UNIVERSIDAD DE DEUSTO

AUTOMATED MACHINE LEARNING TO SUPPORT MODEL SELECTION IN SUPERVISED TRAFFIC FORECASTING

by

Juan Sebastian Angarita Zapata

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, within the PhD Program in Engineering for the Information Society and Sustainable Development

Supervised by Dr Antonio Masegosa and Dr Isaac Triguero

UNIVERSIDAD DE DEUSTO

AUTOMATED MACHINE LEARNING TO SUPPORT MODEL SELECTION IN SUPERVISED TRAFFIC FORECASTING

by

Juan Sebastian Angarita Zapata

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, within the PhD Program in Engineering for the Information Society and Sustainable Development

Supervised by Dr Antonio Masegosa and Dr Isaac Triguero

The candidate

The supervisors

Bilbao, October 2020

Automated Machine Learning to support Model Selection in Supervised Traffic Forecasting

Author: Juan Sebastian Angarita Zapata

Supervisors: Dr Antonio Masegosa and Dr Isaac Triguero

Text printed in Bilbao

First edition, October 2020

To the universe and nature that are always challenging and rewarding life.

Abstract

Sensing and telecommunications technologies are generating vast volumes of data that motivate the use of data-driven approaches, with a particular interest in Machine Learning (ML), to analysing this data. Like other research areas, Intelligent Transportation Systems (ITSs) announce the production of tons of hardly manageable traffic data that can be used by different applications such as traveller information systems or Traffic Forecasting (TF) schemes. Recently, TF is gaining relevance due to its ability to deal with traffic congestion through forecasting future states of different traffic measures (e.g., travel time). From a ML perspective, TF is approached through learning and approximating a mapping function from historical data to make traffic predictions when facing unseen data. TF literature reports diverse ML methods that have led to taxonomy proposals that classify the methods based on the mathematical assumptions from which they operate. Conversely, only a few efforts have been dedicated to categorising TF problems and how they are modelled to be approachable by ML methods. Therefore, without knowing what types of TF problems there are, it is hard to figure out the best ML methods in each TF scenario.

TF poses two main challenges to the ML paradigm. First, traffic data can be collected in multiple formats (e.g., traffic-counting measures, GPS tracks) and under different transportation circumstances (e.g., urban, freeway). These characteristics influence the performance of ML methods, and choosing the most competitive method from a set of candidates brings human effort and time costs. Second, raw traffic data usually needs to be preprocessed before being analysed. Therefore, deciding the most suitable combination of data preprocessing techniques

and ML method is a time-consuming task that demands specialised ML knowledge to approach it.

Automated Machine Learning (AutoML) arises as a promising approach that addresses the issues mentioned above in problem domains wherein expert ML knowledge is not always an available or affordable asset such as TF. Specifically, AutoML aims at automatically finding ML pipelines (the workflow from data preprocessing to model validation) that can be competitive on input data without knowing the problem domain wherein the data comes from (general-purpose). AutoML methods have been broadly used in other areas; however, it has been underexplored in TF. The latter raises the question if general-purpose AutoML guarantees competitive results while reducing the human-time costs of ML in TF. However, current AutoML approaches suffer from issues that can also affect its performance in TF as well as in other ML problems. The optimisation process to find competitive pipelines is complicated and computational costly because of the diversity of the search space (multiple preprocessing and ML techniques) and the high evaluation cost of the objective function. Alternative learning approaches (e.g., meta-learning) have been designed to try to overcome these issues, but they could not properly work on diverse datasets such as TF. This context claims for more robust and efficient AutoML methods that can contribute not only to the general-purpose domain but also to TF.

Therefore, this thesis focuses on the development of new AutoML approaches more suited to specific problem domains that can offer also competitive results in TF. The research conducted here is completed through three progressive stages. First, we introduce a new taxonomy that categorises and provides a systematic view of different supervised TF problems. It consolidates traffic-related criteria and proposes new ML modelling attributes that enable to identify well-established trends and gaps in TF from a ML perspective. The second stage comprises a thorough study of AutoML in supervised learning problems.

Concretely, we evaluate the competitiveness of current AutoML approaches using a sample of supervised TF problems extracted from the proposed taxonomy. This analysis allows us to identify key points to improve the performance of AutoML in the general-domain as well as in TF. Therefore, in the third stage of the research, we present a new AutoML method for supervised problems, such as TF, with a search strategy based on the construction of ensembles of multiple classifiers. The novelty of this approach lies in its simplicity, competitiveness and scalability that overcomes some on the most common issues of AutoML.

In summary, this research introduces a novel and improved AutoML mechanism able to better adapt to specific problem domains using data preprocessing techniques, ML methods and raw data. The proposed method can lead to better or competitive results in the general-purpose domain and TF with respect to the state-of-the-art. This is accomplished by taking advantage of the automated generation of ensembles from a predefined set of ML pipelines. The use of these multiple classifier systems significantly speed up the AutoML process, and it also opens the path towards AutoML frameworks based on ensemble strategies. Therefore, this automated learning approach becomes a promising alternative to develop more straightforward AutoML methods that are potentially more suitable to tackle datasets of different sizes.

Acknowledgements

I want to thank my supervisors Antonio Masegosa and Isaac Triguero, for the constant trust they have placed in me over the years. Thank you for allowing me to define my professional and scientific path while guiding, challenging, and helping me with your ideas, comments, and opportunities. Thank you, Antonio, for teaching me that many challenges can solvable with creativity, and when things get hard, you only need to stay calm and keep working. Thank you, Isaac, for teaching me that there is an opportunity to do things better in any challenge, which then gives you the spark not to stop enjoying what you do.

Thanks to Asier Perallos, Pili Elejoste, Alfonso Bahillo, Osane Uriarte, and the DIRS-COFUND office for allowing me to come here, 11 thousand kilometres from Bucaramanga, and be part of Deusto Smart Mobility, DeustoTech, and the University of Deusto. I also want to thank my workmates, friends, and former colleagues (Aimar, Alex, Amgad, Asier, Alejo, Enrique, Florian, Galder, Hugo, Iker, Idoia, Itziar, Jenny, Laura, Leire, Luis, Nacho, Naia, Natasa, Pablo, Pedro, Timothy). Sharing the day-to-day coffee with all of you makes that Bilbao becomes home. Thanks also to Lam, Manuel, Rebecca, and Song for sharing and enjoying with me three months at the Computational Optimisation and Learning Lab at the University of Nottingham.

Thanks to my friends in Colombia for all these years sharing laughter and life no matter the distance. I also want to thank Cesar for being a partner for almost 17 years of my life, no matter how far or near we maybe.

Of course, none of this would have been possible without my family. Thanks to my parents Hugo and Claudia, and my sister Laura. Thanks

for your infinite love and support. You are the basis of everything I have achieved today; no matter where we are or how we are, I will always love you.

And last but not least, eternal thanks to Andrea for accompanying me throughout this life and professional challenge. Thank you for putting up with my boring engineering stuff and recurrent self-justifications, for your patience in overcoming the distance that has separated us temporarily, and most important, thank you for your unconditional love and support.

Thank you everyone,

Juan

October 2020

Contents

List of Figures

List of Tables

Acronyms

AutoML Automated machine learning

BAs Base algorithms

DL Deep learning

DPP Data preprocessing

ET Execution time

Fw Freeway

IR Imbalance ratio

ITSs Intelligent transportation systems

kNN k-Nearest neighbours

LoS Level of service

ML Machine learning

MSP Model selection problem

NNs Neural networks

RF Random forest

SVMs Support vector machines

TF Traffic forecasting

Ub Urban

VDS Vehicle detection systems

The beginning is the most important part of the work.

Plato

CHAPTER

1

Introduction

ITSs are revolutionising how transport and mobility are conceived. Its primary purpose is to provide high-quality transportation services to passengers, transport managers, and policymakers [3]. ITS applications include public transport information services, cooperative vehicular systems or TF schemes, among others [3, 4]. During the last years, TF has been a critical component of ITSs to tackle traffic congestion, as it aims to predict near-future traffic measures based on current and historical traffic information [5]. TF allows travellers to plan their movements ahead of time and assists decision-makers to improve the management of traffic flows.

Initially, TF was focused on the prediction of traffic at single locations within freeway or urban environments. This was approached employing traffic theory [6] and classical statistical methods [7]; however, growing sensing and telecommunications technologies of ITSs are generating big volumes of traffic data that the former modelling approaches can hardly analyse. The latter has changed the TF modelling paradigm towards a data-driven approach [5, 8–10], placing particular emphasis on ML methods [11]. The main strengths of ML are its ability to predict traffic without the need of knowing theoretical traffic mechanisms, and its capacity to mine complex spatial-temporal relationships underlying traffic data.

As well as in other research areas that are faced with the production of huge and complex volumes of data, upcoming ITSs announce the generation of traffic data

at rates higher than the ones reported today. This availability of data poses multiple challenges to ML that range from complex spatial-temporal traffic patterns embedded in data, to the high demand for ML knowledge to develop suitable methods that can effectively analyse and mine this data. However, expert ML knowledge is an expensive and sometimes a scarce resource; which ends up limiting the application of ML in different research areas such as TF. Therefore, the aim of this thesis is contributing to the development of automated methods able to deal with raw-complex data and the absence of ML knowledge. This research endeavour can satisfy the on-going demands of TF as well as the ML demands of other data-driven fields.

In this chapter, the motivation for this research along with the research questions that naturally arise are discussed in Section 1.1. After this, the objectives and research methodology are presented in Sections 1.2 and 1.3, respectively. Next, the contributions and scientific publications are summarised in Section 1.4. Finally, the research context and the outline of this thesis are presented in Sections 1.5 and 1.6, respectively.

1.1 Motivation and Scope

TF is being influenced by the high availability of data provided by ITSs that at the same time motivates the use of data-driven modelling approaches [9], particularly ML as it was exposed above. Some technologies such as Automatic Vehicle Identification, Electronic Tolls, and GPS collect individual traffic data related to each vehicle on the roads; meanwhile, others collect macroscopic traffic measures (averages of many cars) such as Vehicle Detection Systems (VDS). Based on VDS data, the most common type of data available and used in TF literature [12], two supervised ML modelling approaches can be typically used. Whether the traffic measure to be predicted is continuous (e.g., speed or flow), TF should be approached as a supervised regression problem. When the expected value is discrete, the prediction should be addressed as a supervised classification problem (e.g., Traffic Level of Service - LoS). In both cases, the purpose of ML is to train a model that learns and approximates a mapping function (based on historical traffic data), in such a way that when the model faces new and unseen data, it can make accurate predictions.

Transportation literature reports a great variety of ML methods such as Neural Networks (NNs), Support Vector Machines (SVMs), k-Nearest Neighbours (kNN) or Random Forest (RF), among others [5, 9]. This variety has led to different taxonomy proposals that categorise the methods based on the mathematical assumptions from which they operate with respect to traffic theory and statistical methods [12–16]. Similarly to ML [11], the methods are categorised as parametric and non-parametric. The former category assumes the relationship between the explanatory and response variables as known (e.g., logistic regression, perceptron); while methods of the latter category model non-linear relationships without requiring the mentioned assumption (e.g., NNs, RF).

Contrarily, a few efforts have been directed to classify the TF problems and how they can be modelled to be approachable by the aforementioned methods. Therefore, without knowing what types of TF problems there are, it is difficult to figure out the strengths and weaknesses of ML methods in each particular problem. These problems must be classified by characteristics of the transportation scenario (type of data source, context of predictions, etc.) and the modelling specifications imposed by the ML paradigm. However, systematically define and organise how TF problems can be modelled from a supervised learning perspective remains unexplored to date. The research presented here addresses this gap.

TF poses a twofold challenge to the supervised ML paradigm. In the first place, traffic data allows the prediction of traffic under different transportation scenarios that can range from making predictions at multiple freeway segments to forecasting traffic at a whole road network [5]. These characteristics of the transportation context influence the performance of ML models [9] and, therefore, selecting the most appropriate method from a pool of candidates is a time-consuming task that involves a high human effort. The latter can cause the success of ML happens at a high price, especially in research areas wherein expert ML knowledge is not always available such as the case of TF. In the second place, raw traffic data cannot usually be directly fed into ML methods as it could have missing values, noise and diverse formats (e.g., GPS tracks, traffic-counting measures). These data imperfections need to be preprocessed before being correctly analysed and mined [17]. Hence, choosing the most suitable ML method must also include the selection of preprocessing approaches to format the data in shape accepted by ML algorithms and,

thus, being able to exploit the hidden patterns of traffic data [18]. Although this enhances the contributions of ML into TF, it also adds extra human effort and time cost due to the search of suitable data preprocessing techniques that better fit the TF problem at hand and the ML method selected. Consequently, defining the most suitable ML pipeline structure (the combination of data preprocessing techniques with a ML method) is a complex problem that makes expert ML knowledge more necessary, while keeps demanding time, human effort, and high computational capacities.

The automation of the complete ML workflow, known as AutoML, arises as a promising approach to reduce the human effort and time cost of ML in research areas wherein specialised ML knowledge is not an asset that is always available or affordable. Concretely, AutoML seeks to automatically find competitive ML pipeline structures, which maximise or minimise a performance metric on input data without being specialised in the problem domain (general-purpose) wherein the data comes from [19]. AutoML approaches usually follow two strategies for the automated construction of ML pipelines. On the one hand, there could be a purely-based optimisation process that tests different promising combinations. They are generated from a predefined base of preprocessing and learning algorithms to minimise/maximise an objective function during a time budget predefined by the user [20, 21]. On the other hand, there is an alternative approach based on a hybrid search where the optimisation is complemented with meta-learning [1, 22]. In this latter case, the learning approach is in charge of systematically observing how different ML pipelines perform on a wide range of tasks to take advantage of this experience to learn new tasks faster. Thus, when new data is given, meta-learning recommends pipelines that are likely to perform well on the input dataset based on previous experience. This selection of pipelines is later on used for a warm-start of the optimisation process.

AutoML methods have been successfully used in other areas; however, an extensive analysis to determine the strengths and weaknesses of the search strategies mentioned above has not been carried out in very diverse learning tasks such as TF. To the best of our knowledge, only one paper out of this thesis has used AutoML concepts in TF [23]. Although this study showed promising results, it does not consider the complete automation of ML pipelines for TF and does not employ

current state-of-the-art AutoML approaches. In this sense, determining to what extent general-purpose AutoML can be competitive in TF is far from being fully answered. Specifically, in-depth efforts are needed to research whether general-purpose AutoML can offer competitive results with respect to specific-purpose ML methods in tasks such as TF. The research presented here addresses this challenge.

Answering whether or not AutoML works in specific areas such as TF can lead to determine if general-purpose approaches are sufficient to play the role of ML experts and recommend competitive pipelines. However, current AutoML methods suffer from some issues that can affect their performance in TF as well as in other research areas. As it was exposed above, the core of general-purpose AutoML is based on the fine-tuning of pipelines that later can be or not integrated on ensembles [1, 24]. This optimisation process is computationally expensive due to the complexity of the search space (variability of data preprocessing techniques, ML methods and hyperparameters) and the high evaluation cost of the objective function in big datasets. In this context, meta-learning is a suitable strategy to deal with these issues and, therefore, it can improve the performance of AutoML. Nevertheless, it is not easy to characterise a wide variety of learning tasks entirely. Therefore, the assumption of always being able to suggest competitive pipelines for unseen learning task (e.g., TF) that are different from the ones used to train the meta-learning approach needs more research. The latter would be not only useful for the transportation community, but also to other fields not limited to TF.

Considering the limitations mentioned above, the automated process of generating and testing ML pipelines shows a wide margin of improvement. This offers the opportunity to develop new and novel AutoML approaches towards more robust and efficient strategies for the automated construction of learning pipelines, which can be better adapted to specific problems domains. In this sense, a new automated learning approach of ML pipelines can contribute not only to general-purpose AutoML but also to its performance in TF problems. The research presented here addresses this challenge.

Summarising, multiple research questions derive from the motivations presented above:

1. What types of problems are there in TF and how are they modelled from a supervised learning paradigm?

2. Can general-purpose AutoML properly work in TF? What are its strengths and weaknesses dealing with supervised learning problem such as TF?

3. Is it possible to build a new general-purpose AutoML approach that mitigates current AutoML's drawbacks and better adapts to specific learning tasks such as TF?

