●	●	○	○	●
DeepTaylor[201]	●	○	●	●	●
Occlusion [202]	●	○	●	●	●
Saliency [203]	●	○	●	●	●

Table 5.1: Properties of XAI state of the art tools. *Level* is the interpretability coverage: local or global. *Dependency* specifies necessary inputs.

The Local Interpretable Model-Agnostic Explanations (LIME) [190] is one such methods. It uses the models output on a data sample to generate a linear surrogate model that explains the features importance. A similar technique, ANCHORS [198], additionally focuses the most influential input areas, so-called anchors, to formalize decision rules. Both methods do not consider the underlying model (model-agnostic) but use the sample inputs and outputs of the model (data-dependent) to explain a (local-level) decision boundary generated by the model.

A different type of XAI methods is represented by Saliency [200]. They built a visual representation for features importance by highlighting aspects in each sample as a mask of how the model perceives its input data. In contrast to LIME and Anchors, they are only used for artificial neural networks (models-pecific).

Other XAI methods only allow for a low-abstraction, such as visualizing convolutional filters [204], or showing the dataflow through the computational graph [205]. These methods are especially useful for model developers, who want to improve their models using a low abstraction XAI method as a quality metric.

While all the existing methods are highly-specialized to their use-cases and cover the respective insights and application constraints. They provide only some XAI methods without an interactive machine learning (IML) workflow. Therefore, they cannot be used as XAI components in an AutoML DSS. An ideal system for explaining ML models needs to provide a collection of different XAI levels and should be flexible enough to adapt to the AutoML output (model and data agnostic).

5.4 Visual Analytics for AutoML

Recent design recommendations put more focus on the importance of intuitive interfaces, along with a clean and concise presentation, among the explanation facilities, and easy user interactions [206]. Visual Analytics (VA) can be applied to the IML workflow to boost the model development and deployment processes through tailored visual interfaces, and tightly integrates the user to promote further sensemaking during the data analysis workflow [207].

During our review of existing IML/VA systems, we identified *three* stages of explanations according to how they can cover the provenance tracking and reporting of the AutoML. Moreover, we show why a general XAI system, comprising all stages and tasks, is needed to address the variance in interpreting black-boxes AutoML solutions.

The **understanding** stage can be interpreted in different ways depending on the target user group (see Figure 5.2). For a model novice, an interactive VA system can be used as an "educational" tool to explain ML concepts. For instance, Harley [204] visualizes changes of an image along with the affected layers of an ANN. Smilkov *et al.* [208] also provide an interactive, visual representation of an ANN. Further work offers various ways to explore the graphical representation of DNNs. From these examples, we derive the need for an interactive exploration during the understanding phase. In contrast to the educational goals of model novices, model users and developers need to understand the

model inner-workings. Rauber *et al.* [209] focus on this aspect by visualizing the ANN training, as well as, both, neuron-neuron and neuron-data relationships. Bilal *et al.* [210] visualize the hierarchical abstraction of CNNs, highlighting the importance of multiple abstraction layers. Based on the lessons learned from these works, we conclude that there is a need for providing tailored AutoML explanations on different model abstractions levels to understand the algorithms recommendation and diagnosis process.

Visual Analytics systems can address this gap by focussing on model **diagnosis** in an IML workflow to enable the detection of problems on different abstraction levels. Some systems support a model diagnosis by focusing on features importance [200, 211], the reaction of the model to real or adversarial input examples [212, 213]. Others focus on specific elements, such as the neuron activation [214], hidden states of a cell [215] or action patterns of reinforcement learning algorithms to allow model-specific diagnosis. While all these approaches allow for an integrated diagnosis, they fall short of addressing the identified issues in a subsequent refinement step [216].

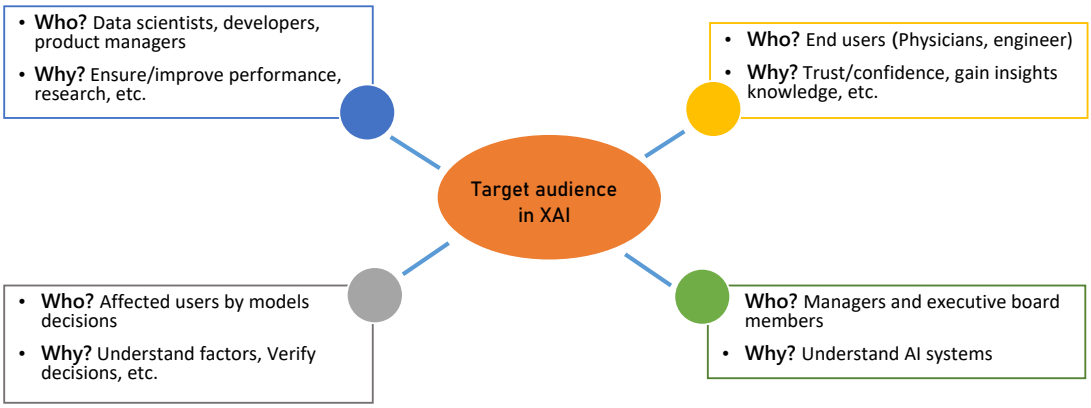

Figure 5.2: Diagram showing the different purposes of explainability in ML models sought by different audience profiles. Image partly inspired by the one presented in [217], used with permission from IBM.

An IML Expert system can go beyond the understandig and diagnosis phases, and target the **refinement** of ML models. We have identified works that are designed to diagnose and refine single ML models, e.g., [218, 219]. Others target multi-model visual comparison for refinement [220, 221]. In addition to this distinction, various interactive refinement approaches are used in iterative cycles on medical images [222]. Such examples highlight the need for interactive and iterative refinement cycles in our self-explainable AutoML vision. Further, the ability to assist on the selection, configuration and refinement of adequat ML

models is essential for assessing the quality of different models and selecting the most suitable for a given context.

In our system implementation, we attempt a transparent and interactive XAutoML pipeline that can cover different pathways through all of the addressed stages and tasks. Given a dataset, the system automatically recommends the most adequate ML configurations and explains the rationale traceability behind its recommendation. It is intended to support the analysis and inspection of all machine learning classification models without any data type or model dependency. The goal is manifold : (1) facilitate the models working and performance inspection through linked visual summaries and textual information, (2) provide a visual summary of all evidence items and their relevance for the computation result, and (3) present a guided investigation of the reasoning behind the recommendation generation and for performances refinement possibilities. In the proposed approach, the end users can explore the AutoML process at different levels, such as described in the following :

- The AutoML-oriented level (i.e. exploring the AutoML process from recommendation to refinement).
- The Data-oriented level (i.e. exploring data properties through different visualization levels).
- The Model-oriented level (i.e. exploring the models provided by the AutoML system (e.g. model performance, what-if-analysis, decision path, etc.)).

5.5 The Conceptual framework

Given a predictive modeling problem for an industrial application, it is often difficult to build an accurate machine learning based predictive model that is easy to develop and to be interpreted by non-ML experts. The key idea for our transparent and explainable automated machine learning vision is to separate recommendations from explanations by using two modules simultaneously. The first module is used to recommend the most adequate ML configuration for a problem at hand and aims to maximize the requested predictive performance metrics (e.g. Accuracy, Precision, Recall). The second module is used for providing the rationale behind the recommended configuration as well as automatically explaining the inner workings of the model.

The following section describes the design and implementation choices of the proposed tool that is intended to provide a complete, transparent and self-explainable AutoML system. As it is shown by the Figure 5.3, for the recommender module (AMLBID), when a new dataset is presented, AutoML is

performed, and a list of candidate pipelines is provided based on the task at hand (cf. Chapter 3). The dataset characteristics, AutoML output and candidate pipelines list are supplied to the explanatory module to generate an interactive dash to help the end-user *understand* the provided results, *diagnose* the performance of the generated pipelines and explore the possibilities of performance *refinements*.

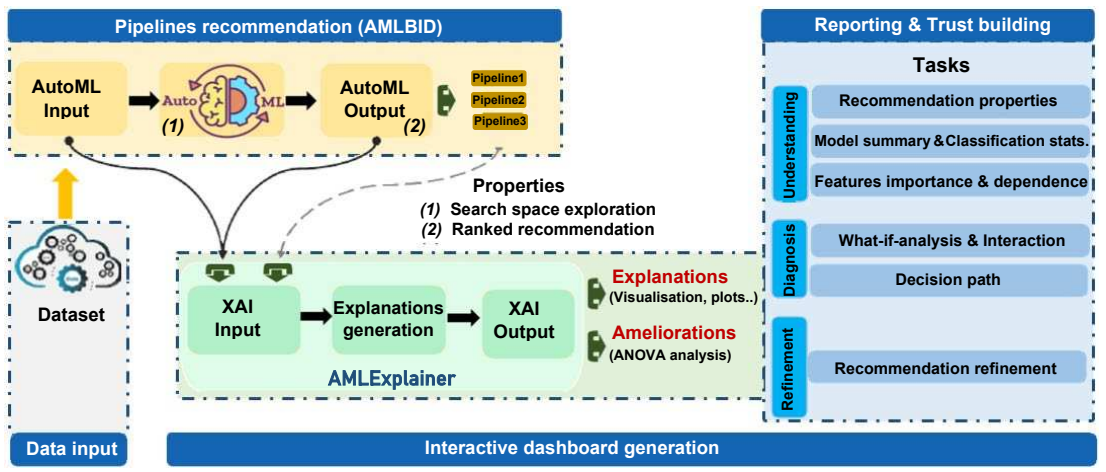

Figure 5.3: The global architecture of the proposed white-box AutoML (AMLBID and AMLExplainer).

AMLExplainer is implemented as a client-server tool integrated with the recommender module. The server coordinates as an AutoML support system. As the client, the visual interface provides graphical interactions with AutoML results and maps the summary data for visualization through a set of different visual summary levels of the recommended models. AMLExplainer users are allowed to explore the models provided by the AutoML process at four main levels of detail (i.e. AutoML Overview, Recommendation-level View, What-if analysis-level View, and Refinement-level View). Meanwhile, AMLExplainer provide end users with a guidance, when AutoML returns unsatisfying results, to improve the predictive performances. Thence increases the transparency, controllability, and the acceptance of AutoML.
The workflow of the proposed auto-explanatory AutoML system consists of two main components :

- The AutoML component, which shows the high-level of the AutoML process from recommendations to refinements.
- The XAutoML component, that allows users to inspect the recommended model's inner working and decision's generation process.

5.5.1 The AutoML Overview

The AutoML overview level (Figure 5.4) summarizes high-level information of the AutoML process. Users will be able to compare and choose between the top K recommended configurations. They can focus their analysis on the top model configuration on the next level view, which highlights the corresponding algorithm in the detail views.

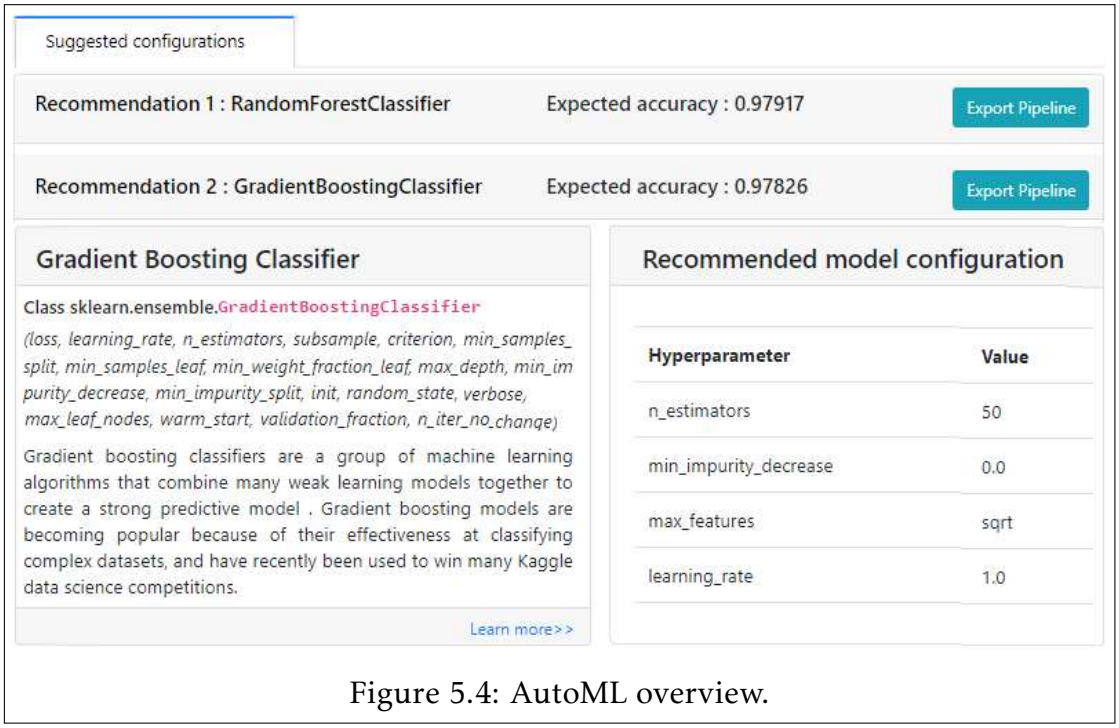

Figure 5.4: AutoML overview.

5.5.2 The recommendation-level View

The recommendation-level view enables users to inspect recommendations with respect to performance distribution. As shown in (Figure 5.5), a detailed explanation about the top performed recommendation is generated through multiple granularity levels of abstraction, such as statistics about the configuration performances (Figure 5.5(A)), Tree based explanation of the conducted predictions (Figure 5.5(B)), the importance of features and the contribution of features to the individual predictions (with the help of SHAP tool that finds the shapely values of a contribution to the predictions) (Figure 5.6),

By providing intelligible explanations about the process and reasoning behind an individual prediction, as illustrated in the Figure 5.5, it is clear that

the decision-maker whether a manufacturing engineer or a machine learning practitioner is much better positioned to make decisions since (s)he usually have prior knowledge about the data and the application domain, which can use to trust in and accept or reject a prediction if the reasoning behind it is well explained.

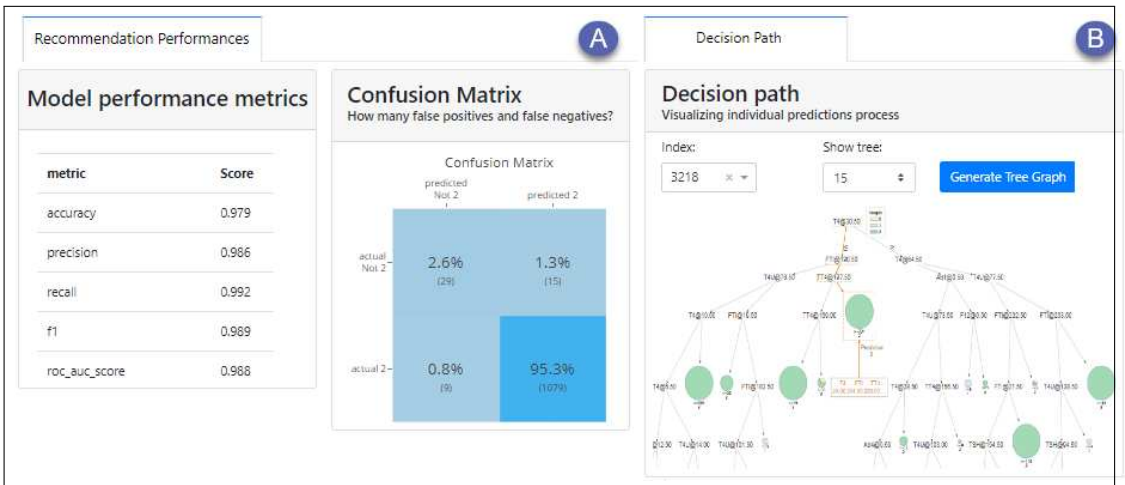

Figure 5.5: Recommendation-level view.

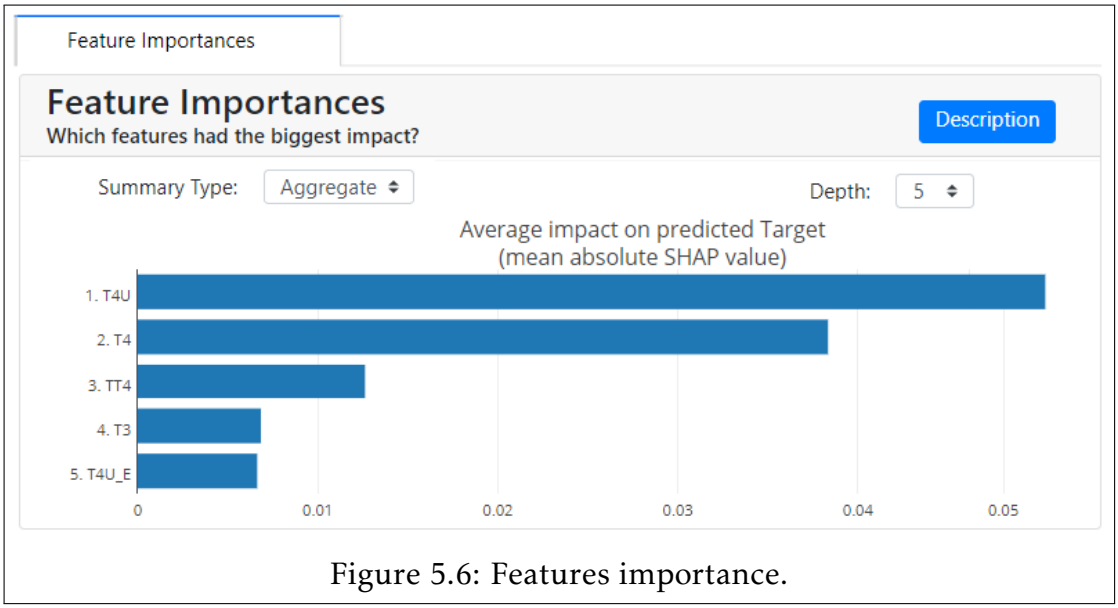

Figure 5.6: Features importance.

5.5.3 The What-if analysis-level View

The What-if analysis-level view : (Figure 5.7) is designed to investigate the machine learning models. It enables understanding models by enabling end users to investigate attribution values for individual input features in relation to model predictions. Explaining the inner working of the model helps to gain an understanding of what the model does and does not do. This is important so that they can gain an intuition for when the model is likely missing information and may have to be overruled. Therefore explore scenarios, test, and evaluate / validate business assumptions, and gain intuition for modification.