1.2 Objectives

As it was stated above, the overall aim of this thesis is to investigate and contribute to the development of new automated methods able to leverage raw-complex data and supply the absence of ML knowledge in specific problem domains such as TF. To achieve this purpose, we define the following progressive objectives:

- **Objective 1**: To understand, organise and systematise the existing knowledge about TF problems modelled from a classical supervised learning perspective. *This objective corresponds to the first research question.*

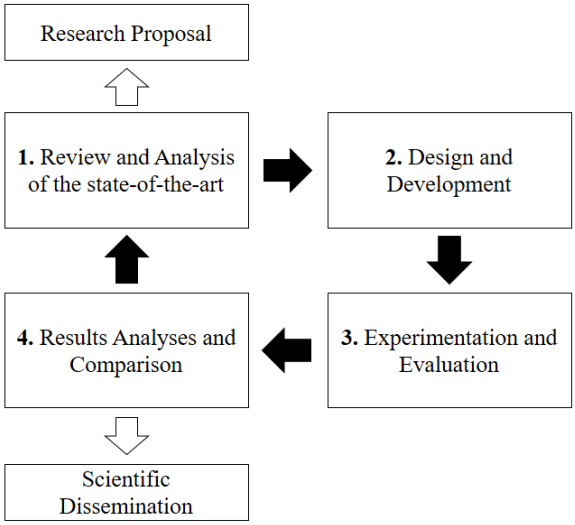
- **Objective 2**: To characterise the performance of AutoML, based on optimisation and meta-learning integrated with optimisation, in supervised learning problems using TF as an application area. *This objective corresponds to second research question.*

- **Objective 3**: To improve the performance of AutoML in supervised learning problems, including TF. *This objective corresponds to the third research question.*

1.3 Research Methodology

The research field of this thesis is evolving fast due to technological advances and the continuous generation of new knowledge in TF and ML. Therefore an iterative research methodology that allows us to review the state-of-the-art regularly was

Figure 1.1: Research methodology of this thesis composed of four main stages linked between them in a cyclical way



followed. The main idea of this cyclical process is that the knowledge acquired in its initial phases helps us to design increasingly original contributions capable of improving the understanding and knowledge in the areas wherein this thesis is focused. This cyclical process has multiple iterations done during the three years of this PhD thesis. Figure 1.1 shows the different phases of this research methodology, and they are briefly described below:

1. **Review and Analysis of the state-of-the-art:** this stage is focused on investigating the state-of-the-art related to the fields (TF and ML) under consideration to identify gaps and challenges in current literature. To achieve this aim, the relevant bibliography is used, reviewing both publications from the scientific community published in journals and proceedings of world-wide conferences. The knowledge acquired in this phase led to formulate the research proposal in the first year of this PhD.

2. **Design and Development:** in this phase, different proposals to approach the identified challenges are designed and developed. To this end, previously acquired or updated knowledge (new literature review) is used to ensure that the solution is always up-to-date with the current state-of-the-art.

3. **Experimentation and Evaluation:** the goal of this phase is to test the proposals resulting from the previous step to a process of experimentation. To carry out this procedure, it is crucial to provide some criteria and evaluation methods with which the results will be compared in the subsequent phase. All these criteria and methods must be built using the knowledge acquired in the first stage of the methodology.

4. **Results Analyses and Comparison:** after carrying out experimentation, results must be analysed and contrasted with those obtained in the state-of-the-art. At this point, it is needed to check if the results obtained are enough to address the challenges identified in the first phase. In such a case, another methodological cycle begins to approach the following challenge identified or to keep working with the challenge under consideration if it was not still solved. In this stage, conclusions must be drawn from analyses of results and knowledge obtained must be materialised in scientific dissemination, either through journals, books, or conferences.

1.4 Contributions and Publications

The work presented in this dissertation focuses on the exploration of novel automated learning approaches for TF problems, which has produced one of the largest and most consolidate bodies of research in ITSs.The main contributions of this thesis and their associated scientific production are presented below:

- A new taxonomy that provides a panoramic view of the different TF supervised problems to provide a common framework that helps to display similarities and differences among the problems. This taxonomy unifies and consolidates traffic-related criteria available in the literature. Besides, it introduces new criteria to categorise TF problems with respect to how they can be modelled from a supervised learning perspective. *This contribution is approached in Chapter 3 and the scientific dissemination associated to it is:*

  Title: A Taxonomy of Traffic Forecasting Regression Problems From a Supervised Learning Perspective.

Authors: Angarita-Zapata JS, Masegosa AD, Triguero I.

Journal: IEEE Access (Impact Factor = 4.098 $\rightarrow$ Q1).

Status: Published. Vol. 7, pp. 68185-68205, 2019.

- A thorough study of general-purpose AutoML when dealing with supervised learning problems using TF as a study case. This is accomplished by an in-depth analysis of AutoML approaches, based on optimisation and meta-learning with optimisation, in two families of TF problems. Thus, we identify AutoML's strengths and drawbacks that give guidelines to improve the performance of general-purpose AutoML in diverse learning tasks such as TF. *This contribution is approached in Chapter 4 and the scientific works associated to it are:*

  1. Title: General-purpose Automated Machine Learning for Transportation: A Case study of Auto-sklearn for Traffic Forecasting.

  Authors: Angarita-Zapata JS, Masegosa AD, Triguero I.

  Congress: 18th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, 2020, Lisbon (Portugal).

  2. Title: Evaluating Automated Machine Learning on Supervised Regression Traffic Forecasting Problems.

  Authors: Angarita-Zapata JS, Masegosa AD, Triguero I.

  Book Chapter: Computational Intelligence in Emerging Technologies for Engineering Applications.

  Status: Published in Studies in Computational Intelligence, vol 872. Springer Cham, 2020.

  3. Title: A Preliminary Study on Automatic Algorithm Selection for Short-Term Traffic Forecasting.

Authors: Angarita-Zapata JS, Triguero I, Masegosa AD.

Congress: XII International Symposium on Intelligent and Distributed Computing, 2018, Bilbao (Spain).

- A new AutoML method for supervised classification problems with a pipeline search strategy purely based on the construction of multiple classifiers. This novel AutoML method can lead to better or competitive results in comparison to the state-of-the-art, becoming a very promising alternative to develop more straightforward AutoML methods that are potentially more parallelisable and suitable to tackle bigger datasets. *This contribution is approached in Chapter 5 and the scientific dissemination associated to it is:*

  Title: Ensembles are all you need: An AutoML method based on ensembles of predefined machine learning pipelines.

  Authors: Angarita-Zapata JS, Masegosa AD, Triguero I.

  Status: Under review - paper submitted in May 22, 2020.

1.5 Research Context

In this section, information about the funding, institutions and research stay that have supported this research is provided.

1.5.1 Research Support and Funding

This research has been funded by the European Union Horizon 2020 Research and Innovation Programme under Agreement 815069 and by the Marie Sklodowska-Curie under Agreement 665959. This research has also been funded and supported by Deusto Smart Mobility Research Group, DeustoTech-Fundación Deusto, the Faculty of Engineering at the University of Deusto (Spain), and the Computational Optimisation and Learning Lab at the University of Nottingham (UK).

1.5.2 Research Stay

During the second year of the PhD, an international research stay was made as part of the research activities. The research stay was carried out at University of Nottingham (UoN) within the Computational Optimisation and Learning (COL) Lab [1]. The stay lasted three months, from April to July 2019, under the supervision of Dr Isaac Triguero. He is a supervisor of this research and Associate Professor at the School of Computer Science in UoN. The objective of the research stay was to collaborate with international experts in the ML field to share knowledge and get feedback from them. Therefore, it was possible to study the basics of ML deeply, data preprocessing, and AutoML that later on ends with the production of one book chapter and one journal paper.

1.6 Structure of the Dissertation

The structure of the remainder of this thesis dissertation is outlined below.

Chapter 2 reviews background and related work about ML and ITSs with special emphasis in AutoML and TF modelled from a ML perspective.

Chapter 3 presents the proposed taxonomy that is built based on traffic and ML modelling specifications to categorise supervised TF problems. This chapter is therefore aligned with *Specific Objective 1*.

Chapter 4 provides a thorough analysis of the benefits and drawbacks of general-purpose AutoML when dealing with supervised regression and classification TF problems. The work presented in this Chapter is therefore directly related to *Specific Objective 2*.

Chapters 5 introduces *AutoEn*, a simple and efficient AutoML method that approaches some of the drawbacks of AutoML methods identified in Chapter 4. This Chapter is aligned with *Specific Objective 3*.

[1] www.nottingham.ac.uk/research/groups/col/index.aspx

Chapters 6 revisits the main goal and specific objectives posed in this PhD thesis, summarises the main contributions of this research, and outlines possible future research.

The only true wisdom is in knowing
you know nothing.

Socrates

CHAPTER

2

Background and Related Work

Increasing computational capacities and telecommunications technologies are producing countless quantities of raw data in diverse research fields. This data embeds valuable information that can be used to solve different data-driven problems. Therefore, raw data needs to be mined and analysed to comprehend the patterns that lie behind it. Such mining process could be done manually by experts of the areas wherein the data is generated; nevertheless, the convergence of computing and communications are creating significant volumes of data that are hardly manageable by hand. In this context, the development of automated approaches of data analysis is fully required to be able to extract efficiently all the valuable knowledge embedded in this data.

This thesis delves into automated methods to disentangle that valuable knowledge from the data, bridging gaps between ML, TF in the context of ITSs, and AutoML approaches. In this chapter, the essential background and related work on these areas is provided. First, ITSs and its main application areas are introduced in Section 2.1. Afterwards, Section 2.2 presents the TF problem that is a key component of ITSs and the central application area of this research. Then, Section 2.3 places concepts of ML with special emphasis on supervised learning and provides

an overview of the ML workflow from data preprocessing to model selection. Following, Section 2.4 gives an introduction and presents related work of AutoML. Finally, Section 2.5 summarises background and related work of how TF can be addressed using ML.

2.1 Intelligent Transportation Systems

Computers, localisation and positioning systems, and sensing technologies are playing an increasingly important role as instruments used for different purposes under different conditions to improve our transportation systems. In this context, ITSs are the set of applications that make use of telecommunications and computation to provide transport services that can be used by passenger, freight transport, transport managers, and policymakers [3].

ITSs consist of systems responsible for collecting data concerning the state of traffic, systems in charge of processing and integrating the data and, finally, systems that are responsible for mining the data and providing transport information to the users mentioned above. Therefore, the data collected in real-time by ITSs may be used to determine the state of the transportation network, to plan a trip, to manage the traffic dynamically, or even to report data from a logistics operator to the customer, among others [25].

Recently, ITSs are addressing significant changes due to huge volumes of traffic data stored by from a wide variety of sources. This data availability can potentially lead to a new generation of ITSs going from conventional technologies and systems to data-driven ITSs. Such evolution of ITSs opens the path towards applications such as advanced traveller information systems, advanced public transportation systems, advanced urban transportation systems, among others [26, 27]. Those applications mentioned above are different in their nature and purposes. However, one element shared by all of them is the prediction of future states of different traffic measures (e.g., speed, travel time), which represent valuable information for the stakeholders of each application.

The prediction of traffic is highly influenced by data that in the majority of the cases has imperfections, heterogeneity and comes in multiple formats due to the diversity of data sources. These data characteristics demand different actions such

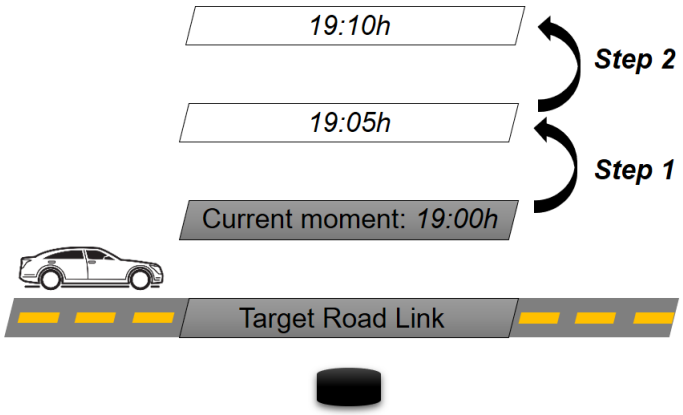
as data cleaning, data imputation and dimension reduction to improve the quality of the data and guarantee the minimum format to analyse and mine it using data-driven methods [25]. In this sense, ML appears as a promising data-driven approach to deal with these challenges through different learning strategies that can address both the preprocessing of data and the data mining processes.

Within this research, we focus on the area of traffic prediction approached from a ML paradigm. These two research areas are presented with more details in the following sections.

2.2 Traffic Forecasting

Within ITSs research, TF has been a relevant topic during the last three decades due to its active role in traveller and traffic management systems as a strategy to deal with traffic congestion. The main objective of TF is the prediction of near-future traffic measures based on current or past traffic data [5], such as can be seen in Figure 2.1. TF allows travellers to plan their trips ahead of time and assists transportation managers to enhance the management of traffic flows.

Figure 2.1: Traffic Forecasting's main idea



In early TF research, most studies were focused on predicting traffic at a single location using simulation models [14], theory-based traffic models [6] and classical statistical methods [7]. However, the emergence of sensing and telecommuni-

cations technologies integrated into transportation infrastructure started to generate vast volumes of traffic data that is hardly manageable by classic traffic models. The latter caused a switch in the modelling paradigm towards a data-driven approach [12]. Since then, a variety of methods have been proposed placing special emphasis on NNs [28, 29], RF [30, 31], kNN [32], ARIMA models [33], Fuzzy logic [10, 34] and Bio-inspired algorithms [8, 35], among others [5].

In recent years ML methods have attracted the interest of the transportation community, and they are present in a vast proportion of literature [36]. This data-driven approach consists of training a model that learns and approximates a mapping function using historical traffic, which then is used to make predictions when the model deals with new and unseen data (more details of this process can be found in Section 2.3). The main strengths of ML with respect to simulation approaches and traffic models are its ability to predict traffic without the need for expert traffic knowledge [16]. Besides, concerning statistical methods, ML is a more robust paradigm able to analyse complex and nonlinear relationships underlying traffic data [5].

As computational capacities and massive data processing techniques have increased, more complex scenarios with different road settings can be tackled with ML (e.g., network-wide predictions) leaving behind traditional approaches to address traffic prediction [9]. In this context, ML can contribute actively to the design and development of current Advanced Traffic Management Systems and Advanced Traveller Information Systems [37], such as it was mentioned in Section 2.1, to predict different traffic conditions in freeway and urban environments.

2.3 Machine Learning

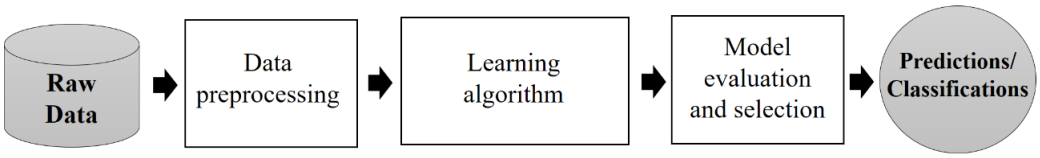
ML is the field focused on algorithms able of modifying and adapting their behaviour employing an iterative learning process without the need of being explicitly programmed. More formally, according to Mitchell [38], a ML algorithm learns from experience E related to a task T and its performance is evaluated by a metric P. Its performance at T improves according to P after experience E.

Applications like e-mail spam and malware filtering, automatic speech recognition, predictive maintenance in the industry are built upon ML. Within these appli-

cation areas, the most used ML approach is supervised learning (see Section 2.3.2), and conventional methods considered in the state-of-the-art are J48, Naïve Bayes, NNs, SVM, kNN, among others [39, 40].

As a common factor, all these applications do not include only ML algorithms but also data preprocessing techniques, as it has proven key to gleaning quality data before applying ML methods [18]. The connection between data preprocessing techniques and a ML algorithm generates a pipeline whose main structure is presented in Figure 2.2.

Figure 2.2: Standard structure of a machine learning pipeline



According to [41], a ML pipeline P can be defined as a combination of algorithms A that transforms input data X into target values Y. Let A be defined as

$$A = \{A_{preprocessing} \cup A_{feature} \cup A_{algorithm}\} \tag{2.1}$$

wherein $A_{preprocessing}$ is a subset of preprocessing techniques, $A_{feature}$ a subset of feature engineering methods, and $A_{algorithm}$ a ML algorithm with configuration of hyperparameters $\lambda^i \in \Lambda$.

Having introduced the formal definition of a ML pipeline, the following sections are devoted to presenting relevant concepts associated with the elements that constitute a pipeline.

2.3.1 Data Preprocessing

Input data must be provided in the format that suits ML algorithms. Unfortunately, real-world databases are highly influenced by the presence of noise, missing values, inconsistent data, among others [42]. Therefore low-quality input data can considerably affect the performance of ML methods [18]. In this section, we describe a general overview of data preprocessing techniques that improve the quality of data

before fed it into ML algorithms. For more details, the interested reader can consult [17].

Data Preparation

Data preparation is usually a mandatory step in supervised learning problems [42]. It converts prior raw and, sometimes useless, data into new data that fits the input of ML methods. If data is not prepared correctly, ML methods will not operate, will report errors during their runtimes, or will generate results that do not make sense within the context wherein the data comes. We present below representative approaches within the data preparation phase [17].