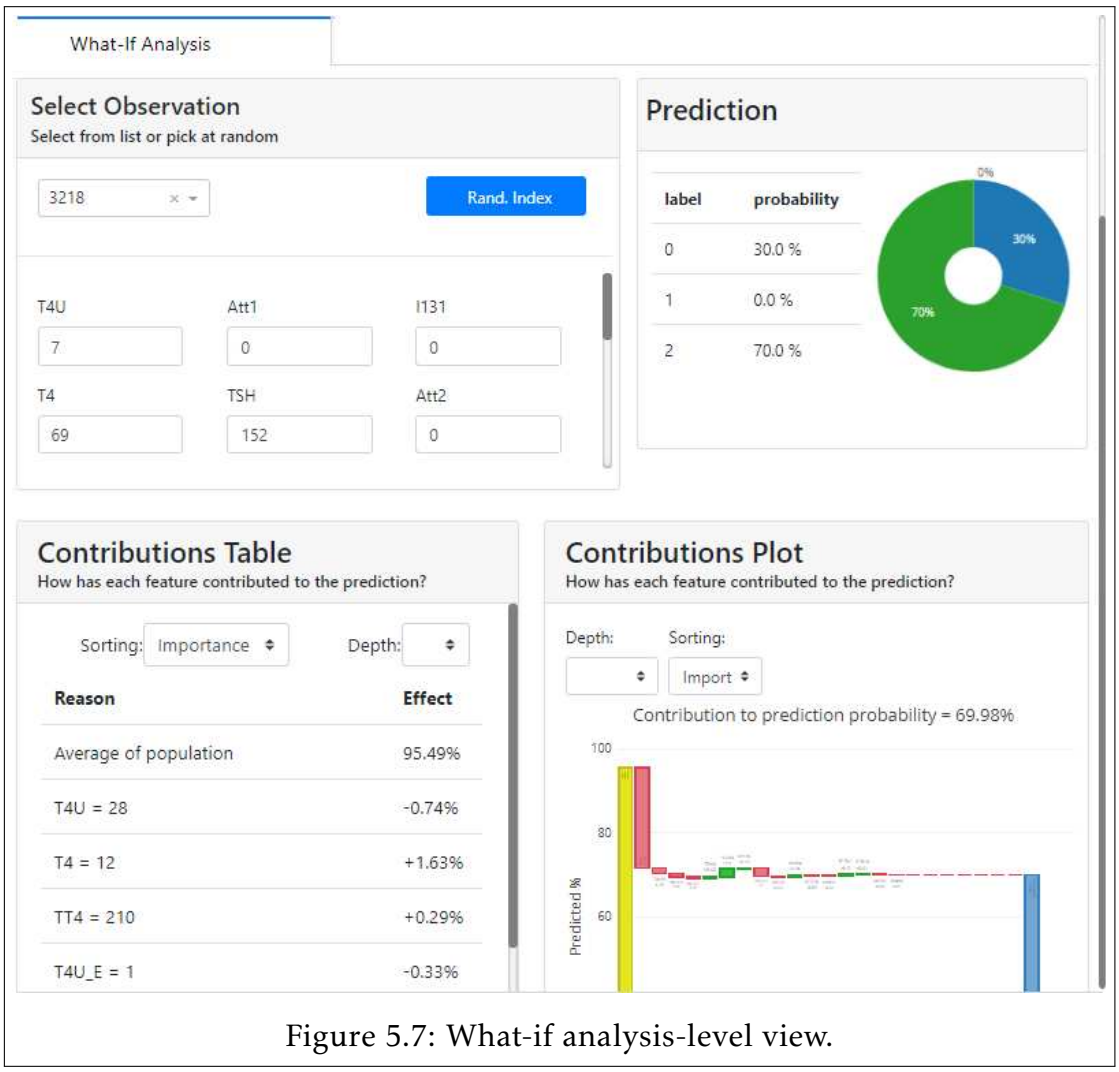

Figure 5.7: What-if analysis-level view.

5.5.4 The refinement-level View

The refinement-level view : (Figure 5.8) shows the correlation between performances and hyperparameters of a recommended algorithm. To accomplish that, we takes as input performances data gathered with different hyperparameters settings of the algorithms (from the recommender module's knowledge-base) , fits a random forest model to capture the relationship between hyperparameters and performances, and then we apply a functional Analysis of variance (ANOVA [223]) to assess how important each of the hyperparameters and each low-order interaction of hyperparameters is to the performance. Guided by this in-depth analysis, end users have a guidance, when AutoML returns unsatisfying results, to improve to predictive performances. In the following we give an overview on how the functional ANOVA is used to efficiently compute the importance of all hyperparameters.

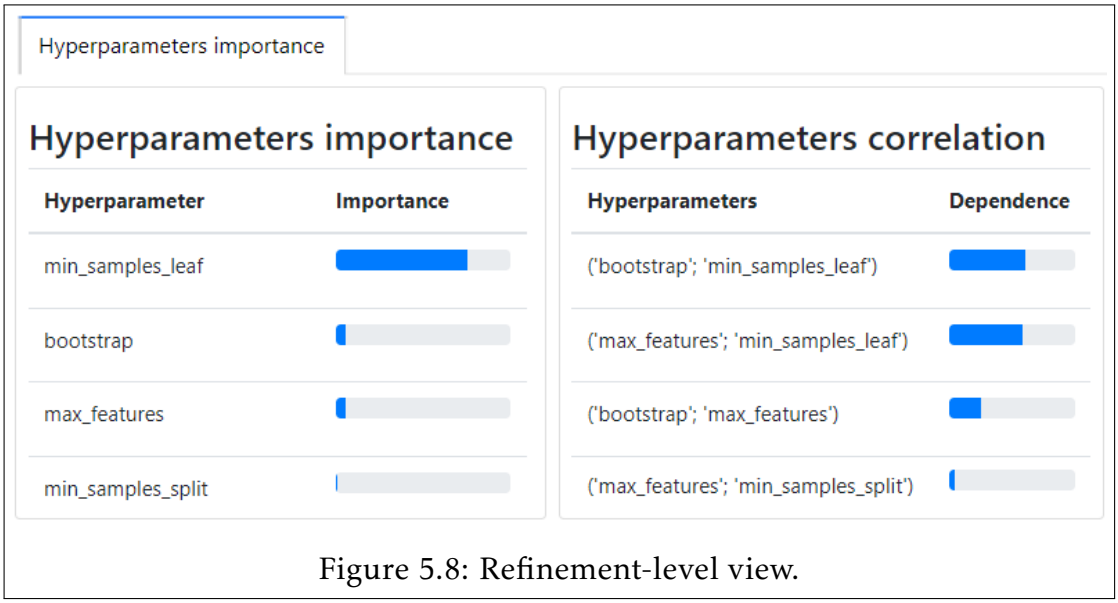

Figure 5.8: Refinement-level view.

Assessing hyperparameters importance

The functional ANOVA framework is an efficient technique for assessing the importance of hyperparameters of a machine learning algorithm based on the efficient computations of marginal performance. More specifically, functional ANOVA specifies the contribution of each hyperparameter to the variance of the ML algorithm performance. We address the following problem, given :

- an algorithm $\mathcal{A}$ with n hyperparameters in the hyperparameters space $\mathcal{H}$;

- a large number of datasets $\mathcal{D}_1,\ldots,\mathcal{D}_m$, with m being the number of datasets (in our study, $m = 400$);
- for each of the datasets, a set of empirical performance measurements $\langle H_n, y_i\rangle_{i=1}^{k}$ for different hyperparameter settings $H_n \in \mathcal{H}$, where y_i is the performance of the algorithm $\mathcal{A}$ measured by the considered performance measures (i.e., Accuracy, Precision, Recall, F1 score);
- the *marginal performance* $\hat{a}_n(H_n)$ is defined to be the average performance of the algorithm $\mathcal{A}$ for all complete configurations H_n that have in common $\mathcal{H}$;

We apply the functional ANOVA on each of the 08 considered classifiers as follows. First we collect performance data $\langle H_n, y_i\rangle_{i=1}^{k}$ for each algorithm $\mathcal{A}$ with the $k = 1000$ different configurations (cf. section 3.4.3). Next, we fit a random forest model to the performance data and then use the functional ANOVA to decompose the variance in performance of the random forest $\hat{y} : H_1 \times H_1 \ldots H_n \rightarrow \mathbb{R}$ into additive components that depends on subsets of the hyperparameters H_n :

$$\hat{y}(H) = \sum_{u \in n} \hat{f}_u(H_u) \tag{5.1}$$

where the components $\hat{f}_u(H_u)$ are defined as follows :

$$\hat{f}_u(H_u) = \begin{cases} \hat{f}_\emptyset & if\, u = \emptyset \\ \hat{a}_u\,(H_u) - \sum_{w \in u} \hat{f}_w(H_w) & \text{otherwise} \end{cases} \tag{5.2}$$

where $\hat{f}_\emptyset$ is the mean value of the function $\hat{y}$ over its domain. The unary function $\hat{f}_{\{j\}}(H_{\{j\}})$ captures the importance of the hyperparameter j average over all possible values for the rest of the hyperparameters, while $\hat{f}_{\{u\}}(H_{\{u\}})$ captures the interaction effects between all hyperparameters in $\mathcal{H}$. The functional ANOVA decomposes the variance V in the $\hat{y}$ into the contributions V_u of all possible subsets of hyperparameters SH_u of the algorithm $\mathcal{A}$.

$$V = \sum_{u \in n} V_u\,, \quad Where \quad V_u = \frac{1}{||H_u||} \int \hat{f}_u(H_u)^2 dH_u \tag{5.3}$$

The importance of an hyperparameter or a set of hyperparameters is captured by the fraction of the variance of the hyperparameter or the set of hyperparameters is responsible for; *the higher the fraction, the more important the hyperparameter or the set of hyperparameters is to the model*. Thus, such a hyperparameter should be tuned further in order to achieve a good performance.

5.6 Conclusion

There has been significant progress in democratizing the application of ML to non-experts of data analysis by providing them with "off the shelf" solutions. However, these powerful support systems fail to provide detailed instructions about the recommended configurations and the inner working of these models, thence making them less trustworthy highly performant black-boxes. In this chapter, we presented an interactive visualization toolbox that supports machine learning experts and neophytes in analyzing the automatic results of an AutoML DSS.

To our knowledge, the proposed toolbox is the first application of the general explanation methods of AutoML systems as decision support systems. We explored several levels of explanations, ranged from individual decisions to the entire model's recommendations and predictions. The explanations of the predictive models and what-if analysis proved to be an effective support for manufacturing related problems. In the next chapter, we present the materialization of the self-explainable AutoML tool as an open source software package. A set of evaluations demonstrate the utility and usability of AMLBID in a real-world manufacturing problem. We show how powerful black-box ML systems could be made transparent and help domain experts to iteratively evaluate and update their beliefs.

Chapter 6

AMLBID : A self-explained AutoML software package

Outline of the current chapter

The growing concern over digital transformation has led to the widespread adoption of machine learning solutions. Although, in most of the current systems, the ML sufficiently assists the large data analysis for the decision-making purposes but the human expertise is often required. The large number of algorithms and hyperparameters configurations could make infeasible exhaustive search executions. Therefore, expert data-scientists are highly desired. The identification

of the most appropriate algorithm in an automatic manner is among the major research challenges to achieve optimal performance of ML tools. In this chapter, we present the open source AMLBID software package. A python based decision support system for automated selection and tuning of implied hyperparameters for machine learning algorithms to cope with the prominent challenges posed by the evolution of industrial big data. Furthermore, the tool is equipped with an explainer module that makes the outomes rather transparent and interpretable for well-performing ML systems. Being based on meta-learning, the tool is able to simulate the role of the machine-learning expert as a decision support system.

6.1 Motivation and significance

The Machine Learning based solutions have achieved a great success in online advertising, recommender systems, bioinformatics, manufacturing and many other fields. In almost all of these successful ML applications, skilled resources are involved in all ML stages including : transforming real world problems into machine learning tasks, collecting data, performing features engineering, selecting or designing the model architecture, tuning model hyperparameters and evaluating model performances. As the complexity of these tasks is often beyond non-experts, the rapid growth of ML applications has created a demand for off-the-shelf methods and solutions that can be used easily without expert knowledge. The algorithms selection and configuration is one of the most difficult tasks in a ML pipeline. The identification of the most appropriate algorithm in an automatic manner is among the major research challenges to achieve optimal performance of ML tools.

The selection of an algorithm or a family of algorithms that are more likely to perform better on a given combination of datasets and their evaluation measures is a challenging task [17]. The algorithms selection and configuration (tuning of hyperparameters) is a complex process as mostly the ML algorithms are used as a "black box". The performance of such algorithms is affected by multiple characteristics of the dataset and hyperparameters [18]. It is therefore, the complexity of the selection and configuration of appropriate algorithm(s) is an error prone and time-consuming process due to the prevailing flaws while establishing the multiple configurations. It hence emphasizes the need to automate this process.

Owing to the immense potential of AutoML, multiple approaches have been proposed to tackle the above problem (cf. Section 2.1). In this context several tools are available in the research community such as Auto-sklearn [18], AutoWEKA [224], and TPOT [71]. However, the lack of explanations for the predicted performance factors makes them typically black-box solutions. These

also involve the computational complexity of these solutions and the great deal of time required to generate recommendations [17]. As such, they only allow the prominent exhibition of input and output parameters, but conceals the visibility of inherent associations among them. Users tend to raise objections to the validity of automatic results due to the lack of transparency in AutoML systems with respect to the Decision Support Systems (DSS). Therefore, the acceptability and the trust in, an AutoML support system are highly dependent on transparency in the recommendation generation process [146].

In this chapter, we show the materialization of our contributions from the previous chapters into a software package. In this context, `AMLBID` [144] is a transparent, interpretable and self-explainable meta-learning based tool for recommending the optimal or near-optimal ML configurations for a given problem, and for explaining the rationale traceability behind a recommendation.

6.2 Software description

Given a predictive modeling problem for an industrial application, it is often difficult to build an accurate predictive model based on machine learning that is easy to be interpreted by ML non-experts [144, 225]. The key idea behind the transparent and auto-explainable automated machine learning vision is to separate the recommendations from the explanations by using two modules simultaneously, as shown in Figure 6.1. The *Recommender module (AMLBID)* for recommending and the *Explanatory module (AMLExplainer)* for explanations. The first module is used to provide the most appropriate ML configurations for a given problem. It is aimed at maximizing the requested predictive metric (e.g. Accuracy, Recall, Precision). The second module is used for providing the rationale behind the recommended ML configurations as well as automatically explaining the insight workings of the model in an interpretable manner through an interactive multi-view toolbox (cf. Chapter 5).

6.2.1 Software architecture

The workflow of the proposed self-explanatory AutoML system consists of two main components : the AutoML component, which presents the AutoML process at the abstract-level (from ML pipelines recommendation to the refinement), and the explanatory one, which allows users to inspect both the process of decision generation and the inner working of the recommended models.

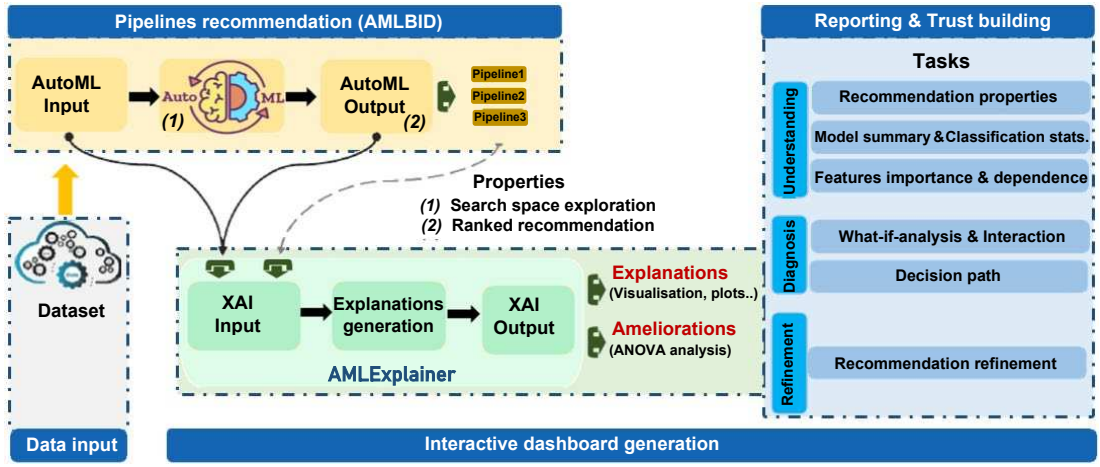

Figure 6.1: The global architecture of the proposed white-box AutoML system.

The recommendation module

The Automated Machine Learning tool for Big Industrial Data (AMLBID) is a meta-learning based system in order to automate the problem of algorithm selection and configuration. It uses a recommendation system that is bootstrapped with a knowledge base. The actual knowledge base is derived on a large set of experiments conducted on *400* real-world manufacturing classfication datasets. These are collected from the popular repositories, such as the OpenML[1], UCI[2] and Kaggle[3], among other real world scenarios. It accumulates the generation of more than *4 million* evaluated ML configurations (pipelines). Each pipeline consists of a choice of a machine learning model and the configuration of its hyperparameters (cf. Chapter 3). The system is able to identify effective pipelines without performing expensive computational analysis through exploring the interactions between datasets's meta-features (characteristics) and pipelines topology.

The recommendation phase is initiated with the arrival of a new dataset to be analyzed. At this stage, the user selects a predictive analytical metric (e.g. *Precision*, *Accuracy*, *Recall*) to be used for the analysis. Then AMLBID automatically provides a set of machine learning algorithms and intended configuration of their related hyperparameters, so that the predictive performance becomes the first-rate.

[1] https://www.openml.org/
[2] https://archive.ics.uci.edu/
[3] https://www.kaggle.com/

Explainer module

AMLExplainer is implemented on a client-server architecture along with its integration with the recommendation module. The server coordinates as the AutoML support system (i.e. AMLBID), while the client-side visual interfaces provide graphical interactions with AutoML services, which maps the data summaries to the visualizations through a set of multiple visual summary levels of the recommended models. Meanwhile, AMLExplainer guides the end-users, in case of the unsatisfying results returned from AutoML, intended to improve the predictive performances (cf. Chapter 5). Hence, it may increase the transparency, controllability, and reliability of AutoML DSS.

6.2.2 The software Functionalities

AMLBID is a Python-package representing a meta-learning based framework intended to automate the process of algorithm selection and the tuning of hyperparameters in supervised machine learning. In the literature we observe that the majority of state-of-the-art tools evaluate a set of pipelines by actually executing them on a given dataset prior to the recommendation. It can be noted that such executions may require considerable computing time while consuming precious resources as per their availability [144]. The proposed system (AMLBID) immediately produces a list of potential top-ranked pipelines using its knowledge base at an imperceptible computational time, hence it notably economises resource cost and their provisional availability. In particular, AMLBID is considered as the pioneer open-source, transparent and auto-explainable AutoML system for recommending the most adequate ML configuration for a given problem. It guides the end-users for improving the utility and usability of the AutoML process with the following main features :

- It automatically selects the most appropriate ML pipelines through the use of a collaborative knowledge base that is continuously improved over time by running more tasks. It makes AMLBID smarter by attaining more experience, based on the growing knowledge base.
- The framework is equipped with an explanation module, which allows the end-user to explore and understand the diagnostic design of the returned ML models using various explanation techniques in a trustful manner, through linked visual summaries—textual information for a higher trust.
- It provides the assistance when AutoML returns unsatisfying results, in order to improve the predictive performances by assessing the importance and the correlation between the algorithm hyperparameters.

Therefore enabling end users to rapidly ask a serie of what-if scenarios when probing opportunities to use predictive models to improve outcomes and reduce costs for various tasks as well as the need of classical collaborations.

6.3 Illustrative Example

AMLBID broadly has the AMLBID_Recommender module for recommending and building highly-tuned ML pipelines and the AMLBID_Explainer module to intercept the inner working of the generated pipeline(s). These are described in the following sub-sections.

6.3.1 Recommender module

Script 6.1 summarizes the interactions required to use AMLBID to recommend a pipeline. Subsequently, it attributes a score to the chosen pipelines and export the best pipeline to a dynamically stored .py file.

```
1  from AMLBID.recommender import AMLBID_Recommender
2  from AMLBID.loader import *
3
4  #Load dataset
5  Data,X_train,Y_train,X_test,Y_test=load_data("Dataset.csv")
6
7  #Generate the optimal configuration
8  model=AMLBID_Recommender.recommend(Data, metric="Accuracy",
9                                     mode="Recommender")
10 model.fit(X_train, Y_train)
11
12 print(model.score(X_test, Y_test))
13
14 #Export configuration's corresponding Python code
15 model.export('Recommended_pipeline.py')
```

Listing 6.1: Illustrative code example of recommendation module.

On line 5, we define the root directory of the dataset to be loaded. The recommend function (as shown on line 8) initializes the meta-learning process to find the highest-scoring pipeline according to the desired performance criterion. Then, the recommended pipeline is trained on the test-set of the provided samples (as shown on line 10). Once this code finishes its execution, Recommended_pipeline.py (shown in script 6.2) shall contain the corresponding Python code for the optimized pipeline using the export function (as shown on line 15).

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

data = pd.read_csv("Dataset.csv")

X = data.drop('class', axis=1)
Y = data['class']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
     test_size=0.3, random_state=42)

model= DecisionTreeClassifier(criterion='entropy',
                              max_features=0.5672564,
                              min_samples_leaf=5,
                              min_samples_split=20)

model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)
score = model.score(X_test, Y_test)

print(classification_report(Y_test, Y_pred))
print(' Pipeline test accuracy:  %.3f' % score)
```

Listing 6.2: Generated python file.

6.3.2 Explainer module

The `AMLBID_Explainer` module allows users to inspect the insight working of the recommeded model and the decision generation process. Its use is illustrated in script 6.3. It provides explanations on several levels of abstraction like feature importance, feature contributions to individual predictions (such as the SHAP tool that provide the interaction shapely values of a contribution for some prediction [226]), "what-if" analysis, visualization of individual decision path, hyperparameters importance, and correlations as shown in Section 5.5 of Chapter 5.

6.4 Impact

In practice, the machine learning modeling process is a highly iterative exploratory process. In particular, there is no one-size-fits-all model solution, i.e,

```
from AMLBID.recommender import AMLBID_Recommender
from AMLBID.explainer import AMLBID_Explainer
from AMLBID.loader import *

#Load dataset
Data,X_train,Y_train,X_test,Y_test=load_data("Dataset.csv")

#Generate the optimal configurations
model,config=AMLBID_Recommender.recommend(Data,
                                          metric="Accuracy",
                                    mode="Recommender_Explainer")
model.fit(X_train, Y_train)