- **Data Cleaning:** This approach includes operations related to inconsistent data corrections and reduction of redundant data. The primary purpose is the detection of discrepancies and dirty data, which means identifying fragments of the original data that do not make sense in the context under study [43, 44].

- **Data Transformation and Data Integration:** In the data transformation process, data is converted to enable that the supervised learning process can be more efficient. Examples of possible paths to follow are feature generation, feature aggregation or data normalisation, among others [42]. For the vase of data integration, this preprocessing approach involves the merging of data that comes from multiple data sources. This process requires caution to avoid redundancies and inconsistencies in the resulting dataset [45, 46].

- **Data Normalisation:** Input data can have multiple variables with different measurement scales. Such diversity of measurement units can affect the data analysis. Therefore all the variables should be expressed in the same measurement units and should use a standard scale or range. This process gives all variables equal or similar weight and is particularly useful in statistical learning methods [42, 47, 48].

- **Missing Data Imputation and Noise Identification:** Here the objective is to fill in the variables of the input data that contain missing values following a particular strategy (e.g., imputation of the mean value of the most recurrent

values). In most of the cases, adding an estimation of the missing data is quite better than leaving blank. Complementary this approach includes smoothing processes whose purpose is to detect random errors or variances in the input data [49, 50].

Data Reduction

Data reduction comprises the set of techniques that obtain a reduced representation of the original input data [42]. Unlike data preparation, data reduction typically maintains the essential structure and integrity of the original data. Still, the amount of data is reduced to enhance efficiency in the learning process of ML algorithms. We present below representative approaches within the data preparation phase.

- **Feature Selection and Instance Selection:** On the one hand, Feature Selection is focused on the reduction of the dataset size by removing irrelevant or redundant features. The objective is to find a minimum set of attributes that facilitates the understanding of the model's outcome and increases the speed of the learning process. On the other hand, Instance Selection is based on choosing a subset of the total available data to achieve the same performance of the ML model as if the complete dataset were considered. This approach seeks to select the best and most representative instances from the original data for not having to use the full set of data [51, 52].

- **Discretisation:** This process transforms quantitative data into qualitative data, that is, numerical attributes into discrete or nominal attributes with a finite number of intervals [42]. Once the discretisation is carried out, the data can be treated as nominal by ML method [53, 54].

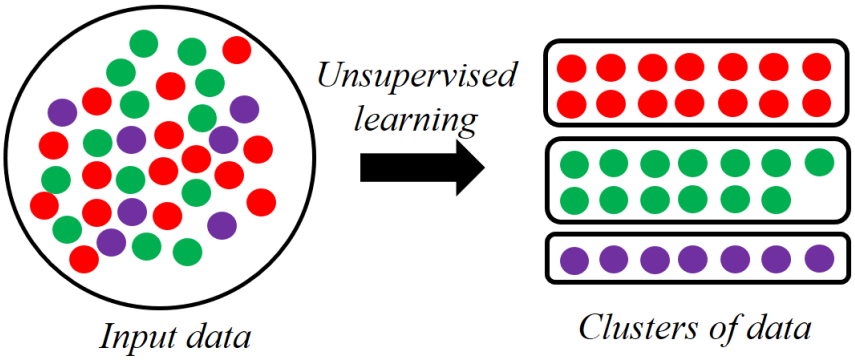
2.3.2 Supervised and Unsupervised Learning

Once data preprocessing is completed, the next step in the Knowledge Discovery in Databases cycle [42] is selecting a ML algorithm in charge of extracting knowledge and valid patterns previously unseen in input data. ML algorithms can be subdivided into multiple areas [55], among which the best known are supervised

learning and unsupervised learning. These two approaches are presented with more details below.

Unsupervised learning is one of the significant areas in ML. Unlike supervised learning that typically uses labelled data, that is, during the training process of a model, the target values are clearly defined in Y; unsupervised learning looks for patterns in data with no pre-existing labels [56]. The central approach of unsupervised learning is usually focused on clustering grouping of data points. Given a set of data points, clustering is in charge of organising X data points into specific groups such as is shown in Figure2.3. Data points that are in the same group should have similar properties, while data points in different groups should have highly different properties and/or features. It is important to note that these potential groups are not previously defined in the input data and is the purpose of unsupervised learning algorithms to discover them. Representative applications of unsupervised learning are marketing segmentation and anomaly detection, among others [57].

Figure 2.3: Unsupervised learning: clustering

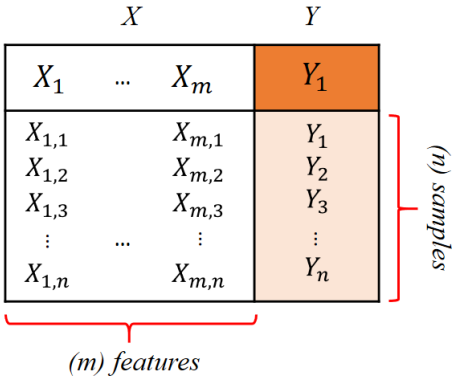


As it was stated above, supervised learning is the other fundamental area of ML [58]. It basically consists in algorithms that learn a function ($f : X \rightarrow Y$) by training with a finite number of input-output pairs, being X the input domain and Y the output codomain. This learning stage can be seen as *E* in Mitchell's definition [38], and the specific task *T* may vary, but usually involves predicting an output given a new and unseen input [55].

Supervised learning problems can be processed by learning from a training dataset composed of instances that take the form (x, y). In this format, $x \in X$ is a

vector of values in the space of input variables (features) and $y \in Y$ is a value in the target variable such as is shown in Figure 2.4. Inputs will usually belong to a subset of $\mathbb{R}^n$ and outputs take values in a specific one-dimensional set, either finite or continuous. Once trained, the obtained model can be used to predict the target variable on unseen instances [55].

Figure 2.4: Format of a machine learning dataset



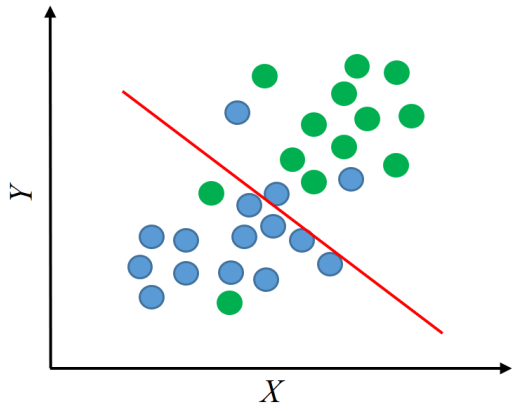
Supervised learning problems can be usually divided into two categories: classification and regression [59, 60]. In both cases, the basis is an input dataset, X, and their difference is the type of target variable, Y, to be predicted. On the classification case, Y is divided into discrete categories, while in regression, the purpose is predicting continuous values.

Standard classification problems [1] can be either binary or multi-class problems [61, 62]. In the former case, an instance can only be associated with one of two values: positive or negative that is equivalent to 0 or 1, such as can be seen in Figure 2.5. Examples of this binary classification are email messages that can be categorised into spam or non-spam. Regarding multi-class problems, they involve cases wherein there are more than two classes under consideration. That is, any given instance will belong to one of the multiple possible categories. For example, a flower image can be categorised within a wide range of plant species.

Diversely, a supervised regression problem [63, 64] consists in finding a function which is able to predict, for a given an example, a real value among a continuous range. The latter is usually an interval in the set of real numbers $\mathbb{R}$. For

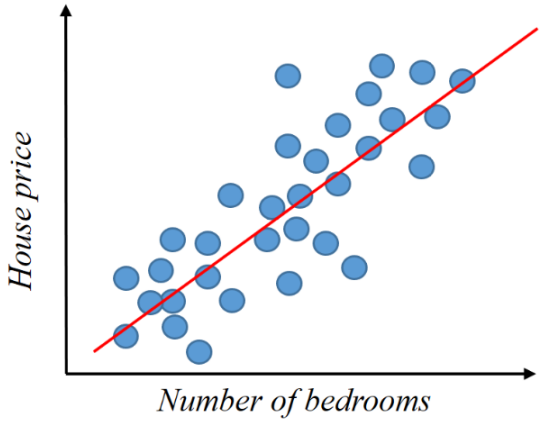
[1] The reader can consult further details of non-standard classification problems in [55]

Figure 2.5: Supervised binary classification.



example, the price of a house may be calculated using multiple characteristics such as the number of bedrooms as can be observed in Figure 2.6.

Figure 2.6: Supervised regression.



In this research, we focus on the prediction of traffic modelled as a supervised learning problem using regression and classification approaches. Although TF can be addressed with different modelling paradigms (e.g., time-series, unsupervised learning), currently, supervised learning is the data-driven approach most used in transportation literature [9, 36].

2.3.3 Ensemble learning

The ML pipeline presented in Figure 2.2 has a crucial component wherein a ML algorithm is chosen for making predictions or classification. However, it is well-known that there is no single ML method that achieves high performance on every possible learning task [65]. The performance of a single-learner can changes drastically across very diverse data-driven problems. Therefore, one state-of-the-art approach to overcome such a challenge is to combine multiple ML methods and either aggregate or combine its outcomes. This approach is known as ensemble learning, and its purpose is to enhance the performance of single methods by combining various classifiers that can be more adaptable to diverse learning tasks and outperform the performance of individual algorithms [66].

Ensemble learning paradigm follows the natural human behaviour that tends to seek several opinions before making any relevant decision. The primary motivation for the combination of classifiers in ensembles is to improve their generalisation ability: each classifier is known to make errors, but since they are different, misclassified examples are not necessarily the same [67]. Ensemble methods are well-established in ML there are two main taxonomies to differentiate the most common approaches.

First, there is a family of ensemble methods based on the collection of classifiers that are minors variants of the same classifier [68]. Although the group of classifiers comes from the same learner, keeping diversity between them is a crucial factor in making accurate ensemble methods. To accomplish such a criterion of variety, there are different approaches focused on building diverse classifiers trained with varying sub-sets of training, which belong to the complete training set at hand. AdaBoost [69] and Bagging [70] are common strategies to carry out the ensemble construction process while guaranteeing the diversity among classifiers.

Secondly, there exists an alternative ensemble approach that combines diverse classifiers that use different training paradigms (e.g., linear or non-linear algorithms). This ensemble strategy is known as ”multiple classifier systems” [71], and its purpose is aggregating the predictions of the classifiers when unknown instances are presented. Multiple-classifiers are used to achieve the best possible classification performance; however, this type of ensembles will work only when

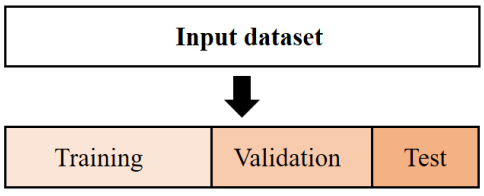
it is possible to build individual classifiers which are more than 50% accurate [71] and independent of each other [72]. If all the classifiers in the ensemble were identical, the multi-classifiers system would not obtain high performance as is expected.
.

2.3.4 Model Selection and Assessment

In its most basic definition, the Model Selection Problem (MSP) is one of the fundamental tasks of scientific inquiry. Determining the mechanism that explains data observations is usually related to a mathematical model that can predict those observations [73]. Therefore, MSP is the task of selecting a statistical model from a set of candidate models given input data. In ML the MSP is the process of choosing one final ML model from a set of candidate models. This task implies estimating the performance of the different models to choose the best one to address the problem at hand.

The best approach to model selection requires enough data that sometimes could not be the case due to the complexity of the problem under study. If we are in a data-rich situation, the best way to proceed is to split the input dataset into three parts randomly [74]: a training set, a validation set, and a test set such as is introduced in Figure 2.7. The training set is used to fit the set of available models; the validation set is then used to estimate prediction error for model selection; and finally, the test set is used to assess the generalisation error of the final chosen model. Then the best model is selected based on the validation error, and the test set should be kept in a "vault" to be brought out only at the end of the process when the best model has been selected [75]. A typical data split maybe 50 % training, and 25% validation and 25% test.

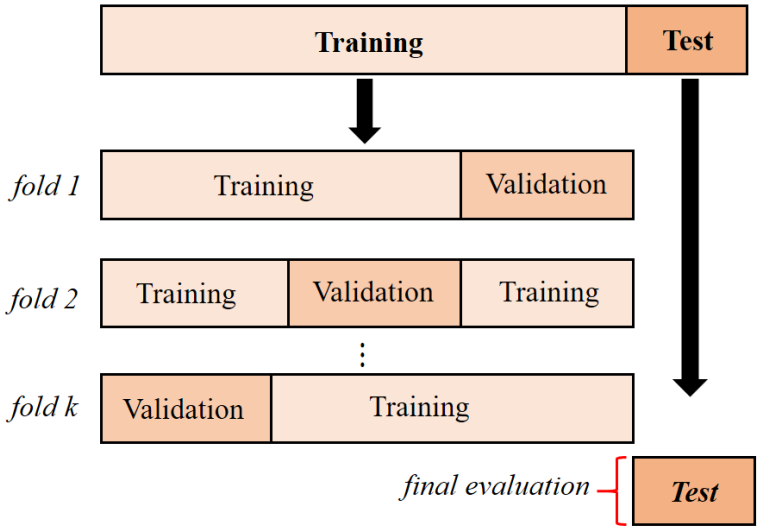
Figure 2.7: Training, validation and test data partitions for model selection



However, the approach mentioned above could be impractical on ML supervised problems wherein there is not sufficient data. In these cases, the most common approach is using re-sampling strategies to carry out the model selection [1]. Cross-validation is the re-sampling strategy most commonly used in situations where there is no enough data [76]. In this approach presented in Figure 2.8, the training set is split into *k* smaller sub-sets and the next steps are followed for each of the *k-folds*:

- a ML method is trained using *k-1* of the folds as training data.
- the resulting trained model is validated on the remaining part of the data (i.e. it is used as a test set to compute a predefined performance metric such as root mean squared error in regression problems or accuracy on binary problems). Usually, this split of the data is used for tuning hyperparameters.

Figure 2.8: Cross validation to approach the model selection problem



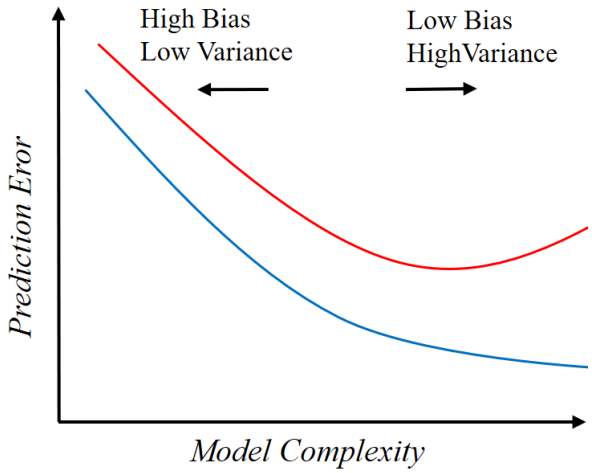
The final performance metric is the average of the metric reported by every *k-fold*. This approach can be computationally expensive, but it does not waste too much data, which is a significant advantage in some supervised learning problems [74].

[1] Other model selection approaches can be consulted with more details in [11]

Having defined the main strategies for model selection, now, how do we measure the actual performance of the chosen model and corroborate that we made the right selection? After selecting the final model, estimating its prediction error (generalisation error) on new data enable us to answer this question; this is known as model assessment [11]. The generalisation performance of a ML method relates to its prediction capability on independent test data. Considering this performance is important because it guides the final choice of a particular learning method and gives a measure of the quality of the chosen model [77].

The generalisation error for a ML algorithm can be divided into Bias error and Variance error. The former is related to assumptions made by a ML algorithm to make the target function easier to learn. Linear algorithms usually have a high bias making them fast to learn and to understand but generally less flexible. Therefore, they have lower predictive performance in complex problems that fail to meet the simplifying assumptions of the algorithms bias [78]. On the other hand, the Variance error is related to how the final estimations can change if different training data was used. ML methods are influenced by the specific characteristics of the training data. This means that the training data affects the parameters used to characterise the mapping function of the learning algorithm [79].

Figure 2.9: Bias and Variance of a machine learning method



According to Figure 2.9 if a ML model is too simple and has very few parameters (model complexity), then it has high bias and low variance. On the other hand,

if the model has a large number of parameters, then it will have high variance and small bias. In this context, the purpose of model selection and assessment is to find a balance between bias and variance with particular emphasis in achieving low bias and low variance. Thus, the ML algorithm under consideration will make good enough performance.

Finally, although the approaches presented above about model assessment and selection in ML brings guidelines to chose the most promising method over a set of candidates, this process is usually tedious and computationally expensive. This is generally done by ML experts who make use of their knowledge or by non-expert users who tackle the problem using a trial and error approach that causes the success of ML comes at a high-cost [80].