#Generate the interactive explanatory dash
Explainer = AMLBID_Explainer.explain(model, config,
                                     X_test, Y_test)
Explainer.dash()
```

Listing 6.3: Illustrative code example of recommendation_explainer module

there does not exist a single model or algorithm which can be used to achieve the highest accuracy for all datasets varieties in a certain application domain. Hence, trying many ML algorithms with different hyperparameters configurations is usually considered as an inefficient, tedious, and time consuming process.

The main objective of the AMLBID has been focused towards the design of a decision support system in order to enable the non-expert practitioners and researchers, prospectively in the domain of industry 4.0 to take maximum benefit of ML techniques. In [36, 227], we studied the effectiveness of the recommender module for the selection and parameterization of ML for the problems more often related to the manufacturing industry. The evaluation results respond the basic research question that how some machine learning oriented manufacturing works could be further improved, simply through the use of a better ML algorithm configuration using the AMLBID decision support system. Since AMLBID is built upon the meta-learning concept, in the broader sense, it is not only beneficial for the manufacturing actors and researchers but also for the general public.

In this context, the application of AMLBID is twofold: primarily it makes possible for non-data science specialists (engineers and researchers) to build robust ML pipelines without the need for specialist's assistance or intervention and even having to write a single line of code. Subsequently, the white-box specificity of the proposed AutoML tool makes it possible to interactively inspect the inner-workings of the ML predictive models without having to depend on a data scientist to generate and interpret all the extreme plots and tables. Finally,

the AMLBID is useful for academic purposes, helping academia to build and understand ML predictive models behavior. The complete documentation and a detailed list of features with an illustrative example are available in the Github repository[4] and can also be installed easily via the Python package manager.

6.5 Utility and usability study

The following section describes the evaluation methodology of the proposed auto-explainable AutoML tool in an empirical evaluation with humans. We globally draw the insights from the feedback that we have received from the various target users.

6.5.1 Demonstration test case : application to manufacturing quality prediction

The proposed system is designed for the machine learning based predictive modeling problems. The major goal of the work has been to show the feasibility to achieve maximum possible performance for a specific predictive modeling problem, and automatically explain the results for any machine learning predictive model. We evaluate the intuitiveness and the usability of the automatic explanation method on a Manufacturing Quality Prediction use case (real-life environment). The data contains 187.156 historical 1-year records of a production unit. Among these records, 74.39% are diagnosed as compliant products.

6.5.2 User interview

Participants and Apparatus To evaluate the proposed white-box AutoML system as a decision support system, we conducted a semi-structured qualitative user study with two different groups of target users. These groups range from ML novices to experts (53% male and 47% female who are aged between 24 and 38 years with an average age of $\mu = 26.78$ years). Among the ML-users, 48% were the participants with particular knowledge in the industrial big data analysis. While, among the ML-experts, 52% were the participants with experience in developing ML models for their domain problems. All of these participants have the experience in ML or data analysis, but none of theme had prior experience with AutoML. The evaluation studies are conducted on a set of dedicated computers equipped with an Intel Core i5-3,10GHz - 8Go RAM DDR4.

[4] https://github.com/LeMGarouani/AMLBID

Tasks and Procedure The evaluation study began with a tutorial session, in which the tasks and the usage of the self-explainable AutoML system were introduced to the participants. Participants were asked to complete the Post-Study System Usability Questionnaire (PSSUQ), third version [228], when performing the explanatory module on the three tasks *understand*, *diagnose*, and *refine*.

The PSSUQ-3 is a 16 item measure (as shown in the questionnaire sheet 6.4). The questionnaire consists of an overall satisfaction scale (the mean value of items 1 through 16) and three subscales. *System usefulness* subscale assesses the ease of learning and use of the system (i.e. the mean value of items 1 through 6). The *information quality* subscale evaluates the feedback provided by the system to the user (i.e. the mean value of items 7 through 12). Finally, the *interface quality* subscale quantifies the familiarity of the user with the system, as whether the system has met the the expected functionality (i.e. the mean value of items 13 through 16). For all the scales, the rating range was between 1 and 7; *the lower the score, the higher the satisfaction with the tool*.

		Strongly agree						Strongly disagree
		1	2	3	4	5	6	7
1	Overall, I am satisfied with how easy it is to use this system.	○	○	○	○	○	○	○
2	It was simple to use this system.	○	○	○	○	○	○	○
3	I was able to complete the tasks and scenarios quickly using this system.	○	○	○	○	○	○	○
4	I felt comfortable using this system.	○	○	○	○	○	○	○
5	It was easy to learn to use this system.	○	○	○	○	○	○	○
6	I believe I could become productive quickly using this system.	○	○	○	○	○	○	○
7	The system gave error messages that clearly told me how to fix problems.	○	○	○	○	○	○	○
8	Whenever I made a mistake using the system, I could recover easily and quickly.	○	○	○	○	○	○	○
9	The information (such as online help, on-screen messages, and other documentation) provided with this system was clear.	○	○	○	○	○	○	○
10	It was easy to find the information I needed.	○	○	○	○	○	○	○
11	The information was effective in helping me complete the tasks and scenarios.	○	○	○	○	○	○	○
12	The organization of information on the system screens was clear.	○	○	○	○	○	○	○
13	The interface of this system was pleasant.	○	○	○	○	○	○	○
14	I liked using the interface of this system.	○	○	○	○	○	○	○
15	This system has all the functions and capabilities I expect it to have.	○	○	○	○	○	○	○
16	Overall, I am satisfied with this system.	○	○	○	○	○	○	○

Figure 6.4: The Post-Study System Usability Questionnaire.

The results of the usability questionnaire are summarized in Table 6.1 and Figure 6.5. The PSSUQ overall and subscale scores were extremely positive, with an overall total mean score of 1.53 (standard deviation 0.71) and a range from strongly agree to neutral (1 to 5). In this context, the mean system usefulness,

information quality, and interface quality subscale scores were 1.74, 1.09, and 1.22 respectively.

PSSUQ	Score
Total score	1.53 ± 0.71 (1 – 5)
System usefulness subscale	1.74 ± 0.83 (1 – 5)
Information quality subscale	1.09 ± 0.37 (1 – 3)
Interface quality subscale	1.22 ± 0.95 (1 – 4)

Table 6.1: Post-Study System Usability Questionnaire (PSSUQ) overall and subscale scores of the decision support tool. Data are presented as : score ± standard deviation (range).

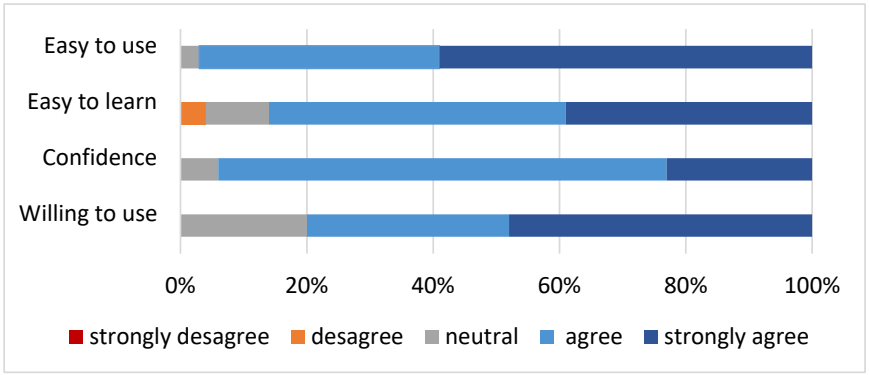

Figure 6.5: Results of the usability test.

As shown in Figure 6.5, most of the participants agreed that the auto-explainable AutoML DSS is easy to learn and use. Among them, 80% of the participants strongly agreed that they are confident in their recommended model(s). We also conducted semi-structured post-study interviews to gather more detailed feedback from participants. The interviews reflect the difference between the initial expectation and the experience during the pair analytic regarding the workflow of the system.

We collected the participants feedback about the AutoML module as a black-box decision support system assisting the experts to choose and configure ML models for their problems initially and afterwards with the entire system (recommendation module and the explanatory one). Based on their feedback, we summarize two main appreciations of the proposed software :

- AutoML can help stakeholders (neophyte as well as experts) to improve the applications of machine learning algorithms. AutoML enables quick

experimentation with a large number of models and configurations, whose results could provide useful knowledge to ML researchers and domain practitioners. On the test set, the recommended machine learning predictive model configuration achieved an accuracy of 97.81% while their configuration based on their understanding of the algorithms and their observation of the data achieved a predictive accuracy of 91.42%. These findings can inform the users about the importance of hyperparameters tuning for ML algorithms. The participants highlight the fact that being able to match prior knowledge about machine learning to the visualizations produced by AMLExplainer creates confidence in the underlying AutoML process and increases the likelihood of adopting AutoML.

- The participants appreciated the human-machine interaction introduced in AMLExplainer. They observed such interaction could improve an AutoML process and enhance user experience and make such powerful black-boxes trust worthy. One of the experts commented : *"Users with more domain knowledge, such as myself, are usually critical of automated methods and like to be in control. I do not like getting a score back and hearing* trust me*"*.

Overall, the feedback on the system remains positive. In addition, the users provided several suggestions for complementary features. For the *understanding and diagnosis* tasks, in addition to the provided explanation levels, users wanted to gain insight into the underlying data. Such exploratory data analysis feature [229] is an integral part for any knowledge discovery process. For example, a data profiling level could review the dataset characteristics and quality and show it to the user. For results reporting, the feedback is mostly unified. All participants liked the *code export* function of the recommended ML pipeline and they are aspirant to use that for the communication of their results. Furthermore, in the perspectives of the code export, participants came up with several suggestions for enhancing this feature, such that possibility to store /export the overall explanation levels as PDF report file.

For the *refinement* task, there are mixed feedback and expectations. Most of the participants are optimistic and they suggest additional ways to interactively refine the recommended model(s). Rather than providing static guidance content, some individual non-ML experts ask further guidance to select the appropriate refinements. Moreover, the ideas to enhance its functionality include the propositions of code fragments, providing building blocks, or even scaling it up to a *click-to-refine* functionality. In the future, the current system should be extended with the suggested refinement methods and additional guidance to select the appropriate refinements.

Our documentation of the real-world evaluation case illustrates how to overcome the transparency problem of AutoML systems as decision support systems.

For instance, the absence of human interaction and analysis of the inner working and reasoning of such tools. This could extend the *use-of* and *trust-in* the intelligent AutoML systems to areas where they are so far neglected due to their insistence on comprehensible models. Separating the automatic selection and configuration of machine learning algorithms from model explanation is another benefit of expert and intelligent AutoML DSS.

6.6 Conclusion

The machine learning based applications are increasingly desired due to their robustness for the large data analysis. Also, they can rapidly integrate “off-the-shelf" solutions in multiple areas. However, the non-expert data analysts are more inclined to adapt the ML based solutions that are more easily persuadable, among diverse algorithms, with the help of their rational traceability. We argue that the adaptability of the powerful decision support systems based on the ML based solutions can be further enhanced with the help of comprehensive instructions regarding the recommended pipelines and their insights. Thus, making them more trustworthy instead of black-box solutions.

Aiming to bring our contributions in this thesis into real life practice, we presented the requirements, architecture, characteristics and components of the self-explainable AutoML software packge developed in this thesis. It is a novel transparent and auto-explainable AutoML support system. To our knowledge, the proposed system is the first application of the general explanation methods of AutoML systems as decision support systems. A set of evaluations demonstrate the utility and usability of AMLBID in a real-world manufacturing problem. We show how powerful black-box ML systems can be made transparent and help domain experts to iteratively evaluate and update their beliefs.

Part III

Conclusion

Chapter 7

Conclusion

Outline of the current chapter

In this chapter, we present the conclusions of our work and highlight the current challenges along with suggesting future research directions.

7.1 Conclusion

The increasing data availability has fueled the popularity of ML-based solutions able to relieve humans from many risky, repetitive and tedious activities. In many cases, this requires that ML algorithms are used in new and innovative ways. This development process is heavily based on human experts to perform manual tasks such as data preprocessing, features engineering and evaluation of several possible ML algorithms. Most of the employed ML algorithms have hyperparameters, which usually affect their predictive performance. Although HPs tuning may lead to more accurate models, the optimization process for finding these HPs settings is still very time-consuming. Nevertheless, there is no guarantee that tuning will generate better results than just using the default HPs settings provided by ML packages and tools (Chapter 1). Therefore, when the technical expertise and computational resources are limited, knowing beforehand which ML algorithm is more adequate can reduce the computational cost

of the ML task and increase productivity. It is also important to know which HPs values to set when tuning is required. Hence, as the complexity of such processes increases, so does the demand for automated support solutions that can be used easily and without high human intervention.

In this sense, the so-called MtL and AutoML can increase the widespread use of efficient ML solutions and free data scientists and practitioners from repetitive and time-consuming tasks. Thus, more applications can benefit from the use of ML, and data scientists can allocate their time on more creative and important tasks than fine tuning algorithms. The goal of this dissertation is to provide support to the analyst in selecting the most appropriate learning algorithm for a classification problem refraining from the tedious task of systematic experimentation with various learning algorithms.

As a first step to that goal, the survey part of this dissertation present a thorough overview of important dimensions of meta-learning for the algorithms selection and answer the research questions that were formulated on three important dimensions i.e.,*meta-features*, *meta-models* and *meta-targets*. Related works from literature are summarized and critically analyzed in this regard (Chapter 2).Within this approach our work spans the whole range of tasks required for the solution of a typical classification problem. That is, we searched for an appropriate formulation of the meta-learning space, and we constructed it in such a way so that it closely simulates the steps followed by the analyst when he has to select among different learners (Chapter 3).

As the performance of algorithms recommendation methods is largely dependent on the quality of meta-features, special care was given to the meta-features extraction part of the process, in order to have a set of characteristics that can best discriminate between different datasets and the inherent biases of various candidate algorithms. A step that involved the conception of new latent meta-features with lower dimentionality but more significant and meaningful data characteristics. We proceeded to a systematic experimentation of different learners on the meta-level and compared the set of characteristics that we established with sets of characteristics from previous similar works (Chapter 4). We thoroughly made advances towards the acceptance of and the trust in AutoML as black boxes support systems (Chapter 5).

To this end, we materialized our contributions into the transparent, interpretable and auto-explainable AutoML software package AMLBID, that given a classification algorithm recommends transformations that positively impact the analysis (Chapter 6). We extensively evaluated our recommendations from three perspectives. In the first one, we checked how accurate our predictions were. In the second, we analyzed how much gain they provided to the final non-experienced user. Finally, in the third, we analyzed the performance of

AMLBID compared to humans in a realistic utility and usability scenario.

7.2 Publications

Several papers have been published during the development of the research for this thesis. Hence, part of the results reported throughout this thesis can be found in these publications, as presented :

Journals

- Moncef Garouani *et al*. "Towards big industrial data mining through explainable automated machine learning". In: *The International Journal of Advanced Manufacturing Technology* (2022). doi :10.1007/s00170-022-08761-9

- Moncef Garouani *et al*. "AMLBID: An auto-explained Automated Machine Learning tool for Big Industrial Data". In: *SoftwareX* 17 (2022). doi :10.1016/j.softx.2021.100919

- Moncef Garouani *et al*. "Using meta-learning for automated algorithms selection and configuration: an experimental framework for big industrial data". *Journal of Big Data* 9, 57 (2022). doi :10.1186/s40537-022-00612-4

- Moncef Garouani *et al*. "Autoencoder-kNN meta-model based data characterization approach for an automated selection of AI algorithms". [submitted to *Journal of Big Data*]

- Moncef Garouani *et al*. "AMLBID2.0: An auto-explained Automated Machine Learning tool for Big Industrial Data". [submitted to *SoftwareX*]

International Conferences

- Moncef Garouani *et al*. "Towards the Automation of Industrial Data Science: A Meta-Learning Based Approach". In: *23rd International Conference on Enterprise Information Systems*. 2021, pp. 709–716.doi :10.5220/0010457107090716

- Moncef Garouani *et al*. "Towards meta-learning based data analytics to better assist the domain experts in industry 4.0". In: *Lecture Notes on Data Engineering and Communications Technologies* (ICABDE'21). Springer, Cham. doi :10.1007/978-3-030-97610-1_22

- Moncef Garouani *et al*. "Towards an Automatic Assistance Framework for the Selection and Configuration of Machine-Learning-Based Data Analytics Solutions in Industry 4.0". In: *The Fifth International Conference on Big Data and Internet of Things* (BDIoT'21). [**In press**]

Posters

- Moncef Garouani *et al*. "Towards industrial data science through explainable automated machine learning". POSTER In *MTE Pole's Doctoral Day*(2021), ULCO University, Calais, France
- Moncef Garouani *et al*. "Towards explainable Automated Machine Learning". POSTER In *IA² – Institut d'Automne en Intelligence Artificielle* (2021), Sorbonne Université, Paris, France

7.3 Challenges and future directions

While, collectively, the community has made significant progress towards efficient automated machine learning, there remains important outstanding directions for future research. Externally, the *3 Vs*— Volume, Velocity and Variety impact of big data associated with the large-scale machine learning and the shortage of ML talent present tough challenges but also opportunities for advances in AutoML to make an outsized impact. We present some thoughts from our work in terms of challenges and future research directions.

- One of the limitations of developing MtL-based AutoML systems is the computational cost associated with the evaluation of all the candidate algorithms in the algorithm space $\mathcal{A}$ on all the datasets in the problem space $\mathcal{P}$. Furthermore, increasing the number of candidate algorithms or datasets increase the computational cost of enriching the meta-knowledge base. This hinders the rigorous exploration of the algorithms and problems space. Although we used a reasonable number of candidate algorithms and datasets, still the initial computational cost is high for enhancing the system. This drawback can be covered in future works by incorporating data in the meta-knowledge base from platforms like Kaggle[1] and OpenML[2]. Such collaborative data science platforms have on-line data repositories in which analyts share their experimental results from the application of algorithms on datasets. It would not only reduce the computational

[1] `https://www.kaggle.com/`
[2] `https://www.openml.org/search?type=run`

cost but will also provide the opportunity to rigorously explore the problem and algorithm space and will ensure that no bias is induced into the system at the meta-level.