2.4 Automated Machine Learning

As it was described in Section 2.3.4, dealing with the MSP is a task that involves human effort, time and computational resources. In this context, AutoML is an emerging area that can address the MSP. AutoML aims at automatically finding the best combination of preprocessing techniques, ML algorithm and its hyperparameters, according to a particular performance measure on a given dataset without being specialised in the problem domain wherein this data comes from [19]. It allows us to reduce human bias and to improve computational costs by making the construction of ML applications more efficiently. More formally, the process consists of identifying the most promising combination of preprocessing techniques, ML method and hyperparameters, which optimises a given performance metric when the pipeline is trained on training data $D_{train}^{(i)}$ and evaluated on test data $D_{test}^{(i)}$.

Current literature [19, 41, 80] reports a variety of AutoML methods. They differ depending on which parts of a ML pipeline are automated; for instance, data preprocessing, algorithm selection and hyperparameters, or even the entire ML pipeline. In this section, we review key concepts and related work in AutoML, ranging from the isolated automation of data preprocessing to the automatisation of a complete pipeline. First, we introduce automated data preprocessing and feature engineering in Section 2.4.1. Then, we present methods for finding the best combi-

nation of algorithm and hyperparameters setting in Section 2.4.2. Finally, AutoML methods for creating a complete ML pipelines are introduced in Section 2.4.3.

2.4.1 Automated Data Preprocessing

Data preprocessing and feature engineering are key elements when building ML pipelines. They are known for being tasks related to the specific area of knowledge in which ML is applied. However, as the AutoML goal is to be domain agnostic, data preprocessing and feature engineering are not incorporated fully in current AutoML methods [41].

By elimination noise and removing redundant features, a ML pipeline can be trained faster with a lower generalisation error. Besides, the interpretability of the trained models can increase as the number of features considered is smaller. Basic methods for data preprocessing and feature selection (e.g., imputation of missing values, scaling of variables, univariate selection, variance threshold) are already integrated into modern AutoML frameworks [1, 21, 81]. They are usually applied in a sequential way to the input dataset before searching for the pair algorithm and hyperparameters setting. In this sense, the inclusion of preprocessing methods does not consider which of them are actually improving the quality of the data and what others are only adding computational costs without any significant improvement on the performance of ML models.

On the other hand, with regard to features generation, the conventional approach in AutoML is following an iterative process in which based on an initial dataset, a set of candidate features is generated and ranked. For instance, high ranking features are evaluated and potentially added to the dataset using a deep feature synthesis method [82]. In contrast, an alternative strategy is using meta-learning to predict in advance the influence of a set of promising features over the performance of ML models [83, 84]. Nevertheless, approaches to enhance automated feature generation with domain-knowledge are still not considered within AutoML [85, 86].

In summary, data preprocessing and feature engineering techniques included in AutoML are domain-agnostic [41]. Therefore, this approach cannot identify whether preprocessing contributes or not to a specific domain wherein AutoML

is applied. Incorporating domain knowledge to the preprocessing methods could increase the quality of the data drastically [87–90], and enhance the performance of AutoML.

2.4.2 Automated Algorithm Selection

Most of the current advances in AutoML are focused on finding the best combination of algorithm and its hyperparameters setting. This problem is commonly solved using only a optimisation approach, and some of the most representative solutions are described below (to read further about state-of-the-art methods in this AutoML area, please consult [19, 41]).

The first AutoML method for automatic algorithm selection was proposed in 2011 Hutter et al. [91] and Bergstra et al. [92]. They developed a sequential model-based optimisation method to automatically find the most suitable ML algorithm with its best hyperparameters setting. This approach initialises the search of the aforementioned pair generating multiple candidate configurations from a predefined search space. They are subsequently tested, and in each iteration (using Bayesian optimisation) the candidate configuration with the best performance is selected. The procedure is stopped when a fixed time budget is exceeded, or all generated configurations are tested.

One year later, Snoek et al. [93] introduced a method named SPEARMINT that uses surrogate models to propose promising algorithms and hyperparameters settings. A significant limitation of SPEARMINT is that it does not support the inclusion of categorical hyperparameters, which in consequence reduces its scope of application. Then in 2014, Claese et al. [94] developed OPTUNITY that works in a similar way to the method proposed by Hutter et al. [91] and Bergstra et al. [92]. Concretely, OPTUNITY limits the number of total objective function evaluations, using a heuristic approach, to figure out the best pair of algorithm and hyperparameters. Furthermore, categorical hyperparameters are transformed into an integer to be treated as continuous values without having to discard them.

More recently, Falkner et al. [93] proposed the BOHB method to solve the Combined Algorithm Selection and Hyperparameter optimisation (CASH) problem. Specifically, it uses a combination of bayesian optimisation and hyperband

to find the best pair of ML algorithm and hyperparameters with less computational costs. For each candidate pair, BOHB transfers the current budget and the evaluated configuration to the objective function; thus, it is possible to keep traceability of what combinations, from the ones that have been considered, are suitable candidates in terms of performance and computational cost. Finally, in this same year, Gustafson [94] developed a method named BTB. For the hyperparameters tuning, a grid with all of the possible combinations of hyperparameters is created. Each point (called hyper-partition) in this grid is treated as a bandit with unknown reward. To propose a new candidate configuration, a hyper partition is selected via multi-armed bandit learning to be evaluated with an algorithm. This last step is repeated for a fixed number of iterations until finding the best combination of algorithm and hyperparameters.

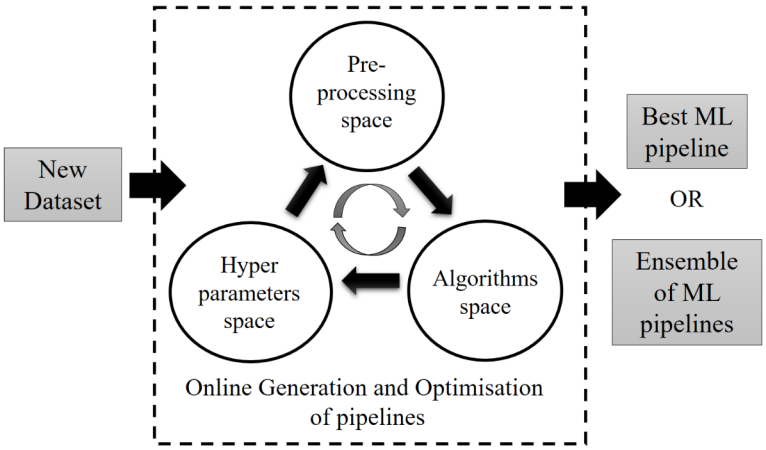
Until now, the sections introduced above have presented representative strategies to automatise the selection of data preprocessing techniques or ML algorithms. Diversity, the next section introduces the main AutoML methods able to automatise at the same time, all the components of the ML workflow.

2.4.3 Automated Machine Learning Pipelines

2.4.3.1 Approaches for the automatic construction of ML workflows

In the literature, we can usually find two main categories of AutoML methods to automatise the construction of ML pipelines. Those categories are based only on optimisation or on meta-learning integrated with optimisation. The first one uses optimisation only to generate, tune, and assess pipelines, as represented in Figure 2.10. In this case, when new data is fed into this type of methods, they start to build multiple possible combinations of pipelines from a predefined search space of preprocessing, ML methods and hyperparameters. Then, after a given time or a fixed number of iterations, they return the best pipeline found or construct an ensemble from a set of competitive pipelines identified during the optimisation. From this perspective, the ML pipeline building problem consists of finding pipeline structures that minimise a cross-validation loss function (Equation 2.2). Representative AutoML methods for this category can be Auto-WEKA [20], TPOT [21], H2O [24], among others [95, 96].

Figure 2.10: General approach of AutoML methods based on a pure optimisation search strategy



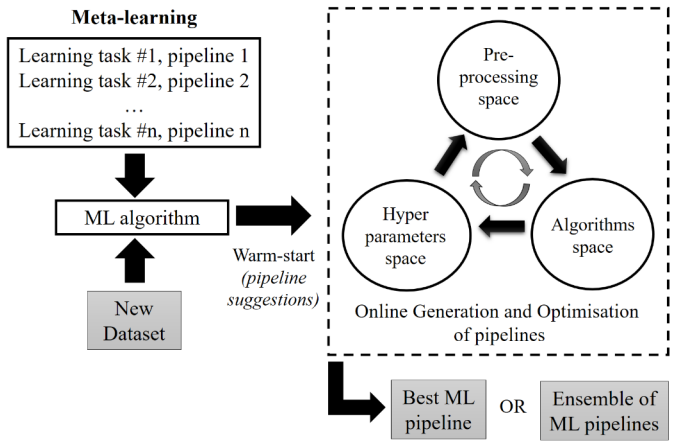
Equation 2.2 shows the loss function to find the best $P_{A,\lambda}$ wherein A_λ is a combination of preprocessing techniques, feature engineering methods, and a ML algorithm with a configuration of hyperparameters $\lambda^i \in \Lambda$, which is trained on training data and evaluated on test data-

$$P_{A^*,\lambda^*} = \underset{A^{(i)} \in A, \lambda^{(i)} \in \Lambda}{argmin} \frac{1}{K} \sum_{i=1}^{k} \gamma \left(P_{A^{(i)},\lambda^{(i)}}, D_{train}, D_{test} \right) \tag{2.2}$$

As shown in Equation 2.2, this search process can be considered as a black-box optimisation problem that it is not easily solvable as the search space can be large and complex. This equation is usually non-smooth and derivative-free, and the convergence speed is a critical problem for building ML pipelines [41]. Some methods to solve this equation are grid search [24], genetic programming [21], and Bayesian optimisation [91].

Regarding AutoML methods that combines optimisation with meta-learning [1, 97, 98], their working process is shown in Figure 2.11. For this type of AutoML methods, in an offline phase, optimisation is used to find ML pipelines with high performance on a repository of datasets. Simultaneously, a group of meta-features are extracted to characterise each dataset that describes its representative characteristics such as data skewness, the entropy of targets, number of instances, variance of the data, among others [98]. Afterwards, the meta-features and the

Figure 2.11: General approach of AutoML methods with a search strategy of pipelines based on meta-learning and optimisation



optimised pipelines are stored in a meta-knowledge base wherein each instance contains the set of meta-features describing every dataset and its corresponding optimised pipeline. Then, in the online stage when new input data comes in, the meta-learning component recommends pipelines that are likely to perform well on the input dataset. This selection of pipelines is then used for a warm-start of the optimisation process, which can end suggesting a single pipeline of an ensemble that contains competitive pipelines.

In a nutshell, there are consolidated AutoML methods that automatise the complete ML workflow. As a common factor, the core of their pipeline search strategies is focused on generating and fine-tuning individual pipelines that later can be or not integrated on ensembles. However, this online optimisation of pipelines is computationally expensive because the complexity of the search space (diverse data preprocessing and ML methods) and the high evaluation cost of the objective function in big datasets. Although AutoML approaches that combine meta-learning with optimisation could potentially reduce the impact of these issues, more research is needed to corroborate these benefits. The reason for the latter is because it is difficult to characterise a wide variety of learning tasks such as TF; therefore, the assumption of always being able to suggest competitive pipelines for a new and unseen learning task needs more in-depth research.

2.4.3.2 **Pioneer methods based on optimisation and meta-learning with optimisation**

In this section, we present an overview of Auto-WEKA and Auto-sklearn. They are the pioneer AutoML methods of the pipeline search strategies introduced above. Besides, these two methods receive special attention in this research, as will be seen later in Chapters 4 and 5.

1. **Auto-WEKA:** Auto-WEKA is the pioneer of AutoML methods purely based on optimisation. Originally published in 2013, it approaches the algorithm selection problem through a Bayesian optimisation method. It considers the space of WEKA's ML algorithms $\boldsymbol{X} = \left\{X^{(1)}, ..., X^{(k)}\right\}$ and their hyperparameter spaces $\boldsymbol{A} = \left\{A^{(1)}, ..., A^{(k)}\right\}$ to identify the combination of algorithm $X^{(i)} \in X$ and hyperparameters $A^{(i)} \in A$, which minimises a cross-validation loss (Equation 1), where $\gamma\left(X_A^{(i)}, D_{train}^{(i)}, D_{test}^{(i)}\right)$ denotes the loss achieved by algorithm $X^{(i)}$ with hyperparameters $A^{(i)}$ when trained on training data-set $D_{train}^{(i)}$ and evaluated on test data-set $D_{test}^{(i)}$.

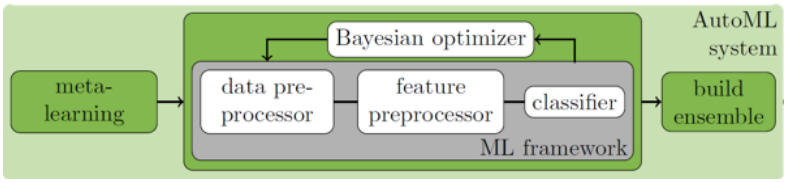
$$X_A^* * = \underset{X^{(i)} \in X, A^{(i)} \in A}{argmin} \frac{1}{K} \sum_{i=1}^{k} \gamma\left(X_A^{(i)}, D_{train}^{(i)}, D_{test}^{(i)}\right) \tag{2.3}$$

   Thornton et al. [20] call this the combined algorithm selection and hyperparameter optimisation (CASH) problem: determining $argmin_{\theta \in \Theta} f\left(\Theta\right)$ wherein each configuration $\theta \in \Theta$ contains the choice of algorithm $X^{(i)} \in X$ and its hyperparameters setting $A^{(i)} \in A$. With this problem definition, the Bayesian optimisation fits a probabilistic model to capture the relationship between different hyperparameter configurations and their performance. It then uses this model to select the most promising hyperparameter setting, assesses it, updates the model with the result of configuration chose, and iterates until a predefined time budget is reached. As can be seen, this is a brief description of the method. The interested reader is referred into [20] for further details.

2. **Auto-sklearn:** Auto-sklearn is the pioneer of AutoML methods that use hybrid search strategies, this means, methods that complement the optimisation with meta-learning to boost the performance of the former. Concretely,

Auto-sklearn employs meta-learning, Bayesian optimisation and ensemble selection to find promising ML pipelines composed of data preprocessing methods and one ML classifier. Here we provide a brief description of the method. The interested reader is referred to [1] for further details. Figure 2.12 summarises its overall workflow.

Figure 2.12: Auto-sklearn's workflow composed of meta-learning, bayesian optimisation and ensemble learning [1]



In an offline phase, for a repository of 121 datasets, Bayesian optimisation is used to determine an optimised ML pipeline with high performance on every dataset. These pipelines are generated from a search space of 15 classifiers, 14 feature preprocessing methods, and 4 data preprocessing techniques. Then, for each dataset, a set of 38 meta-features is extracted to characterise every set of data. Those meta-features include information-theoretic and statistical information such as statistics about the number of data points, features, the number of classes, data skewness, the entropy of the targets, among others. Later on, instead of storing the 121 datasets, their meta-features and the ML pipelines are saved in a meta-knowledge base wherein each instance contains the set of meta-features describing every dataset and the optimised pipeline that works well on it.

In the online phase, that is, when a new dataset D_{new} is given, Auto-sklearn computes its meta-features, ranks all the datasets stored in the meta-knowledge base (stored in the form of meta-features and not the data itself) by their $L1$ distance with respect to D_{new}, and selects the stored ML pipelines for the k nearest datasets (by default $k = 25$). The assumption is that these selected pipelines are likely to perform quite well in D_{new} as they performed well on datasets with similar meta-features (pipelines closer to the first position of the ranking would expect higher performance on D_{new}). This selection

of K most promising pipelines is used then to seed the Bayesian optimisation component as a warm-start approach, which boosts the performance of the optimisation. In addition to the recommendations done by the meta-learning component, the Bayesian optimisation process (under a time budget constraint) generates and tests new pipeline structures from the same search space mentioned above. In the final step of Auto-sklearn's workflow, the best pipelines identified during the Bayesian search process are used to construct an ensemble. This automated ensemble construction avoids to commit itself to a single hyperparameter setting, and it is more robust than only using the best pipeline found with the optimisation component.

2.5 Machine Learning in Traffic Forecasting

Having introduced the background of ML and AutoML, in this last section, we review the advances of these two learning paradigms in TF, specifically, from a supervised learning approach.

2.5.1 Supervised Learning Modelling approaches

From a supervised learning perspective, TF is approached by building a model using historical traffic data to make predictions on new and unseen data. Depending on the type of input and output (predicted) data, different ML modelling approaches can be used, such as is presented below.

When the objective is to predict a continuous traffic variable (e.g., speed or flow), the possible modelling approaches can be supervised regression or clustering-pattern recognition. In the first case, the focus is on using ML algorithms to learn a functional form based on the input data, without prior models or data distribution assumptions [36]. Diversely, when the modelling choice is clustering-pattern recognition, the focus consists of finding the relationships of different locations by characterising similar traffic measure values from one road to another. Then the traffic locations are grouped into clusters that divide the road network into correlated groups. Once the clusters have been identified, the next step is to use a

supervised regression approach to predict the traffic conditions, cluster by cluster, based on historical traffic data belonging to each group.

When the objective is forecasting a discrete traffic measure, the modelling approach should be supervised classification that also learns a mapping function based on historical data. For instance, ML methods can forecast the LoS of a specific road location. The latter is a categorical variable that measures the quality of the traffic through letters from A to E in a gradual way, category A being moderate traffic and category E extended delays [99]. It is essential to clarify that the forecasting of discrete variables could also be addressed as a supervised regression problem on some occasions, predicting either speed or density (continuous values), and then discretising these predictions to obtain the categorical outputs.

2.5.2 Machine Learning methods

Currently, there is a broad set of ML algorithms that have been used to approach the prediction of traffic from a supervised learning perspective. In this section, we provide an overview of ML methods used to make traffic predictions.

In [13], Vlahogianni et al. presented and characterised ML methods applied to TF in terms of the hypotheses made about the statistical distribution followed by the data, the quantity and quality of the data needed for making predictions, and the accuracy of the methods. Similarly to the ML field [74], the authors categorised two types of methods: parametric and non-parametric ones. The parametric category assumes the relationship between the explanatory and response variables as known. Representative ML methods within this category are logistic regression, linear discriminant analysis and perceptron, among others [74]. Contrary, non-parametric methods can model non-linear relationships without requiring the mentioned assumptions. Commonly non-parametric algorithms are NNs, SVMs, kNN and RF, among others [74].

More recently, Ergamun and Levinson [36] updated the classification of methods introduced previously by [13]. The authors proposed a new taxonomy wherein parametric and non-parametric ML methods were compared against statistical techniques. The authors stated that ML methods had attracted more attention in recent years outperforming statistical methods such as historical average and exponential

smoothing. Despite this, the authors stressed that both approaches differ on purposes and model development process and, therefore, their use depends on the particular TF problem at hand. Statistical methods concern inference and estimation providing a model that offers insights on the data, considering both data distributions and model restrictions; while ML methods are focused on providing efficient and accurate predictions without prior models or data distribution specifications.

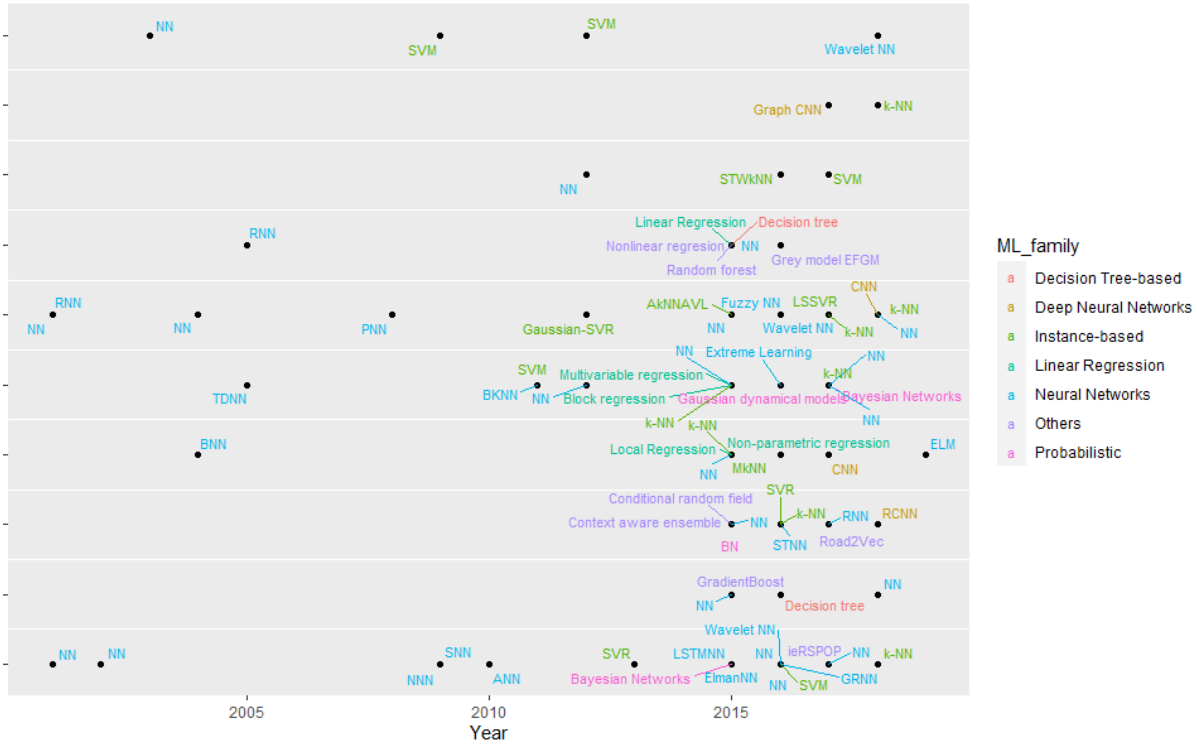
Considering the main categories of methods presented above, we made a literature review between 2000 and 2019 to extract representative papers in the area of TF wherein ML methods have been used. As a result, we identified six main families of ML methods; each family groups methods with common features at the moment of approaching TF problems. The families are Decision tree-based, Deep NNs, Instance-based, Linear regression, NNs, and Probabilistic.

Figure 2.13 maps how different types of ML methods have been used to approach TF over the last decades. As can be seen, NNs were the dominant ML methods to approach the prediction of traffic between 2000 and 2010. From then to now, Instance-based methods such as k-Nearest Neighbours and Support Vector Machines have been introduced to the transportation literature. Within this time, these methods were used to solve not too complex TF problems wherein the purpose was to make predictions at particular locations on a freeway or urban environments.

More recently, with the appearance of sensing technologies able to measure traffic in a more realistic way (e.g., GPS), traditional ML methods have presented difficulties dealing with these data. In this context, deep NNs have been introduced to the transportation literature to approach more complex transportation scenarios and to enable making predictions at the network level. As can be observed in Figure 2.13, this change has been the trend in the transportation literature during the last five years.

The remaining families of ML methods (decision tree-based, linear regression, probabilistic) have been broadly applied to handle more straightforward problems wherein it is not needed to make traffic predictions at the network level. In such transportation contexts, deep NNs could also be applied; however, the implementation of classical ML methods guarantees small computational costs, while obtaining fair accuracy in predictions.

Figure 2.13: ML methods most commonly used to approach TF during the last two decades



Summarising, the wide variety of ML methods has led to different taxonomy proposals that classify the methods based on the mathematical assumptions from which they operate [12–16]. Contrarily, a few efforts have been directed to categorise the TF problems and how they can be modelled to be approached by those methods. Therefore, without knowing what the types of TF problems are, it is difficult to characterise the performance of ML methods in each particular problem. TF problems should be classified by characteristics of the transportation scenario (e.g., traffic data source, the context of predictions) and the modelling specifications imposed by the supervised learning paradigm. However, systematically categorise and identify the types of TF problems remains unexplored to date.

2.5.3 Model Selection Problem in Traffic Forecasting

Despite the great variety of ML methods, dealing with the MSP in TF is not a trivial task because there are no clear guidelines to decide what methods are better to use depending on the TF problem at hand. The general approach to tackle the MSP in TF consists of testing a set of algorithms with multiple hyperparameter combinations and select the best one. Besides, if the method is going to be used in a real scenario, it requires to be complemented with data preprocessing techniques. The latter increases the complexity of the problem because apart of deciding the bet combination of method-hyperparameters, it is needed to select the most suitable data preprocessing technique from a set of multiple options. Thus, the transportation user faces various questions at the moment of building a ML pipeline able to deal with the TF problem at hand:

- What is the best ML method for my TF problem at hand?
- What is the best set of hyperparameters that boosts the performance of my ML method?
- What is the most suitable data preprocessing technique for my traffic data?
- Do I need to include feature engineering techniques to make more accurate traffic predictions?
- What are the best hyperparameters for my data preprocessing and/or feature engineering techniques?

In research areas wherein expert ML knowledge is not always an affordable asset (e.g., TF), building a ML pipeline involves human effort and high computational capacities because there is no single pipeline that achieves high performance on every learning problem [80, 100]. This trial and error approach generates that the success of ML happens at an elevated cost [80].

In this context and as it was discussed in Section 2.4, AutoML appears as a promising opportunity to approach the issues mentioned above. However, although AutoML methods have approached the MSP with high performance in other research areas [41], to the best of our knowledge, only one study [23] out of this

research has used AutoML in TF. In this work [23], Vlahogianni proposed a meta-modelling technique that, based on surrogate modelling and a genetic algorithm with an island model, optimises both the algorithm selection and the hyperparameter setting. The AutoML task is performed from an algorithms base of three ML methods (NN, SVM and Radial Base Function) that forecast average speed in a time horizon of 5 minutes using a time series regression approach. Although Vlahogianni [23] shows promising results, it does not consider the complete automation of ML pipelines for TF and a lot of research is still needed to take advantage of AutoML in TF. There are open issues such as:

- What is the performance of AutoML in supervised classification TF problems?
- What are the benefits of AutoML in TF when using a broader base of ML algorithms?
- What does it work better for TF: AutoML purely based on optimisation or AutoML based on meta-learning and optimisation?
- How does AutoML behave dealing with either balanced or imbalanced traffic data?

Science may be described as the art
of systematic oversimplification.

Karl Popper

CHAPTER

3

A Taxonomy of Supervised Traffic Forecasting Problems

3.1 Introduction

With the advent of data and growing computational resources, the prediction of traffic using ML has become a dominant approach. From a ML paradigm, TF can be tackled through supervised [8, 101] or unsupervised [102–104] modelling perspectives. Within them, the supervised learning approach is, by far, the most widely used modelling paradigm in TF. Transportation literature reports a huge set of supervised ML methods (e.g. RF, SVMs, ensembles) [5, 9]. This variety of automated learning approaches has led to different taxonomy proposals that categorise ML algorithms based on the mathematical assumptions from which they operate (either parametric or non-parametric) [12–16]. However, The categorisation of TF problems and how they can be modelled to be approached by ML is an underexplored area. In this context, without a clear classification of TF problems, it is almost impossible to formulate guidelines that determine what the most suitable ML methods to deal with every TF problem are.

Supervised TF problems must be classified considering two main perspectives at the same time: first, the transportation scenario that imposes particularities to the

problems that may alter the performance of ML methods; and second, modelling specifications imposed by the supervised learning paradigm. Nevertheless, systematically organise, characterise and define the types of TF problems from those two perspectives is a gap in the current literature. Instead, we do find research articles, mainly survey and review papers [5, 9, 12–14, 16, 36], which introduce a set of criteria that allows for the categorisation of these problems based on the type of data source (e.g. loop detectors or GPS sensors), the context of predictions (e.g. freeway or urban environments) and the scale of predictions (e.g. single point, road segment, or network level). Although these studies represent an advance in knowledge, their categorisation attempts have the following drawbacks: 1) there is not a unified set of criteria: they vary from one paper to another; 2) the criteria shared by some of the previously mentioned papers do not have a standard definition; 3) most of the criteria proposed are related to traffic characteristics, and they do not take aspects related to how the problems are modelled from a supervised perspective (e.g. modelling attributes to categorise how raw traffic measures can be shaped to generate a ML dataset).

With these ideas in mind, in this chapter, we propose a taxonomy to categorise TF problems in terms of both ML modelling and traffic specifications. The proposed taxonomy does not aim to highlight all details associated with the problems to maintain its comprehensibility and its size. It is instead designed according to core characteristics that may alter the complexity and the modelling of TF problems. Thus, the taxonomy provides a panoramic view of the different TF supervised problems to provide a common framework that helps to display similarities and differences among the problems and that allows us to identify well-established approaches, gaps and current trends of ML in TF. Besides, the taxonomy unifies and consolidates traffic-related criteria available in the literature to characterise TF problems. It also introduces new criteria to categorise TF problems concerning how they can be modelled from a supervised learning perspective.

This chapter is organised as follows. Section 3.2 introduces the proposed taxonomy, which is built based on traffic and ML modelling specifications. Next, Section 3.3 categorises relevant TF literature, dated from 2000 to 2019, using the proposed taxonomy to check its robustness and its ability to discriminate TF problems. Afterwards, Section 3.4 presents a hierarchical clustering analysis of the

categorisation mentioned above to extract families of TF problems. Finally, Section 3.5 recaps the content and main conclusions of this chapter.

3.2 Proposed Taxonomy

This section presents the proposed taxonomy to categorise TF supervised problems. It is built according to traffic and modelling specifications as can be seen in Figure 3.1. These two classes of specifications have a hierarchical inner structure in which, from top to bottom, there are attributes of at most three levels. The attributes of the first and second level correspond to general characteristics that determine the types of features that the problems may include. At the same time, the bullet points are the final attributes that assign the particular features of every TF problem.

The set of traffic specifications contains three blocks of attributes that classify the transportation context of the problems: data source, scope, and problem definition. The second set of specifications categorises how to model the TF problems from a ML supervised perspective: the input and output modelling, the steps of predictions into the future, and the preprocessing approaches of the input data.

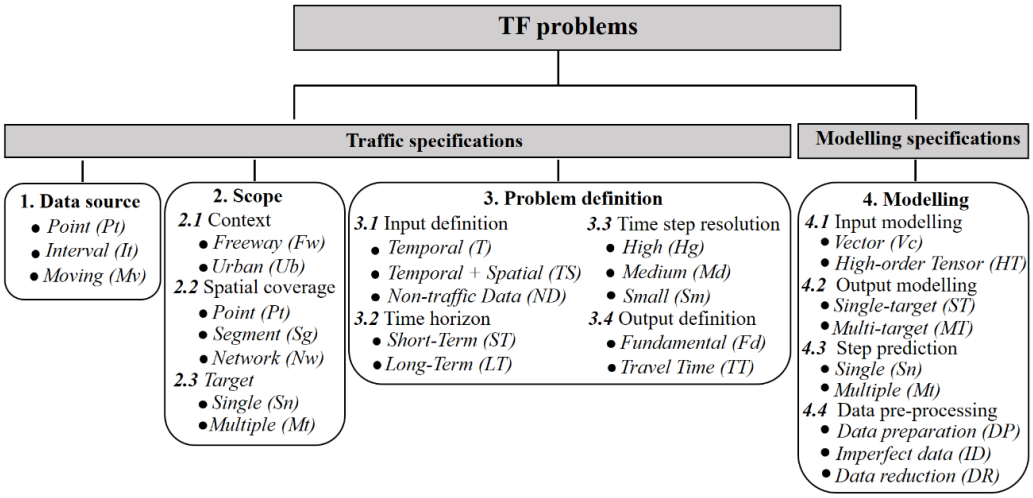
It is essential to clarify that the taxonomy does not aim to describe all the details associated with the problems to keep a moderate level of granularity. The remainder of this section is devoted to presenting and explaining the attributes of the taxonomy.

3.2.1 Traffic Specifications

Previous surveys have proposed traffic attributes to describe the transportation scenarios in which TF methods can be applied [5, 13, 14]. Those attributes are the data source used, the scope of predictions that includes context, spatial coverage and target attributes and, finally, the problem definition attributes. The latter incorporate input data used, the time horizon of predictions, the time step of the data and traffic measures to be predicted.

In this section, we present the aforementioned traffic-related attributes and their respective sub-attributes, including some crucial references that offer more detailed information about them. It is important to clarify that although these transportation

Figure 3.1: A taxonomy of traffic forecasting problems



attributes have been defined before in other studies, to the best of our knowledge, this is the first time that their definitions are unified and consolidated in a single taxonomy, and their influence over the preparation of the ML datasets has been approached.

3.2.1.1 Data Source

This attribute is related to sensing technologies and the traffic data that can be obtained by them. Recent progress in ITS has enabled the extraction of traffic data using different sources that can be classified in several ways [105]. Nevertheless, as we are proposing a generic taxonomy able to categorise the diversity of data sources used in TF, the categorisation presented here is divided into three groups: point detectors, interval detectors, and moving sensors. They differ in their spatial coverage capacity at the moment of sensing traffic. The three groups of data sources are described as follows.

Point

This type of data source is placed at specific locations on the roads to detect the presence of nearby vehicles. Point detectors output basic traffic measures as flow (number of passing vehicles per hour), occupancy (percentage of the time that the

detector is occupied), and density (number of vehicles per unit length of the road). The most conventional sensors within this category are loop detectors, microwave radars, laser radar sensor, active and passive infrared, among others (for more details see [2, 16, 105]).