- Another possibility of research is to expand the idea of AutoML system covered in this thesis. In this case, the system would not only recommend ML algorithms but also preprocessing methods, and post-processing analysis able to explain the experimental results. Thus, given a new dataset, the AutoML system would recommend the most suitable components to solve the input problem. In fact, we have already begun work in this direction.

- Experiments described in Section 4.4.3 also open new possibilities for Neural networks use in MtL. Given the results reported there, a NN system could be developed to recommend which ML pipeline to use. Thus, a more focused tuning could be performed using a smaller number of evaluations and only the most important HPs.

- Finally, our study was limited to classification problems. That is, we are providing user support in ML algorithms selection and parametrization only if the problem at hand is of classification type. However, our method can be directly extended to regression, clustering and distributed ML libraries (e.g., SparkML [230]) since we are dealing with industrial big data.

Chapter 8

Résumé étendu en Français

Outline of the current chapter

8.1 Introduction

La récente explosion des masses de données en termes de volume, de variété et de vitesse a conduit à un grand développement d'outils, de méthodes et de modèles d'apprentissage automatique à grande échelle. Le domaine de l'apprentissage automatique est en constante évolution et produit une multitude de modèles et d'algorithmes pour effectuer des tâches d'analyses avancées, tels que les arbres de décision, les réseaux de neurones, les inducteurs de règles, le plus proche voisin, etc. Cependant, en général, la capacité d'analyser les données est loin derrière la capacité de les collecter. Cela est dû au fait que l'analyse de données consiste en plusieurs étapes difficiles et chronophages, qui ont été regroupées comme suit [4] : *sélection des données*, *prétraitement des données*, *exploration des données et évaluation/interprétation des résultats obtenues* (Figure 8.1).

Une fois qu'un problème d'apprentissage est défini, l'analyste doit trouver les outils d'apprentissage adéquats pour le résoudre. La qualité des données disponibles est l'un des facteurs les plus cruciaux pour obtenir une solution de haute performance. Cependant, nous n'entrerons pas dans les détails de la collecte des données et de la qualité des données disponibles. La tâche principale de

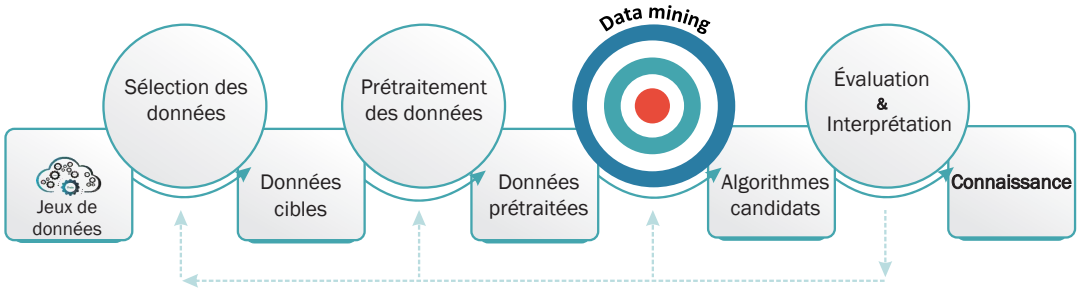

Figure 8.1: Processus d'analyse de données.(répété à partir de la page 2)

l'analyste sera de *sélectionner* et de *paramétrer*, parmi ces modèles et algorithmes, ceux qui correspondent le mieux à la morphologie et aux caractéristiques spéciales d'un problème donné. Cette sélection qui est conduite souvent par des experts de différents domaines ayant peu d'expérience en science des données est un problème extrêmement fastidieux étant donné qu'il n'existe pas de modèle ou d'algorithme qui ait une meilleure performance que d'autres indépendamment des caractéristiques spécifiques du problème, comme cela a été confirmé dans différentes comparaisons empiriques par différents théorèmes du type "no free lunch" [16, 231].

Chaque algorithme a une "supériorité sélective", càd qu'il est meilleur que les autres pour un type de problèmes particulier [231]. Ceci est dû au fait que chaque algorithme à ce que l'on appelle un "biais inductif" engendré par les hyperparamètres faites afin de généraliser d'une donnée d'entrainement à des exemples jamais vus auparavant. Selon Michel Lutz [232], "le biais inductif d'un algorithme d'apprentissage est l'ensemble de toutes les hyperparamètres requises pour justifier ses inférences inductives comme étant des inférences déductives". Donc, l'analyste doit posséder beaucoup d'expérience pour pouvoir identifier la configuration des hyperparamètres de l'algorithme le plus approprié à la morphologie du problème posé (cf. Chapitre 1).

La tâche de sélection et de paramétrisation des algorithmes d'apprentissage automatique est une tâche itérative. L'analyste doit tout d'abord sélectionner un algorithme ou une classe d'algorithmes, par exemple sélectionner entre la classe d'algorithme d'arbres de décision ou la classe d'algorithme des réseaux de neurones. A l'étape suivante on sélectionne un algorithme particulier implémentant une méthode spécifique pour chercher à travers l'espace représentationnel associé au modèle choisi. L'algorithme est ensuite appliqué et la qualité de ses prédictions est évaluée. Si les résultats d'évaluation sont médiocres, le processus est répété à partir du stade antérieur avec de nouvelles configurations ou sélections (cf. Figure 1.1). La procédure d'évaluation est ainsi assez coûteuse en temps

et devient problématique lorsque le volume de données est important. Ceci implique souvent de leur part un investissement important, pour des analyses parfois triviales. On s'intéresse donc à ce besoin d'*assistance* à l'analyse de données, qui, toujours insatisfait, a donné naissance au domaine de la méta-analyse et d'apprentissage automatique automatisé.

L'apprentissage automatique automatisé (AutoML) est devenu un domaine en plein essor qui cherche à sélectionner, composer et paramétrer automatiquement les modèles d'apprentissage automatique afin d'atteindre un niveau optimal de performances sur une tâche et / ou un problème donné. Ce domaine relativement nouveau présente encore de nombreux verrous. Un des plus fondamentaux concerne les définitions de l'analyse de données, qui, dans la littérature, sont nombreuses et divergent selon les méthodes qu'elles englobent. De plus, l'effervescence de nouvelles approches d'analyse nécessite une adaptation constante, au risque d'une obsolescence rapide des méta-analyses incapables de prendre en compte les innovations du domaine. Enfin, l'interaction avec des utilisateurs d'autres disciplines se révèle souvent très complexe, nécessitant un important travail d'adaptation de la part de l'utilisateur pour comprendre les concepts d'analyse manipulés, souvent au détriment de la réalité de terrain qui l'intéresse (cf. Chapitre 2).

Les premières approches d'assistance à l'analyse basés sur le méta-analyse se révélant ainsi souvent similaires et peu abouties (aucune à notre connaissance n'a atteint un réel déploiement). L'objectif principal de notre étude est d'étudier de nouvelles approches de méta-analyse pour adresser ce problème d'assistance à l'analyse de données. Notre objectif, issu d'un cheminement décrit au Chapitre 3, peut se résumer comme suit : Proposer de nouvelles approches performantes de méta-analyse à des fins d'assistance à l'analyse de données notamment adaptée à des utilisateurs n'ayant pas l'expertise technique nécessaire de mener des analyses avancées.

En particulier, cela consistera à prendre avantage de la connaissance que l'on peut avoir du domaine pour recommander des processus d'analyse adaptés au contexte de l'utilisateur, comme illustré en Figure 8.2. Pour ce faire, on dispose d'une connaissance du domaine de l'analyse de données sous la forme d'une base d'expériences passées (voir partie inférieure de la Figure 8.2), chacune représentant l'application d'un processus d'analyse complet (ou workflow) à un jeu de données et en qualifiant les résultats. Afin de trouver dans cette base les expériences passées potentiellement pertinentes pour le problème de l'utilisateur, on peut discriminer selon deux critères :

1. L'une des hypothèses fondatrices du méta-apprentissage stipule qu'un même algorithme exhibera souvent des performances similaires sur des jeux de données semblables [13]. On peut étendre ici cette hypothèse à l'ensemble

du domaine de l'analyse pour supposer qu'un processus d'analyse qui s'est bien comporté sur un jeu de données semblable à celui de notre utilisateur a de bonnes chances d'être également efficace sur ce dernier. On peut alors simplement employer les dissimilarités proposées au chapitre 4 pour trouver ces jeux de données semblables, et donc les processus y ayant été employés (voir 1 en Figure 8.2).

2. Ensuite, on peut exprimer le besoin d'utilisateur par un ensemble d'indicateurs de performances attendus, et chercher parmi les expériences passées celles maximisant ces indicateurs (voir 2 en Figure 8.2).

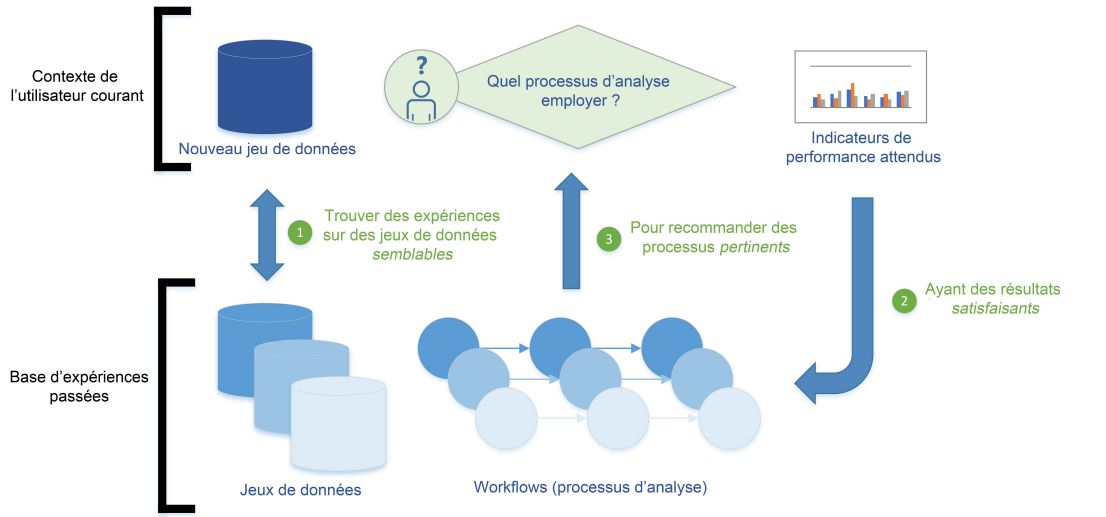

Figure 8.2: Principe de méta-analyse

Afin d'atteindre nos objectifs et de proposer de nouvelles approches performantes de méta-analyse pour l'assistance à l'analyse de données, différents verrous ont dû être levés. On rappellera ci-dessous ce cheminement et le bilan des travaux et contributions qui en ont résultés.

8.2 Contributions

Assistance à l'analyse de données - Motivés par la tendance vers des modèles toujours plus efficaces, des espaces de recherche plus grands et les limites des outils existants, le premier verrou à adresser était spécifiquement lié à la problématique de d'assistance des utilisateurs dans le processus d'analyse de données. Plus précisément, l'enjeu consistait à définir de nouvelles méthodes efficaces, interopérables et simples d'utilisation pour l'assistance des utilisateurs

dans l'analyse de données. Par conséquent, notre réponse a été de proposer et développer un système de cadre avec un tel objectif. À cette fin, nous avons proposé un framework qui tire parti des idées du méta-apprentissage reposant sur les dissimilarités entre jeux de données capable de fournir un support dans le but d'améliorer l'analyse et de réduire le temps passé dans la sélection et la paramétrisation des algorithmes. C'est un outil qui, pour la première fois, ne vise pas à fournir un support d'analyse de données uniquement pour la sélection et le paramétrage des algorithmes, mais plutôt, il est orienté vers une contribution positive à la confiance dans un système d'aide à la decision aussi puissant en fournissant automatiquement un ensemble de niveaux d'explications pour inspecter les résultats fournis sans avoir à dépendre d'un data scientist pour générer et interpréter tous les tracés et résultats extrêmes. Nous avons implémenté un prototype du cadre proposé, AMLBID, sur une architecture client-serveur, où le serveur coordonne en tant que système d'assistance AutoML, qui, étant donné un problème (ensemble de données), une métrique prédictive souhaitée (précision, rappel, score F1) recommande des algorithmes ML avec une configuration d'hyperparamètres associés qui sont classés en fonction de leurs impact sur le résultat final de l'analyse (cf. Chapitre 3), tandis que le côté client est composé d'une interface graphique conviviale qui facilite la manipulation des ensembles de données, prend en charge la simulation visuelle de divers scénarios, et faciliter l'interprétation des résultats obtenus (cf. Chapitre 5). Parallèlement, nous implémentons un module basé sur des règles d'inférences qui guide les utilisateurs finaux, en cas de résultats insatisfaisants renvoyés par l'AutoML, destiné à améliorer les performances prédictives. Par conséquent, cela peut augmenter la transparence, la contrôlabilité et la confiance en AutoML.

Une preuve de concept recommandant simplement l'emploi du workflow passé le plus pertinent a été développée, et a permis de valider l'intérêt de l'approche de méta-analyse envisagée. Une importante série d'expériences de méta-apprentissage a été réalisée pour démontrer la praticabilité de ce cadre d'évaluation. Ces contributions ont donné lieu à des publications en conférences et revues internationales [145], [36], [37]. Le cadre d'évaluation produit a de plus été réemployé systématiquement pour évaluer les différentes propositions et améliorations possibles au méta-niveau, lors de la levée des verrous suivants.

Dissimilarité entre jeux de données - Il existe en effet une forte dépendance entre l'efficacité du méta-apprentissage et la caractérisation des jeux de données étudiés. Afin de permettre de nouvelles approches de méta-analyse et pouvoir identifier les expériences pertinentes dans le contexte de l'utilisateur, nous nous sommes ensuite intéressés à un verrou majeur du domaine : *la caractérisation de jeux de données*. Suite au constat d'une importante perte d'information dans les méthodes de caractérisation communément employées, nous avons proposé

l'emploi de nouvelles techniques de dissimilarité entre jeux de données. Nous avons défini un ensemble de propriétés désirables pour proposer des fonctions capables de prendre en compte l'entièreté de l'information disponible, ce qui passe par la caractérisation des attributs particuliers de ces jeux de données (cf. Chapitre 4). Nous avons ensuite montré que ces dissimilarités permettent de caractériser l'adéquation d'algorithmes de classification avec des jeux de données plus efficacement que des distances traditionnelles, et qu'elles peuvent être employées avec de bonnes performances dans un contexte de classification au méta-niveau pour la sélection d'algorithmes. Au-delà du simple intérêt de l'approche, nos évaluations par analyses dimensionnelles ont permis d'étudier en détail l'impact des différents facteurs et composants de ces fonctions de dissimilarité sur la performance au méta-niveau. On peut ainsi proposer un candidat qualifié comme brique de base pour de nouvelles approches de méta-analyse. Ces fonctions de dissimilarité ont enfin permis le développement d'un prototype d'assistant à l'analyse de données basé sur de nouvelles approches de méta-analyse. Cette contribution a donné lieu à une future publication en une revue internationale de Big Data [233].

Reproductibilité des résultats - À cette fin, nous avons matérialisé nos contributions dans le progiciel AutoML transparent, interprétable et auto-explicable AMLBID, qui, étant donné un algorithme de classification, recommande des algorithmes d'apprentissage automatique qui impactent positivement l'analyse (cf. Chapitre 6). AMLBID est implémenté en tant que package Python open source pour reproduire les expériences, des analyses et permettre une analyse plus approfondie. Nous avons longuement évalué nos recommandations sous trois angles. Dans le premier, nous avons vérifié la précision de nos prédictions. Dans le second, nous avons analysé le gain qu'ils apportaient à l'utilisateur final non expérimenté. Enfin, dans le troisième, nous avons analysé les performances d'AMLBID par rapport aux humains dans un scénario réaliste d'utilité et d'utilisabilité.

Bien que AMLBID en est encore à ses débuts, le package a été téléchargé plus de **17893**[1] fois sur PyPI[2] (à l'exclusion de ses liens externes) au cours de sa première année. Les retours de la communauté sont très positifs et plusieurs nouvelles applications ont été proposées en plus des multiples demandes industrielles. A divers stades de maturité, cette contribution a donné lieu à une publication en une revue internationale qui référence les nouvelles découvertes scientifiques en terme de progiciel dans les différents domaines de la recherche scientifique [144].

Le domaine étudié étant large et en grande partie inexploré, on pourra noter

[1] https://pypistats.org/packages/amlbid
[2] https://pypi.org/project/AMLBID/

que chaque verrou levé apporte de nouvelles perspectives, chaque contribution soulève davantage de questions que de réponses. Devant les résultats encourageants des expériences empiriques effectuées, leurs développement est appelé à se poursuivre. Les contributions et propositions détaillées au cours des différents chapitres ont donné lieu à de nombreuses perspectives d'amélioration pour l'assistance à l'analyse de données. Nous détaillerons ainsi dans les paragraphes suivants les principales opportunités de recherche se présentant à court et long termes.

8.3 Perspectives

- L'une des limites du développement de systèmes AutoML basés sur le méta-analyse est la complexité computationelle associée à l'évaluation de tous les algorithmes candidats dans l'espace algorithmique $\mathcal{A}$ et sur tous les ensembles de données dans l'espace problème $\mathcal{P}$. Cependant, l'augmentation du nombre d'algorithmes ou d'ensembles de données augmente le coût de calcul pour l'enrichissement de la base de méta-connaissances. Cela entrave l'exploration rigoureuse de l'espace d'algorithmes et de problèmes. Bien que nous avons utilisé un nombre raisonnable d'algorithmes et d'ensembles de données, le coût de calcul initial est toujours élevé pour améliorer le système. Ce challenge peut être couvert dans des travaux futurs en incorporant des données dans la méta-base de connaissances à partir des plates-formes Kaggle et OpenML. Ces plates-formes collaboratives de science des données disposent de référentiels de données en ligne dans lesquels les analystes partagent leurs résultats expérimentaux issus de l'application d'algorithmes sur des ensembles de données. Cela réduirait non seulement le coût de calcul, mais offrirait également la possibilité d'explorer rigoureusement l'espace d'algorithmes et de problèmes et garantirait qu'aucun biais n'est induit dans le système au méta-niveau.

- Une autre possibilité de recherche est d'élargir l'idée du système AutoML couvert dans cette thèse. Dans ce cas, le système recommanderait non seulement des algorithmes d'apprentissage automatique, mais également des méthodes de prétraitement et d'analyse post-traitement capables d'expliquer les résultats expérimentaux. Ainsi, étant donné un nouvel ensemble de données, le système AutoML recommanderait les composants les plus appropriés pour résoudre le problème d'entrée. En fait, nous avons déjà commencé à travailler dans ce sens.

- Les expériences décrites à la section 4.4.2 ouvrent également de nouvelles

possibilités d'utilisation des réseaux de neurones au applications basés sur le méta-apprentissage. Compte tenu des résultats qui y sont rapportés, un réseau de neurones artificiels pourrait être développé pour recommander le pipeline ML à utiliser. Ainsi, un réglage plus ciblé pourrait être effectué en utilisant un plus petit nombre d'évaluations et uniquement les HPs les plus importants.

- Enfin, nos études s'est limitées aux problèmes de classification. Autrement dit, nous fournissons une assistance aux utilisateurs dans la sélection et la paramétrisation des algorithmes ML uniquement si le problème en question est de type classification. Cependant, notre méthode peut être directement étendue à la régression, au clustering et aux bibliothèques d'apprentissage automatique distribuées (par exemple, SparkML [230]).

Bibliography

[1] Jens Kober, J. Andrew Bagnell, and Jan Peters. "Reinforcement Learning in Robotics: A Survey". In: *The International Journal of Robotics Research* 32.11 (Sept. 1, 2013), pp. 1238–1274. DOI: 10.1177/0278364913495721.

[2] Hayit Greenspan, Bram van Ginneken, and Ronald M. Summers. "Guest Editorial Deep Learning in Medical Imaging: Overview and Future Promise of an Exciting New Technique". In: *IEEE Transactions on Medical Imaging* 35.5 (May 2016), pp. 1153–1159. DOI: 10.1109/TMI.2016.2553401.

[3] Chenyi Chen et al. "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015, pp. 2722–2730. DOI: 10.1109/ICCV.2015.312.

[4] Besim Bilalli. "Learning the Impact of Data Pre-processing in Data Analysis". Barcelona: UNIVERSITAT POLITÈCNICA DE CATALUNYA, 2018.

[5] *Data Engineering, Preparation, and Labeling for AI 2019*. Cognilytica. URL: https://www.cognilytica.com/document/report-data-engineering-preparation-and-labeling-for-ai-2019/.

[6] Alexandre QUEMY. "End-to-End Approach to Classification in Unstructured Spaces". Poland: POZNAN UNIVERSITY OF TECHNOLOGY, 2020.

[7] Liam Li et al. "Towards Efficient Automated Machine Learning". 2020.

[8] Esteban Real et al. "Regularized Evolution for Image Classifier Architecture Search". Feb. 16, 2019. arXiv: 1802.01548 [cs]. URL: http://arxiv.org/abs/1802.01548.

[9] Iddo Drori et al. *Automatic Machine Learning by Pipeline Synthesis Using Model-Based Reinforcement Learning and a Grammar*. May 24, 2019.

[10] Lars Kotthoff et al. "Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA". In: *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Cham: Springer International Publishing, 2019, pp. 81–95. DOI: 10.1007/978-3-030-05318-5_4.

[11] Randal S. Olson and Jason H. Moore. "TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning". In: *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Cham: Springer International Publishing, 2019, pp. 151–160. DOI: 10.1007/978-3-030-05318-5_8.

[12] Matthias Feurer et al. "Efficient and Robust Automated Machine Learning". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Cambridge, MA, USA: MIT Press, Dec. 2015, pp. 2755–2763.

[13] Joaquin Vanschoren. "Meta-Learning". In: *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. 2019, pp. 35–61. DOI: 10.1007/978-3-030-05318-5_2.

[14] *Automated Machine Learning: Methods, Systems, Challenges*. en. The Springer Series on Challenges in Machine Learning. 2019. DOI: 10.1007/978-3-030-05318-5.

[15] Mitar Milutinovic. "Towards Automatic Machine Learning Pipeline Design". UC Berkeley, 2019.

[16] Alexandros Kalousis. "Algorithm Selection via Meta-Learning". University of Geneva, 2002. DOI: 10.13097/archive-ouverte/unige:104435.

[17] Noy Cohen-Shapira et al. "AutoGRD: Model Recommendation Through Graphical Dataset Representation". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. CIKM '19. New York, NY, USA, Nov. 2019, pp. 821–830. DOI: 10.1145/3357384.3357896.

[18] Matthias Feurer et al. "Efficient and Robust Automated Machine Learning". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Dec. 2015, pp. 2755–2763.

[19] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *The Journal of Machine Learning Research* 13 (Feb. 1, 2012), pp. 281–305.