Data obtained by a point sensor can be described as an ordered sequence of measurements m_p in a given position p (Equation 3.1), wherein $m_{p,t}$ is the value of the traffic measurement at time t and position p. In this case, the traffic measures and their predictions are only valid for describing the traffic conditions where the sensor is located.

$$m_p = \{m_{p,t}\} \ t = 1, 2, ..., T \tag{3.1}$$

The advantage of point sensors is they are reliable data sources, capturing all vehicles passing near them and collecting macroscopic traffic measures, which means averages of many cars. Their drawback is they cannot sense the paths of vehicles and, therefore, it is hard to find traffic relationships between different road segments. This issue can be overcome using spatial correlation analysis to find connections between the road segments with installed sensors, and then determining what specific sensors will be included in the dataset preparation phase [28, 106].

Interval

Interval detectors are capable of calculating traffic measures between two fixed points on the road. Unlike point detectors, they directly sense travel time. The most common technologies in this category are automatic toll collection systems, video cameras, and license plate recognition systems, among others (see [16, 105]).

Even though data coming from interval detectors can be represented using Equation 3.1, their spatial coverage is not the same as point sensors. In this case, data is valid for describing traffic measures between two points of a road segment, which means a broader spatial coverage beyond a single location.

In contrast to point sensors, interval detectors are not able to detect all the vehicles on the road positions wherein they are located [16]. Hence, having a sample of sensed cars that represents the actual traffic conditions on the roads is a challenge

[107]. Therefore, forecasting traffic with this type of data source requires large volumes of data.

Moving

The appearance of GPS in smartphones and vehicles has given rise to a new type of data source that gathers more detailed traffic information. Moving sensors provide individual traffic data related to vehicles' trajectories on the roads. This data allows for the identification of path patterns of cars in large areas with lower infrastructure costs, which means that it is more feasible to predict traffic at the network level [108, 109]. Nevertheless, aggregate traffic measures (e.g., flow or occupancy) can only be approximated to a certain point depending on the number of available moving sensors [2, 16]. This last condition affects data preprocessing because extrapolating the GPS samples to obtain estimations of aggregated traffic measures is needed [110], which may lead to a biased representation of traffic on roads.

GPS devices send location, direction and speed information every few seconds. This data can be represented as an ordered sequence of measurements such as shown in Equation 3.2. Every sample p_t can be defined as spatio-temporal data $p_t = (x_t, y_t)$ wherein the spatial component contains GPS coordinates, latitude (x_t) and longitude (y_t), and the temporal part includes the time stamp (t).

$$\overline{p} = \{p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_t \rightarrow ...\} \ t = 1, 2, ..., T \tag{3.2}$$

The sequence in Equation 3.2 sometimes contains non-exact records that impact the quality of the data [16, 111]. To overcome this issue and prepare ML datasets, the technique most commonly used is map-matching. The latter is a procedure that pins the drifting positions of data to the correct road links on which vehicles are travelling [112–114].

3.2.1.2 Scope

This attribute consists of the context wherein traffic predictions are made and their spatial extent. We identify three sub-levels within the scope attribute. The first sub-level corresponds to the transportation environment, wherein traffic predictions can

take place. Based on [13], the contexts can be urban or freeway. The second sub-level determines the spatial coverage of forecasts, which is divided into a point, segment, and network [14]. Finally, the third sub-level is related to the number of locations simultaneously considered to make predictions, which can be a single-target (one point, segment, or network) or multiple locations [9].

Context

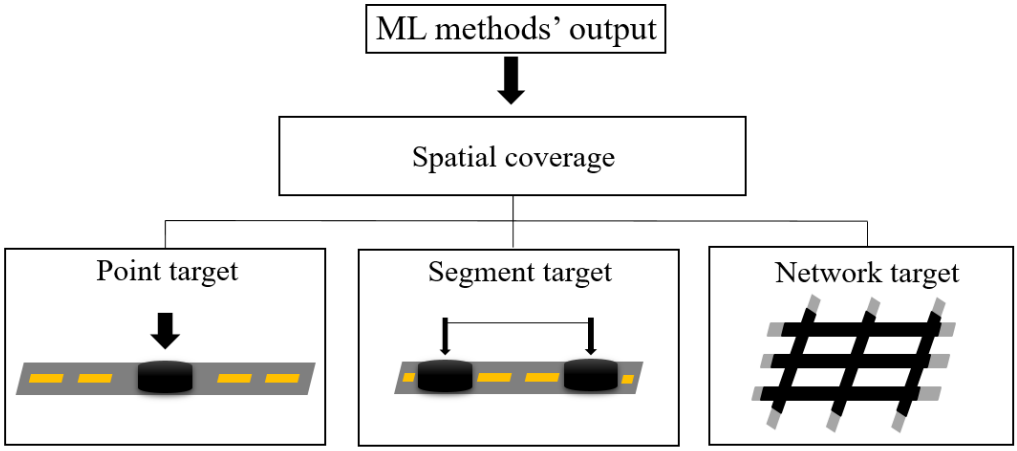
Regarding the context of prediction, every area of a traffic network has characteristics that determine the behaviour of traffic on it. We identify two main types of traffic contexts based on [9, 12, 13, 16]: Urban and Freeway. Most of the supervised ML models in the transportation literature are built using traffic data collected within freeway contexts wherein traffic is generally uninterrupted [115–118]. The main reason behind this trend is the availability of fixed position sensors already installed on freeways around the world; the latter makes the acquisition of data easier [16].

Until recently, research in urban contexts was not so common because of fixed sensor coverage issues [9, 13]. However, the advent of GPS has allowed more research in urban settings that have not yet been covered with fixed position detectors [2]. Regarding the development of predictive models, traffic in urban locations is more complex, and the modelling process has to take aspects such as the influence of traffic lights and intersections into account. This requires more elaborate ML models that consider that variability and provide reliable traffic predictions [9, 12].

Spatial Coverage

This attribute represents the spatial coverage of predictions provided by ML methods (Figure 3.2). As is defined in [14], the spatial covers are point, segment, and network. Until the most recent and comprehensive literature reviews [5, 12], most forecasting efforts were focused on point and segment predictions [28, 117–124]. However, as more moving sensor data becomes available and ML methods able to deal with temporal and spatial data appear, an increase of traffic forecasting at the network level is reported in the transportation literature [9, 29, 125, 126].

Figure 3.2: Spatial coverage of predictions provided by ML methods

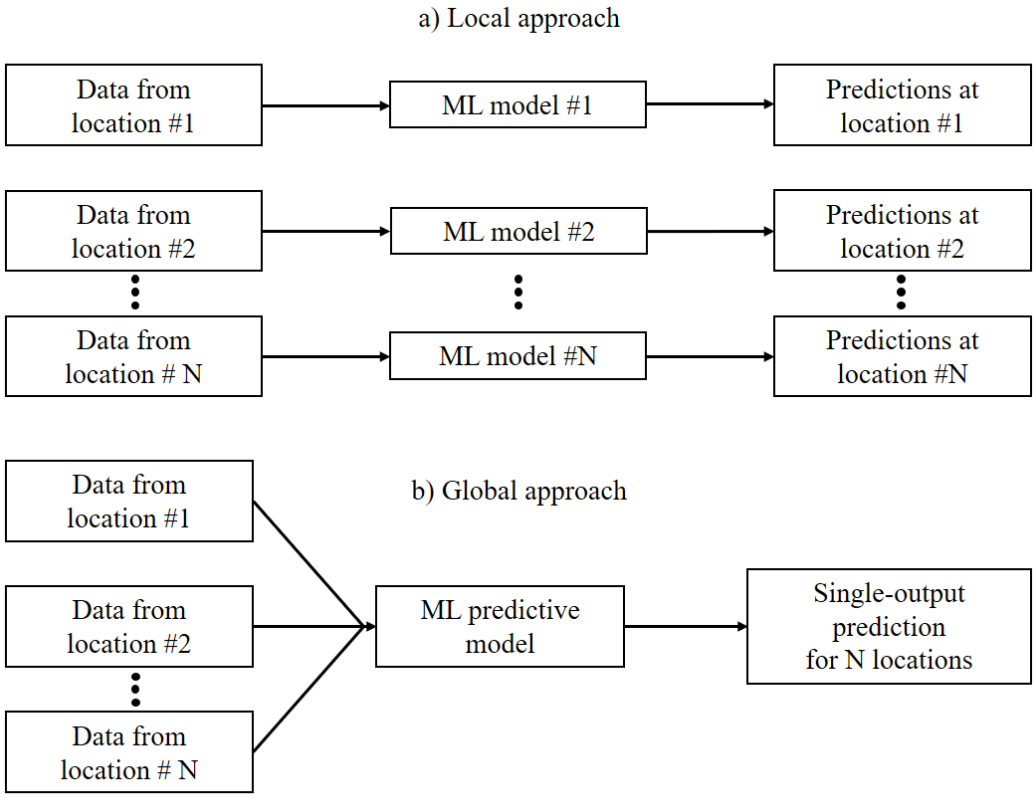


Target

The target attribute represents the number of locations for which predictions are carried out. Based on [9], the TF problem at hand can require traffic predictions to be made for more than one point or segment, or even for a whole network. In the case of a single target, the TF problem is approached as a simple regression or classification problem in which a unique ML model is trained and tested to predict traffic at the target location.

On the other hand, when the forecasting problem requires predictions to be made simultaneously for multiple locations, it can be handled as a multi-target problem [127]. In this case, there are two possible strategies to solve this problem, such as the ones shown in Figure 3.3. The first strategy is to define a local approach [127] that transforms the multiple locations into independent problems, and each of them is then solved using a simple regression or classification approach. This means developing one ML model for every site in which predictions are performed. The second strategy is to use a global approach [127] that adapts a single ML model to handle multiple datasets coming from the locations under study directly. This latter approach is usually more challenging because it aims not only to predict traffic at the various locations but also to model the dependencies among the sites [127], which can increase the computational cost.

Figure 3.3: Multi-target approaches to forecast traffic in multiple locations



3.2.1.3 **Problem Definition**

Within this attribute, we have identified four sub-levels. The first sub-level is related to the type of data used as an input. Based on [5], there are three types of inputs that can be used at the moment of feeding a ML model: using only temporal traffic data, temporal and spatial traffic data, and non-traffic data (e.g., calendar data) to enrich any of the two aforementioned inputs. The second sub-level consists of the time horizons for which predictions are made; in this taxonomy, we categorise them into two groups: short- and long-term time horizons. The third sub-level determines the time step defined for the input data, which, according to [12, 13] can be grouped in three ranges: high, medium, and small resolutions. Finally, the fourth sub-level is the output definition in terms of the kind of traffic variable to be predicted. Based on [13], the output variables can be travel time or

a variable within the group of fundamental macroscopic variables (flow, density, occupancy, speed). The four sub-levels mentioned above are presented as follows.

- **Input Definition:** A proper definition of input data is of great importance to ensure the excellent performance of ML methods in general [18], which also applies to TF. The main idea behind TF is to make predictions from a few seconds to possibly a few hours into the future using current and past traffic data, which is known as the temporal domain of traffic data [5]. Additionally, in recent research, including spatial traffic data has been an essential consideration in TF [9]. Several research articles have supported the improvement of predictions due to the incorporation of upstream or downstream traffic data [28, 29, 106, 116, 117, 121, 124, 128, 129].

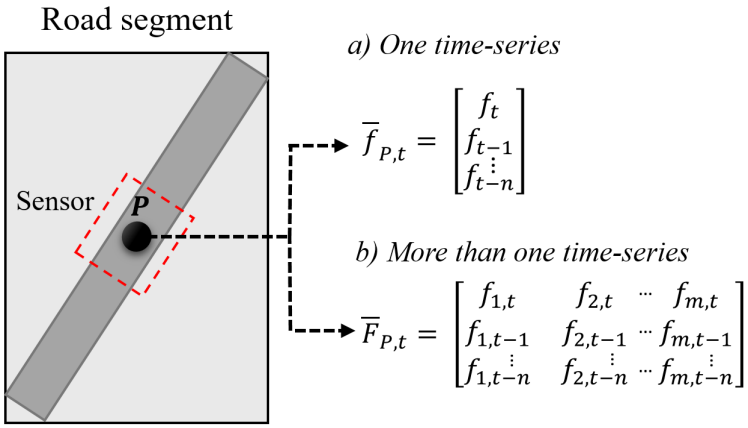
  On the contrary, some factors affect traffic but are not part of its pattern behaviour such as weather or calendar data. Feeding this type of non-traffic data into ML methods can enhance their predictions [130]. In this context, TF problems can contain up to three categories of input data, which are described below: temporal, temporal and spatial, and non-traffic data.

  - **Temporal:** Traffic predictions can be made using only temporal traffic data of the scale of prediction under study. In the transportation literature, we can find studies that use either one time-series associated with a single traffic measure or several time-series of different measurements taken by the same data sensing technology [5]. These two approaches are compared in Figure 3.4.

    Feeding ML models with only temporal data can be considered the most simple TF problem [131–135]. In the case of only using one time-series, the definition of the input data ends with a vector representation, which includes the current state f of the traffic variable of interest at time t and its previous n states f_{t-n}. On the other hand, when using more than one time-series belonging to m traffic variables, the input definition is a matrix representation that includes the current traffic states $f_{m,t}$ of the variables and their previous n values $f_{m,t-n}$. It is important to clarify that in both cases, the preparation of the dataset results

in a structure that contains features and samples. This is explained in more detail in the Modelling specifications (see Section 3.2.2).

Figure 3.4: Temporal traffic data. Adapted from [2]

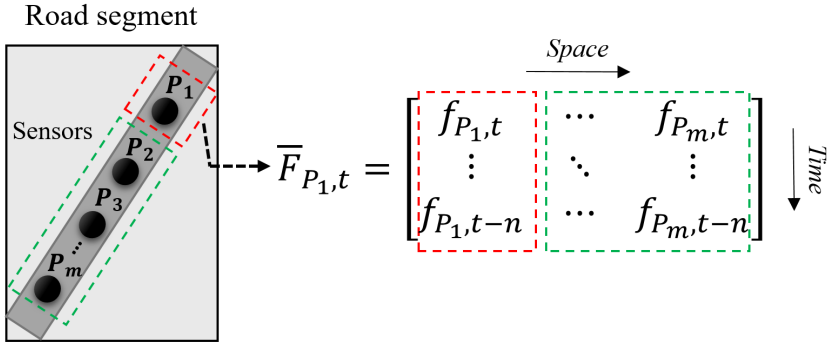


– **Temporal and Spatial:** Although there is extensive literature demonstrating that reasonable accuracy can be achieved using only temporal traffic data [132, 136–141], there is evidence that shows how incorporating the spatial component of traffic can improve the accuracy of predictions [9, 36]. Representing the predictions of traffic as a function of time and space is theoretically valid because considering temporal and spatial data from other physical locations allows the dynamics of traffic to be captured [13].

  The input definition of this data leads to a matrix representation that incorporates time and space domains using the temporal data from both the target position $P_{1,t-n}$ and the other m positions $P_{m,t-n}$, as is shown in Figure 3.5. This process requires identifying the spatial correlations of the locations under study to determine what positions are included and which can increase the computational complexity of the problem at hand. To reveal this dependency, it is necessary to examine the target location with its upstream and downstream sites and surrounding areas. The latter can be tackled using correlation analysis or by including a set of measurement positions within a pre-defined r radius [29, 125, 126]

with respect to the target location. Once this has been done, the next step is to transform the matrix representation to a dataset format with features and samples (see Section 3.2.2).

Figure 3.5: Temporal and spatial traffic data. Adapted from [2]



– **Non-traffic Data:** Factors that are not part of the seasonal dynamism of traffic can play an important role in the accuracy of predictions. Weather and time of year, among others, are elements that matter in the forecasting process. Although some of them can be difficult to predict, their inclusion in the preparation of datasets enhances the performance of ML models [2, 9, 130]. The addition of non-traffic data leads to developing responsive forecasting schemes that improve the decision-making process of traffic management [5, 9].

  Transportation literature reports that the incorporation of non-traffic data with temporal and spatial traffic data is an open issue [9]. Besides, calendar and weather data are the most frequently used exogenous variables for TF [9, 113, 131, 142–147]. From a data preprocessing perspective, the incorporation of non-traffic data increases the model complexity and dimensionality of the datasets. Besides, it is necessary to take standard procedures into account to integrate exogenous factors from different data sources into the traffic data provided by point, interval, or moving sensors (read further about approaches used for data fusion in traffic forecasting in [148]).

- **Time Horizon:** This attribute represents the extension of time into the future over which a traffic variable is predicted. We have identified two sub-levels of this attribute: a short-term attribute that categorises the most common TF problems that fall into the time horizon interval at less than 60 minutes ([28, 29, 106, 116–120, 125, 125, 128, 129, 149]); and a long-term attribute that enables the categorisation of TF problems focused on long-term time horizons at more than 60 minutes [115, 121, 131, 136, 139, 141, 144, 145, 150].