[20] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. "Initializing Bayesian Hyperparameter Optimization via Meta-Learning". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, Jan. 2015, pp. 1128–1135.

[21] Randal S. Olson et al. "Evaluation of a Tree-Based Pipeline Optimization Tool for Automating Data Science". In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. GECCO '16. 2016, pp. 485–492. DOI: 10.1145/2908812.2908918.

[22] Sigrún Andradóttir. "A Review of Random Search Methods". In: *Handbook of Simulation Optimization*. International Series in Operations Research & Management Science. New York, NY: Springer, 2015, pp. 277–292. DOI: 10.1007/978-1-4939-1384-8_10.

[23] Rafael Gomes Mantovani et al. "Rethinking Default Values: A Low Cost and Efficient Strategy to Define Hyperparameters". July 8, 2021. arXiv: 2008.00025 [cs, stat].

[24] Taciana A. F. Gomes et al. "Combining Meta-learning and Search Techniques to SVM Parameter Selection". In: *2010 Eleventh Brazilian Symposium on Neural Networks*. 2010 Eleventh Brazilian Symposium on Neural Networks. Oct. 2010, pp. 79–84. DOI: 10.1109/SBRN.2010.22.

[25] Luis Carlos Padierna et al. "Hyper-Parameter Tuning for Support Vector Machines by Estimation of Distribution Algorithms". In: *Nature-Inspired Design of Hybrid Intelligent Systems*. Studies in Computational Intelligence. Cham: Springer International Publishing, 2017, pp. 787–800. DOI: 10.1007/978-3-319-47054-2_53.

[26] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". Aug. 29, 2012. arXiv: 1206.2944.

[27] Rémi Bardenet et al. "Collaborative Hyperparameter Tuning". In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. Proceedings of Machine Learning Research 2. Atlanta, Georgia, USA: PMLR, June 17–19, 2013, pp. 199–207. URL: https://proceedings.mlr.press/v28/bardenet13.html.

[28] Kirthevasan Kandasamy et al. "Multi-Fidelity Gaussian Process Bandit Optimisation". Mar. 15, 2019. arXiv: 1603.06288.

[29] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. "Gradient-Based Hyperparameter Optimization through Reversible Learning". Apr. 2, 2015. arXiv: 1502.03492.

[30] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "Sequential Model-Based Optimization for General Algorithm Configuration". en. In: *Learning and Intelligent Optimization*. Lecture Notes in Computer Science. Berlin, Heidelberg, 2011, pp. 507–523. DOI: 10.1007/978-3-642-25566-3.

[31] James Bergstra et al. "Algorithms for Hyper-Parameter Optimization". In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*. NIPS'11. Red Hook, NY, USA: Curran Associates Inc., Dec. 12, 2011, pp. 2546–2554.

[32] Jichang Dong et al. "A framework of pavement management system based on IoT and big data". In: *Advanced Engineering Informatics* 47 (2021), p. 101226. DOI: 10.1016/j.aei.2020.101226.

[33] Jacqueline Schmitt et al. "Predictive model-based quality inspection using Machine Learning and Edge Cloud Computing". In: *Advanced Engineering Informatics* 45 (Aug. 2020), p. 101101. DOI: 10.1016/j.aei.2020.101101. URL: https://doi.org/10.1016/j.aei.2020.101101.

[34] David Lechevalier et al. "A Methodology for the Semi-Automatic Generation of Analytical Models in Manufacturing". In: *Computers in Industry* 95 (2018), pp. 54–67. DOI: 10.1016/j.compind.2017.12.005.

[35] Zhaoguang Xu, Yanzhong Dang, and Peter Munro. "Knowledge-driven intelligent quality problem-solving system in the automotive industry". In: *Advanced Engineering Informatics* 38 (2018), pp. 441–457. DOI: 10.1016/j.aei.2018.08.013.

[36] Moncef Garouani et al. "Towards the Automation of Industrial Data Science: A Meta-Learning Based Approach". In: *23rd International Conference on Enterprise Information Systems*. May 2021, pp. 709–716. DOI: 10.5220/0010457107090716.

[37] Moncef Garouani et al. "Towards Big Industrial Data Mining through Explainable Automated Machine Learning". In: *The International Journal of Advanced Manufacturing Technology* (Feb. 10, 2022). DOI: 10.1007/s00170-022-08761-9.

[38] Junlong Chen, Zijun Zhang, and Feng Wu. "A data-driven method for enhancing the image-based automatic inspection of IC wire bonding defects". In: *International Journal of Production Research* 59.16 (Sept. 2020), pp. 4779–4793. DOI: 10.1080/00207543.2020.1821928. URL: https://doi.org/10.1080/00207543.2020.1821928.

[39] Dimitrios Kateris et al. “A machine learning approach for the condition monitoring of rotating machinery”. In: *Journal of Mechanical Science and Technology* 28.1 (Jan. 2014), pp. 61–71. DOI: 10.1007/s12206-013-1102-y. URL: https://doi.org/10.1007/s12206-013-1102-y.

[40] Xian-Bo Wang et al. “Automatic representation and detection of fault bearings in in-wheel motors under variable load conditions”. In: *Advanced Engineering Informatics* 49 (2021), p. 101321. DOI: 10.1016/j.aei.2021.101321.

[41] Yiyuan Gao and Dejie Yu. “Intelligent fault diagnosis for rolling bearings based on graph shift regularization with directed graphs”. In: *Advanced Engineering Informatics* 47 (2021), p. 101253. DOI: 10.1016/j.aei.2021.101253.

[42] Cong Zhou, J. Geoffrey Chase, and Geoffrey W. Rodgers. “Degradation evaluation of lateral story stiffness using HLA-based deep learning networks”. In: *Advanced Engineering Informatics* 39 (2019), pp. 259–268. DOI: 10.1016/j.aei.2019.01.007.

[43] Rubén Medina et al. “Gear and Bearing Fault Classification under Different Load and Speed by Using Poincaré Plot Features and SVM”. In: *Journal of Intelligent Manufacturing* (2020) (). DOI: 10.1007/s10845-020-01712-9.

[44] Juan Pablo Usuga Cadavid et al. “Machine Learning Applied in Production Planning and Control: A State-of-the-Art in the Era of Industry 4.0”. In: *Journal of Intelligent Manufacturing* 31.6 (2020), pp. 1531–1558. DOI: 10.1007/s10845-019-01531-7.

[45] Real Carbonneau, Kevin Laframboise, and Rustam Vahidov. “Application of Machine Learning Techniques for Supply Chain Demand Forecasting”. In: *European Journal of Operational Research* 184.3 (2008), pp. 1140–1154. DOI: 10.1016/j.ejor.2006.12.004.

[46] Qi Wu. “Product Demand Forecasts Using Wavelet Kernel Support Vector Machine and Particle Swarm Optimization in Manufacture System”. In: *Journal of Computational and Applied Mathematics* 233.10 (2010), pp. 2481–2491. DOI: 10.1016/j.cam.2009.10.030.

[47] Miguel Cuartas et al. “Machine Learning Algorithms for the Prediction of Non-Metallic Inclusions in Steel Wires for Tire Reinforcement”. en. In: *Journal of Intelligent Manufacturing* (July 2020). DOI: 10.1007/s10845-020-01623-9.

[48] Rubén Medina et al. "Gear and Bearing Fault Classification under Different Load and Speed by Using Poincaré Plot Features and SVM". en. In: *Journal of Intelligent Manufacturing* (Nov. 2020). DOI: 10.1007/s10845-020-01712-9.

[49] Anahid Jalali et al. "Predicting Time-to-Failure of Plasma Etching Equipment Using Machine Learning". In: *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*. June 2019, pp. 1–8. DOI: 10.1109/ICPHM.2019.8819404.

[50] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". Apr. 23, 2014. arXiv: 1206.5538.

[51] Gang Luo. "PredicT-ML: A Tool for Automating Machine Learning Model Building with Big Clinical Data". In: *Health Information Science and Systems* 4.1 (June 8, 2016), p. 5. DOI: 10.1186/s13755-016-0018-1.

[52] Chris Thornton et al. "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms". In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '13. 2013, pp. 847–855. DOI: 10.1145/2487575.2487629.

[53] Thorsten Wuest et al. "Machine Learning in Manufacturing: Advantages, Challenges, and Applications". In: *Production & Manufacturing Research* 4.1 (2016), pp. 23–45. DOI: 10.1080/21693277.2016.1192517.

[54] Sirian Caldarelli et al. "A Signal-Based Approach to News Recommendation". In: *UMAP*. 2016.

[55] Claudio Biancalana et al. "Context-Aware Movie Recommendation Based on Signal Processing and Machine Learning". In: *Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation*. CAMRa '11. New York, NY, USA: Association for Computing Machinery, Oct. 23, 2011, pp. 5–10. DOI: 10.1145/2096112.2096114.

[56] Melissa Onori, A. Micarelli, and G. Sansonetti. "A Comparative Analysis of Personality-Based Music Recommender Systems". In: *EMPIRE@RecSys*. 2016.

[57] Giuseppe Sansonetti et al. "Enhancing Cultural Recommendations through Social and Linked Open Data". In: *User Modeling and User-Adapted Interaction* 29.1 (Mar. 1, 2019), pp. 121–159. DOI: 10.1007/s11257-019-09225-8.

[58] Giuseppe Sansonetti. "Point of Interest Recommendation Based on Social and Linked Open Data". In: *Personal and Ubiquitous Computing* 23.2 (Apr. 1, 2019), pp. 199–214. DOI: 10.1007/s00779-019-01218-z.

[59] Alessandro Fogli and Giuseppe Sansonetti. "Exploiting Semantics for Context-Aware Itinerary Recommendation". In: *Personal and Ubiquitous Computing* 23.2 (Apr. 1, 2019), pp. 215–231. DOI: 10.1007/s00779-018-01189-7.

[60] Ruturaj Kulkarni, Shruti Dhavalikar, and Sonal Bangar. "Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning". In: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). Aug. 2018, pp. 1–4. DOI: 10.1109/ICCUBEA.2018.8697819.

[61] Jose-Raul Ruiz-Sarmiento et al. "A Predictive Model for the Maintenance of Industrial Machinery in the Context of Industry 4.0". In: *Engineering Applications of Artificial Intelligence* 87 (2020) (). DOI: 10.1016/j.engappai.2019.103289.

[62] Irfan Khan et al. "A Literature Survey and Empirical Study of Meta-Learning for Classifier Selection". In: *IEEE Access* 8 (2020), pp. 10262–10281. DOI: 10.1109/ACCESS.2020.2964726.

[63] D.H. Wolpert and W.G. Macready. "No Free Lunch Theorems for Optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 67–82. DOI: 10.1109/4235.585893.

[64] Bobak Shahriari et al. "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1 (Jan. 2016), pp. 148–175. DOI: 10.1109/JPROC.2015.2494218.

[65] Jonathan Waring, Charlotta Lindvall, and Renato Umeton. "Automated Machine Learning: Review of the State-of-the-Art and Opportunities for Healthcare". en. In: *Artificial Intelligence in Medicine* 104 (Apr. 2020), p. 101822. DOI: 10.1016/j.artmed.2020.101822.

[66] Iddo Drori et al. *AlphaD3M Machine Learning Pipeline Synthesis*. 2018.

[67] Besim Bilalli et al. "PRESISTANT: Data Pre-Processing Assistant". In: *Information Systems in the Big Data Era*. Cham, 2018, pp. 57–65. DOI: 10.1007/978-3-319-92901-9_6.

[68] Roman Vainshtein et al. "A Hybrid Approach for Automatic Model Recommendation". In: CIKM '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1623–1626. DOI: 10.1145/3269206.3269299.

[69] Matthias Reif et al. "Automatic Classifier Selection for Non-Experts". In: *Pattern Analysis and Applications* 17.1 (2014), pp. 83–96. DOI: 10.1007/s10044-012-0280-z.

[70] Lars Kotthoff et al. "Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA". In: *Automated Machine Learning: Methods, Systems, Challenges*. Cham, 2019, pp. 81–95. DOI: 10.1007/978-3-030-05318-5_4.

[71] Randal S. Olson and Jason H. Moore. "TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning". en. In: *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Cham, 2019, pp. 151–160. DOI: 10.1007/978-3-030-05318-5.

[72] Lars Kotthoff et al. "Auto-WEKA 2.0: Automatic Model Selection and Hyperparameter Optimization in WEKA". In: *Journal of Machine Learning Research* 18.25 (2017), pp. 1–5.

[73] Matthias Feurer et al. "Auto-Sklearn: Efficient and Robust Automated Machine Learning". In: *Automated Machine Learning: Methods, Systems, Challenges*. Cham, 2019, pp. 113–134. DOI: 10.1007/978-3-030-05318-5_6.

[74] *RapidMiner | Data Science & Machine Learning Platform*. URL: https://rapidminer.com/.

[75] *H2O.Ai | AI Cloud Platform*. URL: https://www.h2o.ai/.

[76] *BigML*. Machine Learning made easy. URL: https://bigml.com/.

[77] *DataRobot*. DataRobot | AI Cloud. URL: https://www.datarobot.com/.

[78] Isabelle Guyon et al. "Analysis of the AutoML Challenge series 2015-2018". In: *AutoML*. Springer series on Challenges in Machine Learning. 2019.

[79] Brenden M. Lake et al. "Building Machines That Learn and Think Like People". Nov. 2, 2016. arXiv: 1604.00289.

[80] R. Vilalta et al. "Using Meta-Learning to Support Data Mining". In: *Int. J. Comput. Sci. Appl.* (2004).

[81] Hilan Bensusan, Christophe Giraud-Carrier, and Claire Kennedy. "A Higher-Order Approach to Meta-Learning". In: *Proceedings of the ECML'2000 workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination* (2000), pp. 109–117. URL: https://research-information.bris.ac.uk/en/publications/a-higher-order-approach-to-meta-learning.

[82] Bernhard Pfahringer. “Meta-Learning by Landmarking Various Learning Algorithms”. In: (May 23, 2001).

[83] Mikhail M. Meskhi et al. “Learning Abstract Task Representations”. Jan. 28, 2021. arXiv: 2101.07852.

[84] Yonghong Peng et al. “Improved Dataset Characterisation for Meta-learning”. In: *Discovery Science*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2002, pp. 141–152. DOI: 10.1007/3-540-36182-0_14.

[85] Bruno Almeida Pimentel and André C. P. L. F. de Carvalho. “A New Data Characterization for Selecting Clustering Algorithms Using Meta-Learning”. In: *Information Sciences* 477 (Mar. 1, 2019), pp. 203–219. DOI: 10.1016/j.ins.2018.10.043.

[86] Adriano Rivolli et al. “Characterizing Classification Datasets: A Study of Meta-Features for Meta-Learning”. Aug. 26, 2019. arXiv: 1808.10406.

[87] R. G. Mantovani. “Use of Meta-Learning for Hyperparameter Tuning of Classification Problems”. In: 2018. DOI: 10.11606/T.55.2018.TDE-15102018-092202.

[88] Matthias Reif et al. “Automatic Classifier Selection for Non-Experts”. In: *Pattern Analysis and Applications* 17.1 (Feb. 2014), pp. 83–96. DOI: 10.1007/s10044-012-0280-z.

[89] Ciro Castiello, Giovanna Castellano, and Anna Maria Fanelli. “Meta-Data: Characterization of Input Features for Meta-learning”. In: *Modeling Decisions for Artificial Intelligence*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 457–468. DOI: 10.1007/11526018_45.

[90] M. Reif. “A Comprehensive Dataset for Evaluating Approaches of Various Meta-learning Tasks”. In: *ICPRAM*. 2012. DOI: 10.5220/0003736302730276.

[91] Matthias Reif et al. “Automatic Classifier Selection for Non-Experts”. In: *Pattern Analysis and Applications* 17.1 (Feb. 1, 2014), pp. 83–96. DOI: 10.1007/s10044-012-0280-z.

[92] Petr Kuba et al. “Exploiting Sampling and Meta-learning for Parameter Setting forSupport Vector Machines”. In: (2002). URL: https://www.semanticscholar.org/paper/Exploiting-Sampling-and-Meta-learning-for-Parameter-Kuba-Brazdil/a3c5fbdf3a1a3d8f12412186cc9eb4a8c126908c.

[93] A. Kalousis and M. Hilario. "Model Selection via Meta-Learning: A Comparative Study". In: *Proceedings 12th IEEE Internationals Conference on Tools with Artificial Intelligence. ICTAI 2000*. Nov. 2000, pp. 406–413. DOI: 10.1109/TAI.2000.889901.

[94] Pieter Gijsbers. *Automatic Construction of Machine Learning Pipelines*. Eindhoven: Eindhoven University of Technology, Oct. 2017. URL: https://research.tue.nl/en/studentTheses/automatic-construction-of-machine-learning-pipelines.

[95] Johannes Fürnkranz and Johann Petrak. "An Evaluation of Landmarking Variants". In: *Undefined*. 2001. URL: https://www.semanticscholar.org/paper/An-Evaluation-of-Landmarking-Variants-F%C3%BCrnkranz-Petrak/13d8643dbad0f78d650c7fda8f60b3ba095a4402.

[96] B. Pfahringer and Hilan Bensusan. "Tell Me Who Can Learn You and I Can Tell You Who You Are: Landmarking Various Learning Algorithms". In: *Undefined*. 2000. URL: https://www.semanticscholar.org/paper/Tell-me-who-can-learn-you-and-I-can-tell-you-who-Pfahringer-Bensusan/78e71a6a649dd6778bb1c0923f626d6573cc2b06.

[97] Daren Ler. *Utilising Regression-based Landmarkers within a Meta-learning Framework for Algorithm Selection*. 2005. URL: https://www.semanticscholar.org/paper/Utilising-Regression-based-Landmarkers-within-a-for-Ler/5202b6dbf82aba17e8d95221ec9525f46bf31dec.

[98] Luís P. F. Garcia, André C. P. L. F. de Carvalho, and Ana C. Lorena. "Noise Detection in the Meta-Learning Level". In: *Neurocomputing*. Recent Advancements in Hybrid Artificial Intelligence Systems and Its Application to Real-World Problems 176 (Feb. 2, 2016), pp. 14–25. DOI: 10.1016/j.neucom.2014.12.100.

[99] Ana C. Lorena et al. "Data Complexity Meta-Features for Regression Problems". In: *Machine Learning* 107.1 (Jan. 1, 2018), pp. 209–246. DOI: 10.1007/s10994-017-5681-1.

[100] Tin Kam Ho and M. Basu. "Complexity Measures of Supervised Classification Problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.3 (Mar. 2002), pp. 289–300. DOI: 10.1109/34.990132.

[101] R. Vilalta and Youssef Drissi. "A Characterization of Difficult Problems in Classification". In: *ICMLA* (2002). URL: https://www.semanticscholar.org/paper/A-Characterization-of-Difficult-Problems-in-Vilalta-Drissi/6fed0d3a06b9d4f38812824c6f77757b1dff44ab.

[102] Yang Zhongguo et al. “Choosing Classification Algorithms and Its Optimum Parameters Based on Data Set Characteristics”. In: *Journal of Computers (Taiwan)* 28 (Oct. 1, 2017), pp. 26–38. DOI: 10.3966/199115992017102805003.

[103] Rattan Priya et al. “Using Genetic Algorithms to Improve Prediction of Execution Times of ML Tasks”. In: *Hybrid Artificial Intelligent Systems*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 196–207. DOI: 10.1007/978-3-642-28942-2_18.

[104] Fábio Pinto et al. “autoBagging: Learning to Rank Bagging Workflows with Metalearning”. June 28, 2017. arXiv: 1706.09367.

[105] María De Mar Molina et al. “Meta-Learning Approach for Automatic Parameter Tuning: A Case of Study with Educational Datasets”. In: *EDM*. 2012.

[106] Parker Ridd and Christophe G. Giraud-Carrier. “Using Metalearning to Predict When Parameter Optimization Is Likely to Improve Classification Accuracy”. In: *Proceedings of the International Workshop on Meta-learning and Algorithm Selection*. Vol. 1201. CEUR Workshop Proceedings. 2014, pp. 18–23.

[107] Alex Lazinica. *Particle Swarm Optimization*. 2009. 490 pp.

[108] Matthias Reif, Faisal Shafait, and Andreas Dengel. “Meta-Learning for Evolutionary Parameter Optimization of Classifiers”. In: *Machine Learning* 87.3 (June 1, 2012), pp. 357–380. DOI: 10.1007/s10994-012-5286-7.

[109] Chengrun Yang et al. “OBOE: Collaborative Filtering for AutoML Model Selection”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (July 25, 2019), pp. 1173–1183. DOI: 10.1145/3292500.3330909.

[110] Max Bramer. “Estimating the Predictive Accuracy of a Classifier”. In: *Principles of Data Mining*. Undergraduate Topics in Computer Science. London: Springer, 2013, pp. 79–92. DOI: 10.1007/978-1-4471-4884-5_7.

[111] Christian Kopf Charles, Charles Taylor, and Jorg Keller. “Meta-Analysis: From Data Characterisation for Meta-Learning to Meta-Regression”. In: *Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support,Meta-Learning and ILP*. 2000.

[112] Silvio B. Guerra, Ricardo B. C. Prudêncio, and Teresa B. Ludermir. "Predicting the Performance of Learning Algorithms Using Support Vector Machines as Meta-regressors". In: *Artificial Neural Networks - ICANN 2008*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 523–532. DOI: 10.1007/978-3-540-87536-9_54.

[113] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. "Sequential Model-Free Hyperparameter Tuning". In: *2015 IEEE International Conference on Data Mining*. 2015 IEEE International Conference on Data Mining. Nov. 2015, pp. 1033–1038. DOI: 10.1109/ICDM.2015.20.

[114] Péricles B. C. de Miranda et al. "An Experimental Study of the Combination of Meta-Learning with Particle Swarm Algorithms for SVM Parameter Selection". In: *Computational Science and Its Applications – ICCSA 2012*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 562–575. DOI: 10.1007/978-3-642-31137-6_43.

[115] Matthias Reif, Faisal Shafait, and Andreas Dengel. "Prediction of Classifier Training Time Including Parameter Optimization". In: *KI 2011: Advances in Artificial Intelligence*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 260–271. DOI: 10.1007/978-3-642-24455-1_25.

[116] Guangtao Wang et al. "A Feature Subset Selection Algorithm Automatic Recommendation Method". In: *Journal of Artificial Intelligence Research* 47 (May 15, 2013), pp. 1–34. DOI: 10.1613/jair.3831. arXiv: 1402.0570.

[117] Enrique Leyva et al. "On the Use of Meta-Learning for Instance Selection: An Architecture and an Experimental Study". In: *Information Sciences: an International Journal* 266 (May 1, 2014), pp. 16–30. DOI: 10.1016/j.ins.2014.01.007.

[118] Guangtao Wang, Qinbao Song, and Xiaoyan Zhu. "An Improved Data Characterization Method and Its Application in Classification Algorithm Recommendation". In: *Applied Intelligence* 43.4 (Dec. 1, 2015), pp. 892–912. DOI: 10.1007/s10489-015-0689-3.

[119] Xiaoyan Zhu et al. "A New Classification Algorithm Recommendation Method Based on Link Prediction". In: *Knowledge-Based Systems* 159 (Nov. 1, 2018), pp. 171–185. DOI: 10.1016/j.knosys.2018.07.015.

[120] Mario A. Muñoz et al. "Instance Spaces for Machine Learning Classification". In: *Machine Learning* 107.1 (Jan. 1, 2018), pp. 109–147. DOI: 10.1007/s10994-017-5629-5.

[121] A. Bernstein, Shawndra Hill, and F. Provost. "Intelligent Assistance for the Data Mining Process: An Ontology-Based Approach". In: (2002).

[122] Mustafa V Nural. "Ontology-Based Semantics vs Meta-Learning for Predictive Big Data Analytics". University of Georgia, Athens, GA, USA, 2017.

[123] Mustafa V. Nural, Hao Peng, and John A. Miller. "Using Meta-Learning for Model Type Selection in Predictive Big Data Analytics". In: *2017 IEEE International Conference on Big Data (Big Data)*. 2017, pp. 2027–2036. DOI: 10.1109/BigData.2017.8258149.

[124] Mustafa V. Nural, Michael E. Cotterell, and John A. Miller. "Using Semantics in Predictive Big Data Analytics". In: *2015 IEEE International Congress on Big Data*. 2015 IEEE International Congress on Big Data. June 2015, pp. 254–261. DOI: 10.1109/BigDataCongress.2015.43.

[125] Witten Ian H., Frank Eibe, and Hall Mark A. *Data Mining: Practical Machine Learning Tools and Techniques*. Fourth. Morgan Kaufmann, 2017. DOI: 10.1016/C2015-0-02071-8.

[126] *AutoML Tables*. Google Cloud. URL: https://cloud.google.com/automl-tables/docs.

[127] Haifeng Jin, Qingquan Song, and Xia Hu. "Auto-Keras: An Efficient Neural Architecture Search System". Mar. 26, 2019. arXiv: 1806.10282 [cs, stat]. URL: http://arxiv.org/abs/1806.10282.

[128] Thomas Swearingen et al. "ATM: A distributed, collaborative, scalable system for automated machine learning". In: *2017 IEEE International Conference on Big Data (Big Data)*. 2017, pp. 151–162. DOI: 10.1109/BigData.2017.8257923.

[129] Juan S. Angarita-Zapata, Gina Maestre-Gongora, and Jenny Fajardo Calderín. "A Case Study of AutoML for Supervised Crash Severity Prediction". In: 19th World Congress of the International Fuzzy Systems Association (IFSA). Atlantis Press, Aug. 30, 2021, pp. 187–194. DOI: 10.2991/asum.k.210827.026.

[130] Akram Mustafa and Mostafa Rahimi Azghadi. "Automated Machine Learning for Healthcare and Clinical Notes Analysis". In: *Computers* 10.2 (2 Feb. 2021), p. 24. DOI: 10.3390/computers10020024.

[131] Andrew A. Borkowski et al. "Google Auto ML versus Apple Create ML for Histopathologic Cancer Diagnosis; Which Algorithms Are Better?" Mar. 19, 2019. arXiv: 1903.08057.

[132] K. T. Y. Mahima, T.N.D.S.Ginige, and Kasun De Zoysa. "Evaluation of Sentiment Analysis Based on AutoML and Traditional Approaches". In: *International Journal of Advanced Computer Science and Applications (IJACSA)* 12.2 (2 2021/58/01). DOI: 10.14569/IJACSA.2021.0120277.

[133] Maria Tsiakmaki et al. "Implementing AutoML in Educational Data Mining for Prediction Tasks". In: *Applied Sciences* 10.1 (1 Jan. 2020), p. 90. DOI: 10.3390/app10010090.

[134] Loukides Mike. *2021 Data/AI Salary Survey*. Oct. 2021. URL: https://www.oreilly.com/radar/2021-data-ai-salary-survey/.

[135] Pieter Gijsbers et al. *An Open Source AutoML Benchmark*. 2019. arXiv: 1907.00909 [cs.LG].

[136] Ram K. Mazumder, Abdullahi M. Salman, and Yue Li. "Failure Risk Analysis of Pipelines Using Data-Driven Machine Learning Algorithms". In: *Structural Safety* 89 (2021), p. 102047. DOI: 10.1016/j.strusafe.2020.102047.

[137] S. Saravanamurugan et al. "Chatter Prediction in Boring Process Using Machine Learning Technique". en. In: *Int. J. Manuf. Res.* (2017). DOI: 10.1504/IJMR.2017.10007082.

[138] T. Benkedjouh et al. "Health Assessment and Life Prediction of Cutting Tools Based on Support Vector Regression". en. In: *Journal of Intelligent Manufacturing* 26.2 (Apr. 2015), pp. 213–223. DOI: 10.1007/s10845-013-0774-6.

[139] Simon D. Duque Anton, Sapna Sinha, and Hans Dieter Schotten. "Anomaly-based Intrusion Detection in Industrial Data with SVM and Random Forests". In: *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 2019, pp. 1–6. DOI: 10.23919/SOFTCOM.2019.8903672.

[140] Huaxia Deng et al. "A high-speed D-CART online fault diagnosis algorithm for rotor systems". In: *Applied Intelligence* 50.1 (June 2019), pp. 29–41. DOI: 10.1007/s10489-019-01516-2. URL: https://doi.org/10.1007/s10489-019-01516-2.

[141] Jae Kwon Kim, Young Shin Han, and Jong Sik Lee. "Particle swarm optimization–deep belief network–based rare class prediction model for highly class imbalance problem". In: *Concurrency and Computation: Practice and Experience* 29.11 (Apr. 2017). DOI: 10.1002/cpe.4128. URL: https://doi.org/10.1002/cpe.4128.

[142] Kazunori Imoto et al. "A CNN-Based Transfer Learning Method for Defect Classification in Semiconductor Manufacturing". In: *IEEE Transactions on Semiconductor Manufacturing* 32.4 (Nov. 2019), pp. 455–459. DOI: 10.1109/tsm.2019.2941752. URL: https://doi.org/10.1109/tsm.2019.2941752.

[143] Jae Kwon Kim et al. “Feature Selection Techniques for Improving Rare Class Classification in Semiconductor Manufacturing Process”. In: *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer International Publishing, 2017, pp. 40–47. DOI: 10.1007/978-3-319-58967-1_5. URL: https://doi.org/10.1007/978-3-319-58967-1%5C_5.

[144] Moncef Garouani et al. “AMLBID: An Auto-Explained Automated Machine Learning Tool for Big Industrial Data”. In: *SoftwareX* 17 (Jan. 1, 2022), p. 100919. DOI: 10.1016/j.softx.2021.100919.

[145] Moncef Garouani et al. “Towards an Automatic Assistance Framework for the Selection and Configuration of Machine-Learning-Based Data Analytics Solutions in Industry 4.0”. In: *The Fifth International Conference on Big Data and Internet of Things (BDIoT'21)*. Rabat, Morocco, Mar. 2021. URL: https://hal.archives-ouvertes.fr/hal-03159685.

[146] Wojciech Samek and Klaus-Robert Müller. “Towards Explainable Artificial Intelligence”. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 5–22. DOI: 10.1007/978-3-030-28954-6.

[147] Alejandro Gabriel Villanueva Zacarias, Peter Reimann, and Bernhard Mitschang. “A Framework to Guide the Selection and Configuration of Machine-Learning-Based Data Analytics Solutions in Manufacturing”. In: *Procedia CIRP*. 51st CIRP Conference on Manufacturing Systems 72 (2018), pp. 153–158. DOI: 10.1016/j.procir.2018.03.215.

[148] Hergen Wolf et al. “Bringing Advanced Analytics to Manufacturing: A Systematic Mapping”. In: *Advances in Production Management Systems. Production Management for the Factory of the Future*. 2019, pp. 333–340. DOI: 10.1007/978-3-030-30000-5_42.

[149] Adler Perotte et al. “Diagnosis Code Assignment: Models and Evaluation Metrics”. In: *Journal of the American Medical Informatics Association* 21.2 (Mar. 1, 2014), pp. 231–237. DOI: 10.1136/amiajnl-2013-002159.

[150] Erik Lindholm et al. “NVIDIA Tesla: A Unified Graphics and Computing Architecture”. In: *IEEE Micro* 28.2 (Mar. 2008), pp. 39–55. DOI: 10.1109/MM.2008.31.

[151] Chong Chen et al. “Predictive maintenance using cox proportional hazard deep learning”. In: *Advanced Engineering Informatics* 44 (2020), p. 101054. DOI: 10.1016/j.aei.2020.101054.

[152] Yaqiong Lv et al. "A predictive maintenance system for multi-granularity faults based on AdaBelief-BP neural network and fuzzy decision making". In: *Advanced Engineering Informatics* 49 (2021), p. 101318. DOI: 10.1016/j.aei.2021.101318.

[153] Rezvaneh Sahba et al. "Development of Industry 4.0 predictive maintenance architecture for broadcasting chain". In: *Advanced Engineering Informatics* 49 (2021), p. 101324. DOI: 10.1016/j.aei.2021.101324.

[154] Pavel Brazdil et al. *Metalearning: Applications to Data Mining*. Springer Berlin Heidelberg, 2009. DOI: 10.1007/978-3-540-73263-1.

[155] Joaquin Vanschoren et al. "OpenML: Networked Science in Machine Learning". In: *ACM SIGKDD Explorations Newsletter* 15.2 (June 16, 2014), pp. 49–60. DOI: 10.1145/2641190.2641198.

[156] Edesio Alcobaça et al. "MFE: Towards Reproducible Meta-Feature Extraction". In: *Journal of Machine Learning Research* 21.111 (2020), pp. 1–5. URL: http://jmlr.org/papers/v21/19-348.html.

[157] Randal S. Olson and Jason H. Moore. "TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning". In: *Automated Machine Learning: Methods, Systems, Challenges*. 2019, pp. 151–160. DOI: 10.1007/978-3-030-05318-5_8.

[158] Moncef Garouani et al. "Towards the Automation of Industrial Data Science: A Meta-Learning Based Approach". In: 23rd International Conference on Enterprise Information Systems. 2021, pp. 709–716. DOI: 10.5220/0010457107090716.

[159] Hadi S. Jomaa, Lars Schmidt-Thieme, and Josif Grabocka. "Dataset2Vec: Learning Dataset Meta-Features". In: *Data Mining and Knowledge Discovery* 35.3 (May 1, 2021), pp. 964–985. DOI: 10.1007/s10618-021-00737-9.

[160] Tin Kam Ho and M. Basu. "Complexity Measures of Supervised Classification Problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.3 (Mar. 2002), pp. 289–300. DOI: 10.1109/34.990132.

[161] Ricardo Vilalta. "Understanding Accuracy Performance Through Concept Characterization and Algorithm Analysis". In: *Workshop on Recent Advances in Meta-Learning and Future Work, 16th International Conference on Machine Learning*. 1999, pp. 3–9.

[162] Joaquin Vanschoren. "Meta-Learning: A Survey". Oct. 8, 2018. arXiv: 1810.03548.

[163] Justin Matejka and George Fitzmaurice. "Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, May 2, 2017, pp. 1290–1294. URL: https://doi.org/10.1145/3025453.3025912.

[164] Alexandros Kalousis and Melanie Hilario. "Feature Selection for Meta-learning". In: *Advances in Knowledge Discovery and Data Mining*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2001, pp. 222–233. DOI: 10.1007/3-540-45357-1_26.

[165] Yonghong Peng et al. "Decision Tree-Based Data Characterization for Meta-Learning". In: (2002). URL: https://www.semanticscholar.org/paper/Decision-Tree-Based-Data-Characterization-for-Peng-Flach/80e5ef1ac4f08b5064775e90ef698dee246d76d9.

[166] Laith Alzubaidi et al. "Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions". In: *Journal of Big Data* 8.1 (Mar. 31, 2021), p. 53. DOI: 10.1186/s40537-021-00444-8.

[167] Li Deng and Dong Yu. "Deep Learning: Methods and Applications". In: *Foundations and Trends in Signal Processing* 7.3–4 (June 30, 2014), pp. 197–387. DOI: 10.1561/2000000039.

[168] Gábor Gosztolya et al. "DNN-Based Feature Extraction and Classifier Combination for Child-Directed Speech, Cold and Snoring Identification". In: *Interspeech 2017*. Interspeech 2017. ISCA, Aug. 20, 2017, pp. 3522–3526. DOI: 10.21437/Interspeech.2017-905.

[169] Wei Wang et al. "Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. June 2014, pp. 496–503. DOI: 10.1109/CVPRW.2014.79.

[170] Vandana Bhatia and Rinkle Rani. "DFuzzy: A Deep Learning-Based Fuzzy Clustering Model for Large Graphs". In: *Knowledge and Information Systems* 57.1 (Oct. 1, 2018), pp. 159–181. DOI: 10.1007/s10115-018-1156-3.

[171] Pascal Vincent et al. "Extracting and Composing Robust Features with Denoising Autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. New York, NY, USA: Association for Computing Machinery, July 5, 2008, pp. 1096–1103. DOI: 10.1145/1390156.1390294.

[172] Francisco J. Pulgar et al. "AEkNN: An AutoEncoder kNN-Based Classifier With Built-in Dimensionality Reduction". In: *International Journal of Computational Intelligence Systems* 12.1 (Feb. 2019), pp. 436–452. DOI: 10.2991/ijcis.2018.125905686.

[173] Timothy Hospedales et al. "Meta-Learning in Neural Networks: A Survey". Nov. 7, 2020. arXiv: 2004.05439.

[174] Larry Rendell, Raj Seshu, and David Tcheng. "Layered Concept-Learning and Dynamically-Variable Bias Management". In: *In Proceedings of IJCAI-87*. Morgan Kaufmann, 1987, pp. 308–314.

[175] "Machine Learning, Neural and Statistical Classification". USA: Ellis Horwood, 1995. 267 pp.

[176] Doron Laadan et al. "RankML: a Meta Learning-Based Approach for Pre-Ranking Machine Learning Pipelines". In: *arXiv preprint arXiv:1911.00108* (2019).

[177] Bruno Feres de Souza. "Meta-aprendizagem aplicada à classificação de dados de expressão gênica". text. Universidade de São Paulo, Oct. 26, 2010. DOI: 10.11606/T.55.2010.tde-04012011-142551.

[178] Daniel Gomes Ferrari and Leandro Nunes de Castro. "Clustering Algorithm Selection by Meta-Learning Systems: A New Distance-Based Problem Characterization and Ranking Combination Methods". In: *Information Sciences* 301 (Apr. 20, 2015), pp. 181–194. DOI: 10.1016/j.ins.2014.12.044.

[179] Ömer Nebil Yaveroğlu et al. "Revealing the Hidden Language of Complex Networks". In: *Scientific Reports* 4.1 (1 Apr. 1, 2014), p. 4547. DOI: 10.1038/srep04547.

[180] Besim Bilalli, Alberto Abelló, and Tomàs Aluja-Banet. "On the Predictive Power of Meta-Features in OpenML". In: *International Journal of Applied Mathematics and Computer Science* 27.4 (Aug. 8, 2017), pp. 697–712. DOI: 10.1515/amcs-2017-0048.

[181] H. Hotelling. "Analysis of a Complex of Statistical Variables into Principal Components." In: *Journal of Educational Psychology* 24.6 (1933), pp. 417–441. DOI: 10.1037/h0071325.

[182] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition*. 3e édition. Birmingham Mumbai: Packt Publishing, 2019. 770 pp.

[183] Kevin Beyer et al. "When Is "Nearest Neighbor" Meaningful?" In: *Database Theory — ICDT'99*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1999, pp. 217–235. DOI: 10.1007/3-540-49257-7_15.

[184] John T. Hancock and Taghi M. Khoshgoftaar. "Survey on Categorical Data for Neural Networks". In: *Journal of Big Data* 7.1 (Apr. 10, 2020), p. 28. DOI: 10.1186/s40537-020-00305-w.

[185] Guijuan Zhang, Yang Liu, and Xiaoning Jin. "A Survey of Autoencoder-Based Recommender Systems". In: *Frontiers of Computer Science* 14.2 (Apr. 1, 2020), pp. 430–450. DOI: 10.1007/s11704-018-8052-6.

[186] Geoffrey E Hinton and Sam Roweis. "Stochastic Neighbor Embedding". In: *Advances in Neural Information Processing Systems*. Vol. 15. MIT Press, 2003. URL: https://proceedings.neurips.cc/paper/2002/hash/6150ccc6069bea6b5716254057a194ef-Abstract.html.

[187] Don Johnson and Sinan Sinanovic. "Symmetrizing the Kullback-Leibler Distance". In: *IEEE Transactions on Information Theory* (Mar. 20, 2001). URL: https://scholarship.rice.edu/handle/1911/19969.

[188] Noy Cohen-Shapira and Lior Rokach. "Automatic Selection of Clustering Algorithms Using Supervised Graph Embedding". June 7, 2021. arXiv: 2011.08225.

[189] Feiyu Xu et al. "Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges". In: *Natural Language Processing and Chinese Computing*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 563–574. DOI: 10.1007/978-3-030-32236-6_51.

[190] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778.

[191] Riccardo Miotto et al. "Deep Learning for Healthcare: Review, Opportunities and Challenges". In: *Briefings in Bioinformatics* 19.6 (Nov. 27, 2018), pp. 1236–1246. DOI: 10.1093/bib/bbx044.

[192] Milad Moradi and Matthias Samwald. "Post-Hoc Explanation of Black-Box Classifiers Using Confident Itemsets". In: *Expert Systems with Applications* 165 (Mar. 1, 2021), p. 113941. DOI: 10.1016/j.eswa.2020.113941.

[193] Robert L. Heath and Jennings Bryant. *Human Communication Theory and Research: Concepts, Contexts, and Challenges*. English. 2nd edition. Mahwah, N.J: Routledge, June 2000.

[194] David Gunning et al. "XAI—Explainable Artificial Intelligence". In: *Science Robotics* 4.37 (Dec. 2019). DOI: 10.1126/scirobotics.aay7120.

[195] Donghee Shin and Yong Jin Park. "Role of Fairness, Accountability, and Transparency in Algorithmic Affordance". en. In: *Computers in Human Behavior* 98 (Sept. 2019), pp. 277–284. DOI: 10.1016/j.chb.2019.04.019.

[196] Davide Castelvecchi. "Can We Open the Black Box of AI?" In: *Nature News* 538.7623 (Oct. 2016), p. 20. DOI: 10.1038/538020a.

[197] Moncef Garouani. et al. "Towards the Automation of Industrial Data Science: A Meta-learning based Approach". In: *Proceedings of the 23rd International Conference on Enterprise Information Systems - Volume 1: ICEIS*, INSTICC. SciTePress, 2021, pp. 709–716. DOI: 10.5220/0010457107090716.

[198] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-Precision Model-Agnostic Explanations". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2018).

[199] Adam W. Harley. "An Interactive Node-Link Visualization of Convolutional Neural Networks". In: *Advances in Visual Computing*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 867–877. DOI: 10.1007/978-3-319-27857-5_77.

[200] Scott M. Lundberg et al. "From Local Explanations to Global Understanding with Explainable AI for Trees". In: *Nature Machine Intelligence* 2.1 (Jan. 2020), pp. 56–67. DOI: 10.1038/s42256-019-0138-9.

[201] Grégoire Montavon et al. "Explaining Nonlinear Classification Decisions with Deep Taylor Decomposition". en. In: *Pattern Recognition* 65 (May 2017), pp. 211–222. DOI: 10.1016/j.patcog.2016.11.008.

[202] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *Computer Vision – ECCV 2014*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 818–833. DOI: 10.1007/978-3-319-10590-1_53.

[203] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". Apr. 19, 2014. arXiv: 1312.6034 [cs]. URL: http://arxiv.org/abs/1312.6034.

[204] Adam W. Harley. "An Interactive Node-Link Visualization of Convolutional Neural Networks". In: *Advances in Visual Computing*. Lecture Notes in Computer Science. 2015, pp. 867–877. DOI: 10.1007/978-3-319-27857-5_77.

[205] Kanit Wongsuphasawat et al. "Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018), pp. 1–12. DOI: 10.1109/TVCG.2017.2744878.

[206] Juliane Müller et al. "A Visual Approach to Explainable Computerized Clinical Decision Support". en. In: *Computers & Graphics* 91 (Oct. 2020), pp. 1–11. DOI: 10.1016/j.cag.2020.06.004.

[207] Dominik Sacha et al. "VIS4ML: An Ontology for Visual Analytics Assisted Machine Learning". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019), pp. 385–395. DOI: 10.1109/TVCG.2018.2864838.

[208] Daniel Smilkov et al. "Direct-Manipulation Visualization of Deep Networks". Aug. 12, 2017. arXiv: 1708.03788.

[209] Paulo E. Rauber et al. "Visualizing the Hidden Activity of Artificial Neural Networks". In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (Jan. 2017), pp. 101–110. DOI: 10.1109/TVCG.2016.2598838.

[210] Alsallakh Bilal et al. "Do Convolutional Neural Networks Learn Class Hierarchy?" In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018), pp. 152–162. DOI: 10.1109/TVCG.2017.2744683.

[211] Josua Krause, Adam Perer, and Kenney Ng. "Interacting with Predictions: Visual Inspection of Black-box Machine Learning Models". In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI '16. New York, NY, USA: Association for Computing Machinery, May 7, 2016, pp. 5686–5697. DOI: 10.1145/2858036.2858529.

[212] Mengchen Liu et al. "Analyzing the Noise Robustness of Deep Neural Networks". Oct. 9, 2018. arXiv: 1810.03913.

[213] Jiawei Zhang et al. "Manifold: A Model-Agnostic Framework for Interpretation and Diagnosis of Machine Learning Models". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019), pp. 364–373. DOI: 10.1109/TVCG.2018.2864499.

[214] Minsuk Kahng et al. "ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models". Aug. 8, 2017. arXiv: 1704.01942.

[215] Yao Ming et al. "Understanding Hidden Memories of Recurrent Neural Networks". In: *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2017). DOI: 10.1109/VAST.2017.8585721.

[216] Thilo Spinner et al. "explAIner: A Visual Analytics Framework for Interactive and Explainable Machine Learning". In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (Jan. 2020), pp. 1064–1074. DOI: 10.1109/TVCG.2019.2934629.

[217] Rossi Francesca. *AI Ethics for Enterprise AI*. 2019. URL: https://economics.harvard.edu/files/economics/files/rossi-francesca_4-22-19_ai-ethics-for-enterprise-ai_ec3118-hbs.pdf.

[218] Bum Chul Kwon et al. "RetainVis: Visual Analytics with Interpretable and Interactive Recurrent Neural Networks on Electronic Medical Records". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019), pp. 299–309. DOI: 10.1109/TVCG.2018.2865027. arXiv: 1805.10724.

[219] Nicola Pezzotti et al. "DeepEyes: Progressive Visual Analytics for Designing Deep Neural Networks". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018), pp. 98–108. DOI: 10.1109/TVCG.2017.2744358.

[220] Sugeerth Murugesan et al. "DeepCompare: Visual and Interactive Comparison of Deep Learning Model Performance". In: *IEEE Computer Graphics and Applications* 39.5 (Sept. 2019), pp. 47–59. DOI: 10.1109/MCG.2019.2919033.

[221] Mennatallah El-Assady et al. "Progressive Learning of Topic Modeling Parameters: A Visual Analytics Framework". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018), pp. 382–391. DOI: 10.1109/TVCG.2017.2745080.

[222] Carrie J. Cai et al. "Human-Centered Tools for Coping with Imperfect Algorithms during Medical Decision-Making". Feb. 8, 2019. arXiv: 1902.02960 [cs]. URL: http://arxiv.org/abs/1902.02960.

[223] Jeffrey N. Rouder et al. "Model Comparison in ANOVA". In: *Psychonomic Bulletin & Review* 23.6 (Dec. 2016), pp. 1779–1786. DOI: 10.3758/s13423-016-1026-5.

[224] Lars Kotthoff et al. "Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA". en. In: *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Cham, 2019, pp. 81–95. DOI: 10.1007/978-3-030-05318-5.

[225] Matthias Reif et al. "Automatic Classifier Selection for Non-Experts". en. In: *Pattern Analysis and Applications* 17.1 (Feb. 2014), pp. 83–96. DOI: 10.1007/s10044-012-0280-z.

[226] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA, Dec. 2017, pp. 4768–4777.

[227] Moncef Garouani et al. "Towards an Automatic Assistance Framework for the Selection and Configuration of Machine-Learning-Based Data Analytics Solutions in Industry 4.0". In: *The Fifth International Conference on Big Data and Internet of Things (BDIoT'21)*. Rabat, Morocco, Mar. 2021.

[228] Jeff Sauro and James R. Lewis. "Standardized Usability Questionnaires". In: *Quantifying the User Experience (Second Edition)*. Boston: Morgan Kaufmann, Jan. 1, 2016, pp. 185–248. DOI: 10.1016/B978-0-12-802308-2.00008-4.

[229] Tova Milo and Amit Somech. "Automating Exploratory Data Analysis via Machine Learning: An Overview". In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. SIGMOD '20. New York, NY, USA: Association for Computing Machinery, June 11, 2020, pp. 2617–2622. DOI: 10.1145/3318464.3383126.

[230] Matei Zaharia et al. "Apache Spark: A Unified Engine for Big Data Processing". In: *Communications of the ACM* 59.11 (Oct. 28, 2016), pp. 56–65. DOI: 10.1145/2934664.

[231] William Raynaut. "Perspectives de Méta-Analyse Pour Un Environnement d'aide à La Simulation et Prédiction". PhD thesis. Université de Toulouse, Université Toulouse III - Paul Sabatier, Jan. 12, 2018.