  According to transportation literature [12, 13, 151], it has been observed that longer time horizons generally cause more significant inaccuracy, and therefore most recent literature reviews show that research commonly predicts traffic up to 60 minutes into the future [5, 9]. Specifically, research suggests the appropriate prognosis horizon is between the range of 5 to 30 minutes into the future [13, 152]; however, Laña et al. [9] recommend that long-term time horizons, beyond 60 minutes, are needed to improve the management of traffic flows at the network level. In this context, enhancing the forecasting capacity of ML methods is still an open issue approached by very few authors (see Section 3.3 and [9]).

- **Time step:** In ML datasets preparation, defining the appropriate time step is an essential element because it affects the quality traffic information lying in the data. This attribute categorises the time interval upon which predictions are made and the frequency of time used to reach the time horizon defined in every TF problem. Three sub-levels are identified: high, medium, and small, and are described below.

  In general, high-resolution time steps (e.g., traffic data sensed every 30 seconds) incorporate noise to the input data [5, 13], and the resultant ML models are more prone to overfitting. In this sense, data preprocessing is required to reduce noise in the input data [153, 154]. On the other hand, there is the case of small resolution time steps. They influence the elimination of important data variations and traffic information within the data [155], which is needed during the training of models to keep a balance between bias and variance.

According to [5], there is no reliable approach to select the appropriate time step. In spite, the category of the medium resolution, which encompasses TF problems with time steps between 5 and 15 minutes, contributes to having an equilibrium between noise and the loss of valuable information within the data [13].

- **Output definition:** In every TF problem, there is a clear link between the traffic measure to be predicted and the data source used for such a task [16]. This attribute defines what traffic measure is considered as the output of ML methods depending on the type of data available. Here, we have identified two sub-levels. First, there is the category of fundamental macroscopic traffic variables [13] that includes flow, density, occupancy, and speed. On the other hand, the second sub-level is related to forecasting travel time which, as it is explained in the Travel Time Data Collection Handbook [156], provides a common ground for communication between transportation engineers, planners, administrators and non-expert travellers [13, 14, 16].

  - **Fundamental:** This category includes flow, occupancy, speed and the traffic measures that can be calculated based on them, for instance, LoS [99]. Flow and occupancy are traffic measures directly taken by point and interval detectors at the locations where the sensors are situated [16]. Sometimes, the sensors are placed on every lane of a road, and it is up to the modeller to decide if the predictions would be performed for each lane or aggregation of lanes. Current transportation literature shows that research endeavours are well distributed between flow and occupancy forecasting without any special considerations at the moment of choosing any of the two traffic measures [29, 106, 115, 117–120, 122, 123].

    The prediction of speed is strongly connected to whether the available data source senses this traffic measure or not. If the available data comes from point detectors, which do not have the capacity of detecting speed [16], flow and occupancy are used to calculate velocity; giving rise to a speed estimation rather than a prediction. If the available data source is interval detectors, they can collect the speed of each vehicle

by using its travel time between the two sensors [157]. This measure is called point speed, and it is only valid when describing the speed at the points where the sensors are located [158]. To prepare ML datasets with this latter measure, the point speed of all vehicles passing the sensors must be aggregated, generally, by means of the time mean speed technique to generate the velocity feature [156].

Finally, if the available data comes from moving sensors, speed can be incorporated in the GPS traces or not. In the event of not being included, it is possible to calculate the average speed for each road segment, travelled by every moving sensor, which has at least two GPS data points. The resultant measure is the traffic speed at that specific road segment for each vehicle sensed (for more details about how to preprocess this data see [159, 160]). Then to forecast speed, a time-series of traffic speed must be built by aggregating the estimated speeds of every moving sensor available. Such predictions have segment and network spatial coverage that offers more details about the real traffic conditions on the roads [28, 124, 126].

– **Travel Time:** This attribute is associated with another significant direction of TF problems focusing on forecasting travel time. It is defined as the time needed to cross two fixed points along a road [13]. As in the case of speed prediction, travel time forecasting is connected to the availability of the appropriate data for such a task [5, 16]. If the available data source technology supports the direct sensing of travel time (e.g., GPS, AVI systems), it is predicted using the measured data taken by detectors [13] (see [16] for more details regarding sensing technologies of travel time).

  In the case of using point sensors [129, 161–163], travel time forecasting is based on their capability of sensing point speed, and then, making use of trajectory methods [16] during the preparation of ML datasets to calculate travel time. The main idea of these methods is to take a whole road and divide it into smaller segments, where each of them is defined as the length between two detectors. The detector at the beginning of

one segment is called the upstream sensor, while the one at the end of the segment is the downstream detector. With this configuration, the most straightforward way to extend the point speed measurements to the segment is by using piece-wise constant methods [164–166]. Thus, a travel time feature is generated based on the point speed measures belonging to the areas of interest.

3.2.2 Modelling Specifications

Supervised ML is the process of learning a mapping function ($f : X \rightarrow Y$) between Y, the dependent variable/s, and X, the independent variable/s [55]. The focus is on modelling and predicting how the dependent variable/s change/s when the independent variable/s vary/ies over time, as is the case for TF.

Most of the transportation literature describes characteristics of TF problems centered on traffic-related attributes [5, 9, 12–14, 16, 36]. Nevertheless, none of them gives an account of criteria that categorise how to modelling: 1) the input and output data, 2) the steps of predictions, and 3) the preprocessing process of the input data.

In this section, we define the attributes: input modelling, output modelling, step of prediction, and data preprocessing. Each of them, together with their sub-attributes, are presented and described below.

3.2.2.1 Input Modelling

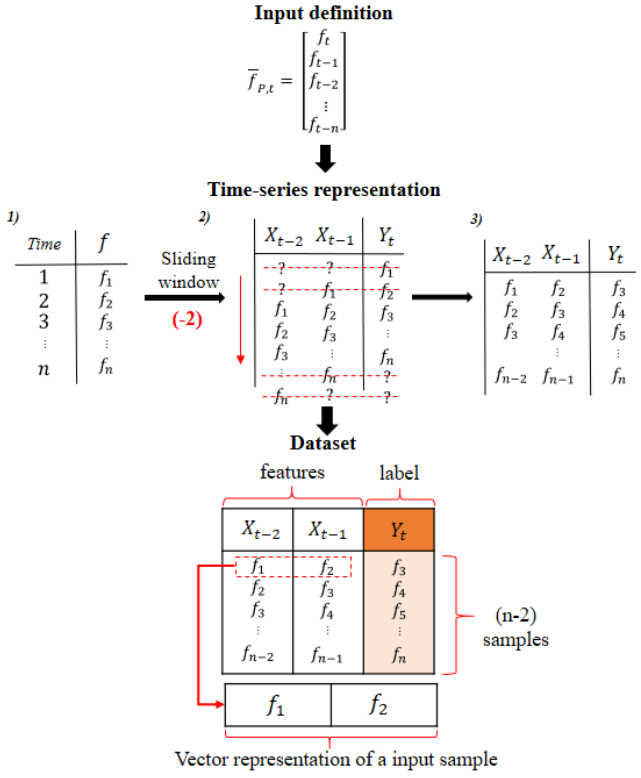
This attribute is related to how the samples of the input data are modelled to generate a dataset. We have identified two sub-levels within it. In the first case, the samples are modelled as vectors, while the second sub-level categorises TF problems where the input samples are modelled as high-order tensors. These two input modelling categories are described as follows.

Vector

This attribute presents the TF problems in which every input data sample is modelled with a vector representation. We give the following two examples to illustrate this.

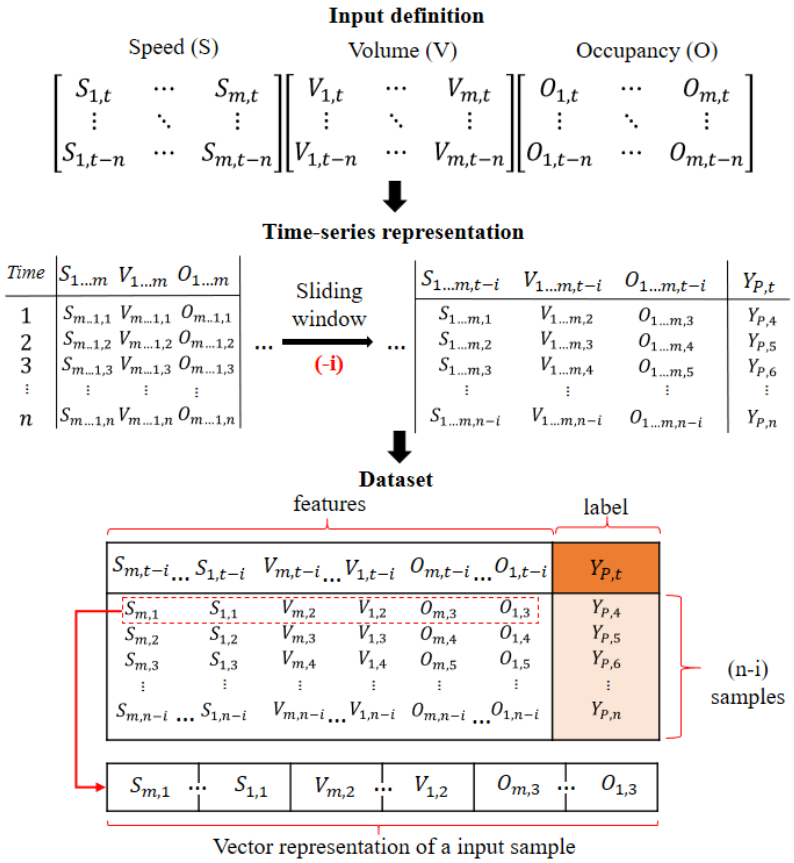
The first example corresponds to a fundamental TF problem whose input contains only temporal traffic data, as is depicted in Figure 3.6. Explicitly, given a sequence of historical data of (n) measures, which belongs to a traffic parameter (f) sensed during a time interval (t) at a position (P), it can be modelled to look like a supervised learning problem employing the sliding window method [78].

Figure 3.6: Vector to modelling temporal traffic data



According to Figure 3.6, after the input definition phase, the next step is to generate a time-series representation from the original sequence of data in which the order of the samples is preserved. Then, two copies of the time-series are used to generate two lagged-input variables named $X_{(t-1)}$ and $X_{(t-2)}$. Having done this, the sliding window method consists of using the values of the previous time steps, within $X_{(t-1)}$ and $X_{(t-2)}$, to predict the values at the next time steps in $Y_{(t)}$. For the illustrative purposes of this example, we use a window size value of two. Nevertheless, careful thought and experimentation are needed within every TF problem to find a window width that results in acceptable model performance.

Figure 3.7: Vector to modelling temporal and spatial traffic data



After applying the sliding window method, the first two rows of $X_{(t-1)}$ and $X_{(t-2)}$ have insufficient data to predict the $f_{(1)}$ and $f_{(2)}$ values in $Y_{(t)}$. Besides, there are no known next values in $Y_{(t)}$ to be predicted using the $f_{(n)}$ measures of $X_{(t-1)}$ and $X_{(t-2)}$. These four rows of missing values are deleted to obtain the final dataset, which has two features ($X_{(t-1)}$ and $X_{(t-2)}$) and a column of target labels ($Y_{(t)}$). In this resultant data structure, each pair ($X_{(i-1)}$, $X_{(i-2)}$) of samples is modelled as a two-dimension vector that can be fed into any of the standard linear and nonlinear ML methods.

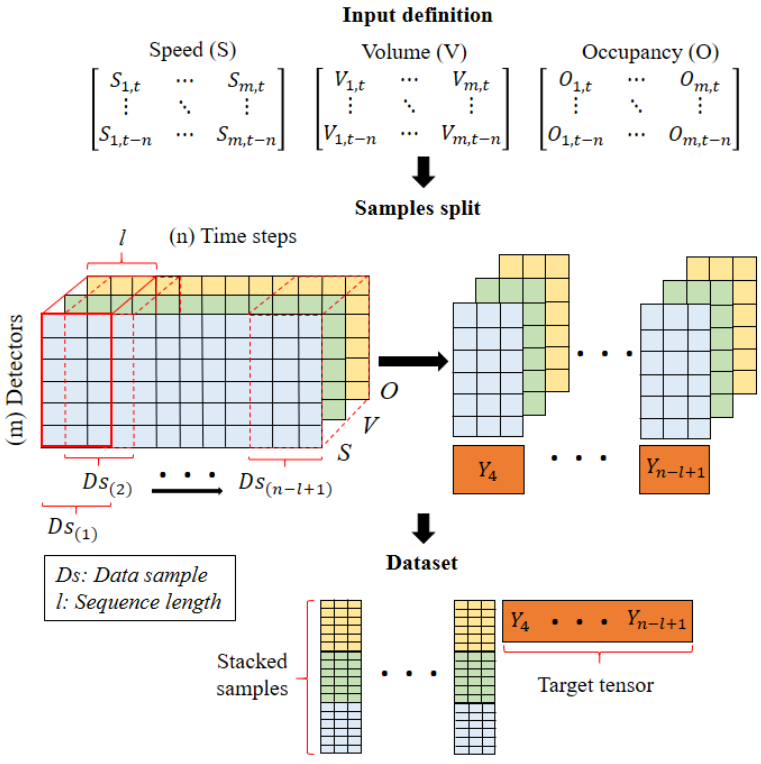
The second example is presented in Figure 3.7, which shows a more complex TF problem. In the input definition phase, there are three different traffic measures: speed (S), volume V, and occupancy O. Each of them has (m) sequences of historical data sensed by (m) detectors spatially ordered along a road segment,

during a time interval (t). Following the process described in Figure 3.6, the sliding window method re-frames the input data as a supervised learning problem with a generic window value of (i). Having done this, the resultant process leads to up to $S_{(m,t-i)}$, $V_{(m,t-i)}$, and $O_{(m,t-i)}$ features being obtained. Their values at the $(n-i)$ previous time steps are used to forecast the real or discrete values within $Y_{(P,t)}$, P being the position where the target detector is located.

High-order Tensor

This attribute categorises the TF problems wherein the input data is modelled using a tensor of an order greater than two. Figure 3.8 illustrates how the three traffic measures ($speed$, $volume$, $occupancy$), used in the example of Figure 3.7, are put together to generate a three-dimensional data matrix of size $(m, n, 3)$. This 3-D data structure contains $(3 * m)$ rows whose historical sequences are sensed during (n) time steps over (m) detectors spatially ordered on a road segment.

Figure 3.8: High-order tensor to modelling temporal and spatial traffic data



The next step is to define a sequence length (for this example with a value of 3) to simultaneously sample the three dimensions of the matrix and extract $(n-l+1)$ data samples of size $(m, 3, 3)$, as is shown in Figure 3.8. These data samples are vertically stacked to produce the data-set that is commonly fed into Deep Learning (DL) methods, which can capture the spatial dependencies of traffic data concerning traditional ML methods in a more realistic way [28, 106, 129]. As with the sliding window method, the sequence length needs to be carefully defined using experimentation to achieve appropriate model performance.

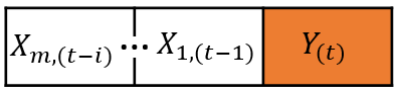
3.2.2.2 Output Modelling

This attribute categorises TF problems depending on the number of traffic variables to be predicted. Within it, we have identified two sub-levels. On the one hand, if there is only one variable to forecast, the modelling process leads to a single-target problem. Contrarily, if the TF problem at hand is focused on making predictions for more than one variable, the modelling process of the output is a multi-target problem. These two input modelling processes are described below.

Single-target

This attribute categorises TF problems in which there is only one traffic measure to be predicted (Figure 3.9). From a ML perspective, this is known as a standard regression or classification problem [55] that consists in training a model able to predict, for either a given vector or high-order tensor sample, a real or discrete value. As shown in Figure 3.9, this problem has the form $Y_{(t)} = X_{1,(t-i)}$ wherein $Y_{(t)}$ is the output vector that contains the value of the following steps values to be predicted using previous time steps $X_{m,(t-i)}$.

Figure 3.9: Single-target output



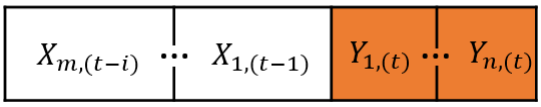
Currently, most research on the transportation literature is focused on predicting a single traffic variable using either only temporal traffic data or both temporal

and spatial traffic data [28, 29, 106, 115–118, 120–125, 125, 126, 128, 129, 149]. Besides, the most common target is any of the fundamental macroscopic traffic variables [131–140, 142, 146, 147, 152]; whilst, the prediction of travel time has been less explored [121, 162, 163, 167].

Multi-target

This attribute presents those TF problems that have more than one output variable to be predicted (Figure 3.10). From a ML approach, this type of problem has an output space of more than one dimension [55]. As can be seen in Figure 3.10, a multi-target problem has the form $(Y_{(1)}, ..., Y_{(n)}) = (X_{1,(t-1)}...X_{m,(t-i)})$ in which there are $Y_{(n)}$ traffic measures to be predicted based on the previous time steps of $X_{(m)}$ input features.