[232] Michel Lutz and Eric Biernat. *Data science : fondamentaux et études de cas: Machine Learning avec Python et R*. 1er édition. Paris: Eyrolles, Oct. 1, 2015. 296 pp.

[233] Moncef Garouani et al. *Autoencoder-kNN Meta-Model Based Data Characterization Approach for an Automated Selection of AI Algorithms*. Manuscript submitted for publication, 2022.

Appendix A

META-LEARNERS' HP SPACE

This appendix presents the full tables of the meta-learners HPs explored in the experiments performed in this thesis. For each algorithm used as meta-learner, the table shows : the selected HPs, their range of values and description.

Hyperparameter	Values	Description
complexity (or: 'C')	$[1e^{-10}, 500]$ (log-scale)	Soft-margin constant, controlling the trade-off between model simplicity and model fit.
Kernel	{'poly', 'rbf'}	The function of kernel is to take data as input and transform it into the required form (linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid).
coef0	[0., 10]	Additional coefficient used by the kernel (sigmoid kernel only).
gamma	$[1e^{-3}, 1.01]$ (log-scale)	Length-scale of the kernel function, determining its locality.
Degree	[2, 3]	Degree for the 'poly' kernel.

Table A.1: SVM hyperparameters tuned in the experiments.

Hyperparameter	Values	Description
algorithm	{SAMME, SAMME.R}	Determines which boosting algorithm to use.
N_estimators	[50, 501]	Number of estimators to build.
learning rate	[0.01, 2.0] (log-scale)	Learning rate shrinks the contribution of each classifier.
Max_depth	[1, 11]	The maximal depth of the decision trees.

Table A.2: Adaboost Hyperparameters tuned in the experiments.

Hyperparameter	Values	Description
bootstrap	{true, false}	Whether to train on bootstrap samples or on the full train set.
Max_features	[0.1, 0.9]	Fraction of random features sampled per node.
Min_samples_leaf	[1, 20]	The minimal number of data points required in order to create a leaf.
Min_samples_split	[2, 20]	The minimal number of data points required to split an internal node.
imputation	mean, median, mode	Strategy for imputing missing numeric variables.
split criterion	{entropy, gini}	Function to determine the quality of a possible split.

Table A.3: Random Forest & Extra Trees Hyperparameters tuned in the experiments.

Hyperparameter	Values	Description
max features	[0.1, 0.9]	Number of features to consider when computing the best node split.
min_samples_leaf	[1, 21]	The minimum number of samples required to be at a leaf node.
Min_samples_split	[2, 21]	The minimum number of samples required to split an internal node.
criterion	{'entropy', 'gini' }	Function used to measure the quality of a split.

Table A.4: Decision Trees Hyperparameters tuned in the experiments.

Hyperparameter	Values	Description
Learning_rate	[0.01, 1]	Shrinks the contribution of each successive decision tree in the ensemble.
criterion	{'friedman_mse', 'mse' }	The function to measure the quality of a split.
N_estimators	[50, 501]	Number of decision trees in the ensemble.
max depth	[1, 11]	Maximum depth of the decision trees. Controls the complexity of the decision trees
Min_samples_split	[2, 21]	The minimum number of samples required to split an internal node.
Min_samples_leaf	[1, 21]	The minimum number of samples required to be at a leaf node.

Table A.5: Gradient Boosting Hyperparameters tuned in the experiments.

Hyperparameter	Values	Description
C	$[1e^{-10}, 10.]$ (log-scale)	Regularization strength.
penalty	{'l2', 'l1' }	Whether to use Lasso or Ridge regularization.
Fit_intercept	True, False	Whether or not the intercept of the linear classifier should be computed.

Table A.6: Logistic Regression Hyperparameters tuned in the experiments.

Hyperparameter	Values	Description
loss	{'hinge','perceptron','log', 'squared_hinge'}	Loss function to be optimized.
penalty	{'l2', 'l1', 'elasticnet' }	Whether to use Lasso, Ridge, or ElasticNet regularization.
learning rate	{'constant', 'optimal', 'invscaling' }	Shrinks the contribution of each successive training update.
fit intercept	{True, False}	Whether or not the intercept of the linear classifier should be computed.
l1 ratio	[0., 1.]	Ratio of Lasso vs. Ridge regularization to use. Only used when the 'penalty' is ElasticNet.
eta0	[0., 5.]	Initial learning rate.
Power_t	[0., 5.]	Exponent for inverse scaling of the learning rate.

Table A.7: SGD Classifier Hyperparameters tuned in the experiments.

Appendix B

SETS OF META-FEATURES

This appendix presents the full table of meta-features used in experiments performed in this thesis. For each meta-feature it is shown : the category it belongs to, its acronym and description, adapted from the PyMFE [156] documentation.

Table B.1: Meta-features used in the experiments.

Type	Acronym	Description
Simple	classes	Number of classes
	attributes	Number of attributes
	numeric	Number of numerical attributes
	nominal	Number of nominal attributes
	samples	Number of examples
	dimension	samples/attributes
	numRate	numeric/attributes
	nomRate	nominal/attributes
Statistical	sks	Skewness
	kts	Kurtosis
	ktsp	Kurtosis for normalized datasets
	absc	Correlation between attributes
	canc	Canonical correlation between matrices
Landmarking	nb	Naive Bayes accuracy
	1nn	1-Nearest Neighbor accuracy
	dt	Decision Trees accuracy

Continued on next page

Table B.1 – continued from previous page

Type	Acronym	Description
Information theoretic	clEnt	Class entropy
	nCIEnt	Class entropy for normalized dataset
	atrEnt	Mean entropy of attributes
	jEnt	Joint entropy
	mutInf	Mutual information
	noiSig	(atrEnt - mutInfyMutInf)/mutInf
Model-based (Tree)	nodes	Number of nodes
	leaves	Number of leaves
	nodeAtr	Number of nodes per attribute
	nodeIns	Number of nodes per instance
	lev (min, max, sd)	Distributions of levels of depth
	bran (min, max, sd)	Distributions of levels of branches
	att (min, max, sd)	Distributions of attributes used
Data Complexity	f1	Maximum Fisher's discriminant ratio
	f2	Overlap of the per-class bounding boxes
	f3	Maximum feature efficiency
	f4	Collective feature efficiency
	n1	Fraction of points on the class boundary
	n2	Ratio of average intra/inter-class NN dist
	n3	leave-one-out error rate of the 1-NN
	t1	Fraction of maximum covering spheres
	t2	Average number of points per dimension
	l1	Training error of a linear classifier
	l2	Nonlinearity of a linear classifier
	l3	Minimized sum of the error distance of a linear classifier

Appendix C

LIST OF DATASETS

This appendix presents the full table of datasets used in experiments performed in this thesis. For each dataset it is shown: the dataset id, the number of instances, attributes, and classes. Figures C.1 and C.2 shows the distribution of the number of classes and attributes in all datasets.

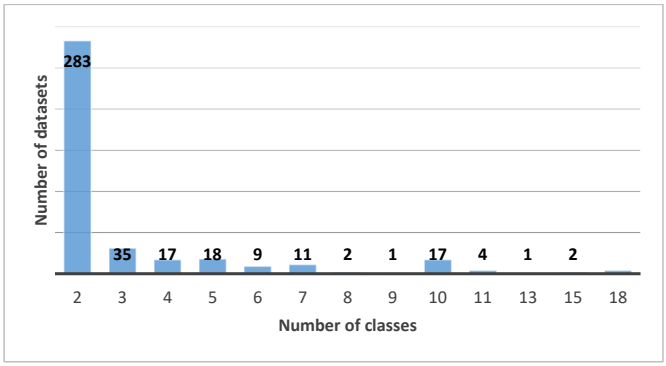

Figure C.1: Histogram of the number of classes in our datasets.

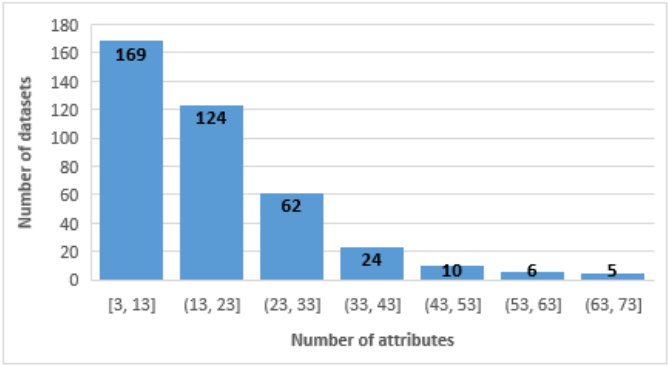

Figure C.2: Histogram of the number of attributes in our datasets.

Table C.1: Datasets used in the experiments.

Dataset	Number of		
	Intances	Attributes	Classes
D1	2000	16	10
D2	1340	11	2
D3	2310	17	7
D4	5000	40	3
D5	8192	32	2
D6	45312	5	2
D7	5100	37	2
D8	1832	11	2
D9	2800	26	5
D10	9465	39	2
D11	57999	9	7
D12	1837	11	2
D13	4999	21	3
D14	43825	11	2
D15	1000	11	2
D16	39948	12	2
D17	43824	10	2
D18	22784	8	2
D19	19020	10	2
D20	5000	19	2
D21	1831	10	2
D22	43825	11	2
D23	959	42	2
D24	1834	11	2
D25	841	66	4
D26	2000	40	10
D27	43825	11	2
D28	5404	5	2
D29	7400	20	2
D30	3200	24	10
D31	1212	17	2
D32	1941	27	7
D33	3196	36	2
D34	2800	26	5

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D35	360	9	15
D36	1833	10	2
D37	3107	6	2
D38	43825	11	2
D39	43824	11	2
D40	98049	29	2
D41	1832	11	2
D42	43825	11	2
D43	98527	23	2
D44	1941	28	7
D45	43825	11	2
D46	1836	11	2
D47	3772	26	2
D48	2000	64	10
D49	1831	10	2
D50	1832	10	2
D51	2310	18	7
D52	11183	6	2
D53	22784	16	2
D54	3772	23	3
D55	16599	17	2
D56	1000	11	2
D57	43825	11	2
D58	1837	11	2
D59	20000	16	2
D60	2800	26	5
D61	3188	60	3
D62	1832	11	2
D63	9517	5	2
D64	350	33	2
D65	2000	16	10
D66	43825	11	2
D67	5473	10	2
D68	14980	15	2
D69	3196	35	2

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D70	1832	11	2
D71	8124	21	2
D72	7200	17	3
D73	4839	5	2
D74	1000	6	2
D75	43825	11	2
D76	10000	12	2
D77	1000	26	2
D78	43825	11	2
D79	43824	11	2
D80	1830	11	2
D81	1832	11	2
D82	43824	11	2
D83	1000	26	2
D84	1835	11	2
D85	1832	11	2
D86	1600	20	2
D87	6598	17	2
D88	1000	11	2
D89	1835	11	2
D90	4052	3	2
D91	43825	11	2
D92	1834	11	2
D93	3772	27	2
D94	8192	28	2
D95	11055	30	2
D96	5000	21	3
D97	4177	8	3
D98	1833	11	2
D99	17897	8	2
D100	40768	10	2
D101	8192	8	2
D102	1600	10	2
D103	1833	11	2
D104	1000	26	2

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D105	4601	57	2
D106	1728	20	4
D107	990	13	11
D108	2310	18	7
D109	43825	11	2
D110	40768	11	2
D111	2000	6	2
D112	1832	11	2
D113	1837	11	2
D114	9822	14	2
D115	8145	17	2
D116	43825	11	2
D117	28056	6	18
D118	1834	11	2
D119	20000	16	2
D120	20000	19	5
D121	43825	11	2
D122	12958	8	4
D123	5456	4	4
D124	32769	9	2
D125	43825	11	2
D126	1828	11	2
D127	5455	24	4
D128	43825	11	2
D129	4177	3	2
D130	3163	25	2
D131	2310	18	2
D132	1000	26	2
D133	6434	37	6
D134	43825	11	2
D135	39365	10	2
D136	14240	30	2
D137	5620	63	10
D138	1835	11	2
D139	12958	8	4

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D140	5100	36	2
D141	3848	5	2
D142	5124	20	2
D143	2126	35	3
D144	3771	23	4
D145	8192	8	2
D146	2000	47	10
D147	1000	6	2
D148	88588	6	2
D149	67557	41	3
D150	44819	6	3
D151	5588	36	2
D152	1832	11	2
D153	3772	21	3
D154	43825	11	2
D155	5456	24	4
D156	1834	11	2
D157	1599	11	6
D158	9961	14	2
D159	7129	4	2
D160	19019	10	2
D161	1832	11	2
D162	4898	11	7
D163	99358	7	7
D164	6574	11	2
D165	2108	21	2
D166	20640	8	2
D167	1000	11	2
D168	8641	4	2
D169	7400	20	2
D170	5500	40	11
D171	8192	4	2
D172	43824	10	2
D173	1000	26	2
D174	14980	14	2

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D175	105908	13	5
D176	1000	6	2
D177	8192	11	2
D178	3186	19	3
D179	1832	11	2
D180	43825	11	2
D181	1600	20	2
D182	1834	11	2
D183	43825	11	2
D184	2800	26	5
D185	1941	33	2
D186	1832	10	2
D187	10935	27	3
D188	30000	22	2
D189	8192	20	2
D190	1000	6	2
D191	1212	10	2
D192	846	19	2
D193	5473	9	5
D194	40768	6	2
D195	15544	5	2
D196	100968	29	8
D197	48842	10	2
D198	2800	26	5
D199	7400	20	2
D200	402	17	2
D201	548	22	2
D202	350	18	2
D203	782	17	2
D204	7478	28	2
D205	19144	17	2
D206	3271	46	2
D207	494051	46	18
D208	1137	13	2
D209	230	47	2

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D210	1288	16	2
D211	351	17	5
D212	1040	46	2
D213	396	17	15
D214	552	12	2
D215	3229	31	2
D216	1630	30	2
D217	2329	24	7
D218	281	33	5
D219	1640	29	2
D220	9923	12	2
D221	2067	14	10
D222	378	17	2
D223	5005	15	7
D224	1221	20	2
D225	325	14	5
D226	8182	30	2
D227	332	18	2
D228	3320	69	3
D229	1507	15	3
D230	438	18	2
D231	105984	23	5
D232	619	13	2
D233	422	18	2
D234	3889	36	4
D235	1237	10	3
D236	708	61	2
D237	1796	9	2
D238	1704	7	2
D239	1323	11	3
D240	2564	10	2
D241	3139	14	2
D242	7531	27	2
D243	8207	30	2
D244	407	31	2

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D245	1698	19	6
D246	517	15	2
D247	783	11	4
D248	612	11	2
D249	5092	27	2
D250	1818	12	4
D251	1524	15	2
D252	3213	14	10
D253	1040	27	2
D254	49182	24	2
D255	2043	26	10
D256	707	12	3
D257	939	18	4
D258	689	15	2
D259	803	21	2
D260	6519	46	6
D261	937	19	2
D262	258	16	5
D263	4655	61	2
D264	3899	32	4
D265	2105	22	10
D266	256	13	2
D267	499	32	13
D268	20022	25	6
D269	550	16	2
D270	426	10	4
D271	3899	34	3
D272	494	41	6
D273	3830	36	3
D274	882	10	2
D275	302	13	3
D276	813	15	2
D277	2087	13	10
D278	11088	23	10
D279	475	53	2

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D280	997	14	2
D281	5512	16	5
D282	1624	24	2
D283	317	21	5
D284	1075	16	2
D285	426	9	2
D286	1781	29	2
D287	410	11	3
D288	212	10	3
D289	5033	46	3
D290	1688	11	2
D291	3249	40	2
D292	420	30	2
D293	100983	37	8
D294	908	10	6
D295	646	10	2
D296	983	10	2
D297	852	16	2
D298	2276	11	2
D299	1519	17	9
D300	5433	15	2
D301	5060	31	3
D302	1055	13	2
D303	783	19	2
D304	407	21	5
D305	661	15	2
D306	5663	71	10
D307	1239	15	2
D308	871	12	2
D309	331	17	2
D310	419	10	5
D311	472	15	2
D312	185	19	2
D313	1364	20	2
D314	362	9	2

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D315	484	24	2
D316	3278	30	10
D317	3262	18	3
D318	1584	13	3
D319	1805	15	2
D320	1259	12	6
D321	370	21	2
D322	767	19	2
D323	58024	16	7
D324	658	35	2
D325	883	25	4
D326	756	19	2
D327	804	16	2
D328	1112	20	11
D329	450	40	5
D330	510	17	3
D331	284	29	2
D332	5323	9	2
D333	1644	24	2
D334	7290	27	3
D335	5605	47	11
D336	28081	16	18
D337	1150	17	2
D338	466	42	2
D339	570	15	3
D340	1178	17	6
D341	241	15	2
D342	276	38	2
D343	264	19	2
D344	2099	27	10
D345	1033	24	2
D346	786	42	18
D347	13081	18	4
D348	542	14	2
D349	1047	10	2

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D350	2118	54	10
D351	212	27	2
D352	67664	47	3
D353	381	21	2
D354	1725	27	2
D355	3834	32	2
D356	1085	23	2
D357	779	24	4
D358	6734	17	2
D359	1306	65	2
D360	1910	15	2
D361	43998	9	2
D362	1935	12	2
D363	1964	17	2
D364	44000	15	2
D365	1896	11	2
D366	43907	18	2
D367	1955	18	2
D368	43918	11	2
D369	1879	13	2
D370	43947	12	2
D371	1939	15	2
D372	1980	9	2
D373	43871	16	2
D374	1927	17	2
D375	43882	14	2
D376	2009	8	2
D377	43951	9	2
D378	1990	10	2
D379	43992	14	2
D380	1934	13	2
D381	1935	17	2
D382	44002	16	2
D383	1934	9	2
D384	43927	9	2

Continued on next page

Table C.1 – continued from previous page

Dataset	Number of		
	Intances	Attributes	Classes
D385	1868	17	2
D386	1997	8	2
D387	43891	15	2
D388	2020	16	2
D389	43990	12	2
D390	1981	15	2
D391	2002	8	2
D392	43993	15	2
D393	40934	11	2
D394	1052	12	2
D395	1135	33	2
D396	1090	6	2
D397	1104	8	2
D398	1143	24	2
D399	1170	13	2
D400	1123	18	2

Appendix D

AeKNN COMPLETE EVALUATION RESULTS

This appendix presents the full table of datasets used in the emperical evaluation of AeKNN meta-model. Figure D.1 shows results of RF, XGB, KNN and AeKNN meta-models for recommending optimal pipelines for test data.

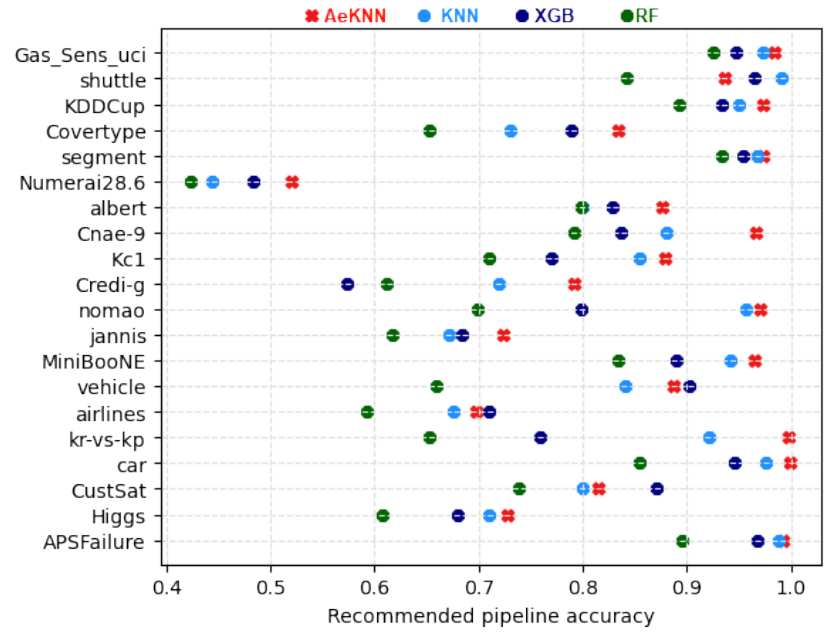

Figure D.1: Performance of baseline meta-models relative to AeKNN on the benchmark datasets.

Dataset	Number of		
	Instances	Attributes	Classes
APSFailure	60000	171	2
CustSat	76020	14	2
car	1728	7	4
kr-vs-kp	3196	37	2
airlines	5473	8	2
vehicle	8463	19	4
MiniBooNE	52147	51	2
Higgs	110000	9	2
jannis	8641	55	4
nomao	31772	119	2
Credi-g	30000	21	3
Kc1	2108	15	4
Cnae-9	63260	33	2
albert	43824	13	3
Numerai28.6	6574	9	4
segment	2310	24	4
Covertype	25524	9	5
KDDCup	49402	25	4
shuttle	57999	37	2
Gas_Sens-uci	4148	21	2

Table D.1: List of benchmark datasets used in the evaluation.

Towards Efficient and Explainable Automated Machine Learning Pipelines Design
Application to Industry 4.0 Data

Abstract

Machine learning (ML) has penetrated all aspects of the modern life, and brought more convenience and satisfaction for variables of interest. However, building such solutions is a time consuming and challenging process that requires highly technical expertise. This certainly engages many more people, not necessarily experts, to perform analytics tasks. While the selection and the parametrization of ML models require tedious episodes of trial and error. Additionally, domain experts often lack the expertise to apply advanced analytics. Consequently, they intend frequent consultations with data scientists. However, these collaborations often result in increased costs in terms of undesired delays. It thus can lead risks such as human-resource bottlenecks. Subsequently, as the tasks become more complex, similarly the more support solutions are needed for the increased ML usability for the non-ML masters. To that end, Automated ML (AutoML) is a data-mining formalism with the aim of reducing human effort and readily improving the development cycle through automation.

The field of AutoML aims to make these decisions in a data-driven, objective, and automated way. Thereby, AutoML makes ML techniques accessible to domain scientists who are interested in applying advanced analytics but lack the required expertise. This can be seen as a democratization of ML. AutoML is usually treated as an algorithms selection and parametrization problem. In this regard, existing approaches include Bayesian optimization, evolutionary algorithms as well as reinforcement learning. These approaches have focused on providing user assistance by automating parts or the entire data analysis process, but without being concerned on its impact on the analysis. The goal has generally been focused on the performance factors, thus leaving aside other important and even crucial aspects such as computational complexity, confidence and transparency. In contrast, this thesis aims at developing alternative methods that provide assistance in building appropriate modeling techniques while providing the rationale for the selected models. In particular, we consider this important demand in intelligent assistance as a meta-analysis process, and we make progress towards addressing two challenges in AutoML research. First, to overcome the computational complexity problem, we studied a formulation of AutoML as a recommendation problem, and proposed a new conceptualization of a Meta-Learning (MtL)-based expert system capable of recommending optimal ML pipelines for a given task; Second, we investigated the automatic explainability aspect of the AutoML process to address the problem of the acceptance of, and the trust in such black-boxes support systems.

To this end, we have designed and implemented a framework architecture that leverages ideas from MtL to learn the relationship between a new set of datasets meta-data and mining algorithms. This eventually enables recommending ML pipelines according to their potential impact on the analysis. To guide the development of our work, we chose to focus on the Industry 4.0 as a main field of application for all the constraints it offers. Finally, in this doctoral thesis, we focus on the user assistance in the algorithms selection and tuning step. We devise an architecture and build a tool, `AMLBID`, that provides users support with the aim of improving the analysis and decreasing the amount of time spent in algorithms selection and parametrization. It is a tool that for the first time does not aim at providing data analysis support only, but instead, it is oriented towards positively contributing to the trust-in such powerful support systems by automatically providing a set of explanation levels to inspect the provided results.

Keywords: data analysis, machine learning, automl, explainable ai

Vers une automatisation efficace et explicable des processus d'apprentissage automatique

Application à l'Industrie 4.0

Résumé

L'industrie du futur introduit de nouveaux concepts, processus et pratiques conduisant à des mutations profondes dans le pilotage des systèmes d'information associés. Une des problématiques cruciales est l'utilisation de la quantité importante de données, notamment celles produites par les différents dispositifs d'acquisition de données (Cyber Physical Systems, etc.), pour en extraire de la connaissance destinée à la maîtrise des processus de l'entreprise à travers un système d'information évolutif, réactif et adapté aux spécificités de l'industrie 4.0.

L'intelligence artificielle et plus particulièrement l'apprentissage automatique fournit les algorithmes, méthodes et outils permettant l'extraction de connaissances et de modèles à partir des données représentant l'activité d'une entreprise et son environnement, et l'apport de plus d'automatisation des processus sous-jacents. Cependant, de nombreuses entreprises ne disposent pas de moyens humains leur permettant de déployer efficacement des solutions d'apprentissage automatique. Cela s'explique notamment par le fait que la construction de telles solutions est un processus long et difficile qui nécessite une expertise hautement technique et intersectorielle et qui est une ressource limitée. Nous nous intéressons donc à ce besoin d'assistance à l'analyse de données, qui commence à recevoir une certaine attention des communautés scientifiques, donnant naissance au domaine dit d'apprentissage automatique automatisé.

L'apprentissage automatique automatisé est devenu un domaine en plein essor qui vise à rendre l'application des méthodes d'apprentissage automatique aussi dépourvue d'intervention humaine que possible. A cet égard, les approches existantes se révèlent souvent similaires et peu abouties. Ces approches sont concentrées sur l'assistance de l'utilisateur en automatisant une partie ou l'ensemble du processus d'analyse de données, mais sans se soucier de son impact sur l'analyse. L'objectif a généralement été axé sur les facteurs de performance, laissant ainsi de côté d'autres aspects importants, voire cruciaux, tels que la complexité du calcul, la confiance et la transparence.

Cette observation nous a amenés à orienter nos recherches vers le domaine du Meta-Apprentissage (MtL) et à développer des méthodes alternatives qui apportent une aide à la construction des techniques de modélisation appropriées tout en fournissant le rationnel des modèles ML sélectionnés. En particulier, nous considérons cette demande importante d'assistance intelligente comme un processus de méta-analyse, et nous progressons vers la résolution de deux défis de la recherche en AutoML. Dans un premier temps, pour palier au problème de la complexité du calcul, nous avons étudié une formulation de l'AutoML en tant que problème de recommandation, puis proposé une nouvelle conceptualisation d'un système expert basé sur le MtL capable de recommander des pipelines ML optimaux pour une tâche donnée. Dans un second temps, nous avons traité l'explicabilité du processus d'aide à la décision de l'AutoML pour prendre en compte la problématique de l'acceptation et la confiance en ces systèmes généralement vus comme des boîtes noires.

Mots clés : analyse de données, apprentissage automatique, automl, explicabilité de l'ia