Figure 3.10: Multi-target output



The transportation community has recently assumed this multi-target approach to model, comprehensively, the exact form of traffic dynamics. It does generate, however, issues related to the selection of the proper data-driven method. The overall experience in multi-target modelling points out the use of non-parametric techniques, such as NNs [13, 168], to predict fundamental macroscopic traffic variables together with travel time [119, 161, 169].

3.2.2.3 Step of Prediction

This attribute is related to modelling the number of time horizons in which traffic predictions are made into the future. We have identified two sub-levels in this attribute: single wherein predictions are performed for a single time step, and multiple wherein predictions are done over more than one time horizon. These two sub-levels are presented below.

Single

This attribute categorises TF problems in which there is only one time horizon. It corresponds to a single-target problem wherein there is a single traffic measure to be predicted. The latter has the form $Y_{(t)} = X_{m,(t-i)}$, being (t) the target step.

In the transportation literature, TF problems with a single step of predictions are approached through parametric and non-parametric ML methods [29, 117–119, 122, 123, 126, 149]. Nevertheless, from a practical perspective, single prediction intervals can support neither the short-term operational decision-making nor the medium- and long-term transportation planning, as they cannot describe how traffic will evolve beyond few minutes into the future [9, 13].

Multiple

This attribute classifies the TF problems wherein there is more than one future time step to be predicted. Specifically, there can be multiple time steps for either one target traffic measure or a set of them; which leads to a multi-target problem in the form $Y_{n,(t+i)} = X_{m,(t-i)}$. Let (n) be the output traffic measures and $(t + i)$ the number of time horizons to forecast.

TF problems with more than one time horizon in the future are quite well-documented in the literature [28, 106, 115, 116, 120, 121, 125, 125, 128, 129]. In contrast to single time step problems, multiple steps provide a more robust ground for decision-makers during traffic flow management [13, 168].

3.2.2.4 Data Preprocessing

This attribute categorises TF problems in terms of the preprocessing tasks required to model their input data in a way that allows them to be processed by ML methods. According to data mining literature [42], Data Preprocessing (DPP) can be divided into data preparation and data reduction (more details of these two categories, with their respective techniques, can be consulted in [42]). However, for the purpose of the proposed taxonomy, we have split them into three categories: data preparation, incomplete data, and data reduction.

Data Preparation

This attribute categorises TF problems with respect to processes required for preparing the raw traffic data into the minimum format accepted by ML methods. Data preparation guarantees that the methods operate correctly without reporting errors during their run-time due to a no valid data format. Within this attribute, three approaches are presented below.

The first approach is data cleaning that is focused on detecting and discarding corrupt records within the raw data. The latter means eliminating incorrect data that does not make sense in the context of traffic measures, for instance, negative values. The second approach is the data transformation, whose objective is to improve the input data to become more efficient in the forecasting process of ML methods. This task involves generating new features and normalising or rescaling the input data to establish the same measurement scale.

Finally, the third approach is data integration that is related to the merging process of data that comes from different sources (for more details see [148]). In this case, such combination corresponds to the fusion of traffic data sensed by multiple sensors or, even, the inclusion of non-traffic data into the traffic data-sets.

Imperfect Data

This attribute classifies TF problems about the processes needed to fill in missing values and to identify noise in the raw data. In the first case, faulty reading, malfunctioning hardware or transmission errors of the traffic detectors can cause empty records. Filling in these gaps is essential to guarantee accurate predictions. This process can be performed using different strategies ranging from a simple null value imputation to complex spatio-temporal context imputation models (see more details of the imputation of missing data for road traffic forecasting in [170]). In the latter case, the objective is dealing with noisy input data to detect random errors and to reduce data variability using smoothing techniques. This can improve the quality of the training data and avoid overfitting issues.

Data Reduction

This attribute categorises TF problems in terms of data reduction techniques applied to obtain a reduced representation of the input data; which enables ML methods to reduce the computational cost of their training process. This representation is smaller at the same time it maintains the integrity and variability of the original data [42, 171]. The three approaches that categorise the Data Reduction of TF problems are explained below.

First, the feature reduction approach aims to remove irrelevant-redundant features to find a minimum set of attributes that increases the speed of the learning process (for more details, see the review in [172]). The second approach is instance reduction, which is related to choosing a subset of samples from the whole traffic dataset to achieve the desired forecasting performance as if the complete data was used by the ML methods [42].

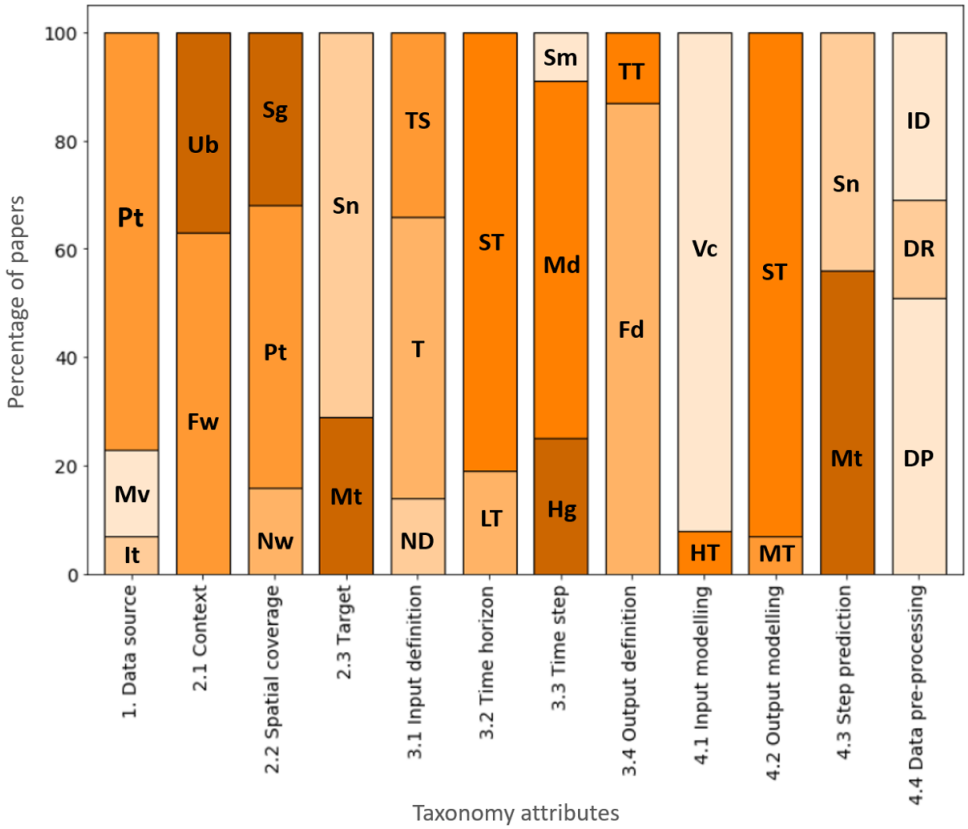
Lastly, the third approach is discretisation, which transforms continuous data values into categorical values within a finite number of intervals. This is useful for the integration of non-traffic data such as weather conditions, wherein constant values are mapped into discrete attributes that classify the climate conditions on the roads based on nominal values.

3.3 The Taxonomy in Action: Categorisation of Traffic Forecasting Literature

In this section, we analyse the taxonomy to check its robustness and its ability to discriminate papers that have approached different TF supervised problems. Table 3.1 is devoted to categorising a sample of relevant articles published between 2017 and 2019, and Table 3.2 includes references dated from 2014 to 2016 that are taken from the most updated survey in TF [9]. Besides, Table 3.3 exhibits a sample of the main references from 2000 to 2013.

Every paper is categorised according to the sub-attributes within the four main blocks of the taxonomy (in Figure 3.1 they are 1. Data Source, 2. Scope, 3. Problem Definition, 4. Modelling). The names of these main blocks are the shaded columns in Tables 3.1, 3.2 and 3.3. When a particular sub-attribute is present in the

Figure 3.11: Summary of 2000-2019 TF literature categorised by the taxonomy



1. Data Source: Point (Pt), Interval (It), Moving (Mv); ***2.1 Context:*** Freeway (Fw), Urban (Ub); **2.2 *Spatial Coverage*:** Point (Pt), Segment (Sg), Network (Nw); ***2.3 Target:*** Single (Sn), Multiple (Mt); ***3.1 Input Definition*:** Temporal (T), Temporal + Spatial (TS), Non-traffic Data (ND); ***3.2 Time Horizon*:** Short-term (ST), Long-term (LT); ***3.3 Time step***: High (Hg), Medium (Md), Small (Sm); ***3.4 Output Definition*:** Fundamental (Fd), Travel Time (TT); ***4.1 Input Modelling:*** Vector (Vc), High-order Tensor (HT); ***4.2 Output Modelling:*** Single-target (ST), Multi-target (MT); ***4.3 Step Prediction:*** Single (Sn), Multiple (Mt); ***4.4 Data Pre-processing:*** Data Preparation (DP), Imperfect Data (ID), Data Reduction (DR).

TF problem approached by an article, its acronym (see Figure 3.1) is added to that specific cell. All described sub-attributes are present in at least one paper, which shows that no unnecessary attributes have been introduced in the taxonomy.

Figure 3.11 presents a summary of the findings extracted from the categorisation of the literature published between 2000 and 2019. Concretely, on its x-axis, all the attributes within the four blocks of the taxonomy can be seen. For each attribute, there is a stacked bar that depicts the percentage of the transportation literature (y-axis) that has addressed its sub-attributes.

As can be seen in Figure 3.11, regarding attribute **1. Data Source**, point detectors are the most common sensing technology used. The majority of the transportation literature categorised has approached TF problems that contain this type of data source exclusively. Only three papers [143, 169] highlighted in Tables 3.1 and 3.2 include more than one kind of data source. In this sense, the integration of multiple data sources is an opportunity from the transportation perspective and a challenge in the ML area. Using multiple data technologies can contribute to representing the traffic conditions on the roads in a more realistic way; however, such a data fusion is a demanding modelling task during the preprocessing process of the input data.

The data source availability mentioned above also affects the spatial coverage of predictions. The stacked bar of attribute **2.2 Spatial Coverage** shows that a few papers have approached TF problems at the network level. The latter is in concordance with the low usage of moving data due to privacy and availability issues [16]. On the other hand, the most common TF problems are the ones in which the prediction of traffic is focused on either point or segment levels using point or interval detectors.

In the case of attribute **2.1 Context** there is an apparent balance in the literature between the TF problems handled in either urban or freeway contexts. Only two studies in Table 3.1 approach the prediction of traffic in both settings at the same time. In this case, the computational challenge is focused on developing methods that learn traffic patterns of both contexts or developing models that separately predict traffic for the two environments.

Concerning the attribute **2.3 Target**, Figure 3.11 shows that most of the literature approaches TF problems that consider a single spatial target for predictions. Multiple targets are considered when traffic is only predicted for more than one point or segment. For multiple targets, research is still needed to find out whether adopting a local or a global strategy to handle the prediction along various locations would more suitable in terms of computational cost and accuracy.

In the case of attribute **3.1 Input Definition**, the majority of papers have addressed the prediction of traffic using only temporal traffic data. When spatial traffic data and non-traffic data are considered to enhance TF, such input enrichment

Table 3.1: Transportation literature, published between 2017 and 2019, categorised by the taxonomy

References	1. Data source	2. Scope	2.1 Context	2.2 Spatial coverage	2.3 Target	3. Problem definition	3.1 Input definition	3.2 Time horizon	3.3 Time step	3.4 Output definition	4. Modelling	4.1 Input modelling	4.2 Output modelling	4.3 Step prediction	4.4 Data pre-processing
[125]	Pt		Ub	Sg	Mt		T	ST	Md	Fd		Vc	ST	Mt	ID + DR
[128]	Pt		**Fw + Ub**	Sg	Mt		TS	ST	Md	Fd		**HT**	ST	Mt	DP + ID
[119]	Pt		Ub	Pt	Sn		T	ST	Md	Fd		Vc	**MT**	Sn	-*
[120]	Pt		Ub	Pt	Mt		T	ST	Md	Fd		Vc	ST	Mt	-*
[115]	Pt		Fw	Sg	Mt		T	LT	Hg + Md	Fd		Vc	ST	Mt	-*
[116]	Pt		Fw	Sg	Sn		TS	ST	Hg	Fd		Vc	ST	Mt	DP
[117]	Pt		Fw	Pt	Mt		TS	ST	Md	Fd		Vc	ST	Sn	DP
[118]	Pt		Fw	Pt	Sn		T	ST	Hg	Fd		Vc	ST	Sn	-*
[149]	Pt		Fw	Pt	Mt		T	ST	Md	Fd		Vc	ST	Sn	-*
[121]	It		Fw	Sg	Sn		TS	LT	Md	TT		Vc	ST	Mt	DP
[129]	Pt		Fw	Pt	Mt		TS	ST	Md	Fd		**HT**	ST	Mt	DP
[28]	Mv		Ub	Nw	Sn		TS	ST	Md	Fd		**HT**	ST	Mt	-*
[106]	Mv		Ub	Nw	Sn		TS + ND	ST	Md + Sm	Fd		**HT**	ST	Mt	DP
[29]	Pt		Fw	Pt	Mt		TS	ST	Md	Fd		Vc	ST	Sn	DP
[173]	**Pt + Mv**		**Fw + Ub**	Pt + Sg	Mt		TS	ST	Md	Fd		**HT**	ST	Sn	DP + ID
[126]	Mv		Ub	Nw	Sn		TS	ST	Md	Fd		Vc	ST	Sn	DR
[122]	It		Fw	Sg	Sn		T	ST	Md	Fd		Vc	ST	Sn	-*
[123]	Pt		Ub	Sg	Sn		T	ST	Md	Fd		Vc	ST	Sn	ID + DR
[124]	Mv		Ub	Sg	Mt		TS	ST	Md	Fd		Vc	ST	Sn	ID + DR

is usually modelled through vector representations (stacked bar **4.1 Input Modelling**). The latest can lead to losses in the spatial dependencies of traffic data. As is highlighted in Tables 3.1 and 3.2, a few studies [28, 106, 128, 129, 184] have used a high-order tensor to generate multi-dimensional data input structures.

Regarding attributes **3.2 Time Horizon** and **3.3 Time Step**, the forecasting of traffic is focused on short-term predictions using medium resolution data (between 5 and 15 minutes). In the case of both small and high time steps, the literature avoids these data resolutions because of the loss of valuable information and the inclusion of noise in the input data. In this context, the data science challenge lies in determining how the structure of the input data can contribute to obtaining long-term predictions without losing accuracy and keeping low the computational cost of training ML models.

In the area of what traffic measure is predicted (attribute **3.4 Output Defini-**

Table 3.2: Transportation literature, published between 2014 and 2016, categorised by the taxonomy

References	1. Data source	2. Scope	2.1 Context	2.2 Spatial coverage	2.3 Target	3. Problem definition	3.1 Input definition	3.2 Time horizon	3.3 Time step	3.4 Output definition	4. Modelling	4.1 Input modelling	4.2 Otput modelling	4.3 Step prediction	4.4 Data pre-processing
[174]	Pt		Ub	Sg	Sn		T	ST	Hg	Fd		Vc	ST	Sn	DP + ID
[169]	**Pt + It + Mv**		Fw	Sg	Sn		T	ST	Md	Fd + TT		Vc	**MT**	Mt	ID + DR
[142]	Mv		Ub	Nw	Sn		TS + ND	ST	Md	Fd		Vc	ST	Mt	DP + ID
[131]	Pt		Fw	Sg	Sn		T + ND	LT	Md	Fd		Vc	ST	Mt	DP
[136]	Pt		Ub	Pt	Mt		T	LT	Sm	Fd		Vc	ST	Mt	DP + ID
[146]	Pt		Ub	Nw	Sn		TS + ND	ST	Hg	Fd		Vc	ST	Mt	DP
[137]	Pt		Ub	Pt	Mt		T	ST	Md	Fd		Vc	ST	Sn	ID
[138]	Pt		Ub	Pt	Mt		T	ST	Sm	Fd		Vc	ST	Mt	-*
[147]	It		Ub	Nw	Sn		TS + ND	ST	Hg	Fd		Vc	ST	Mt	DP + DR
[132]	Pt		Fw	Pt	Sn		T	ST	Hg	Fd		Vc	ST	Mt	ID
[139]	Pt		Fw	Sg	Sn		T	LT	Hg	Fd		Vc	ST	Mt	-*
[133]	Pt		Fw	Pt	Sn		T + ND	ST	Md	Fd		Vc	ST	Sn	DP
[134]	Pt		Fw	Sg	Sn		TS	ST	-	Fd		Vc	ST	-	DP
[152]	Pt		Fw	Nw	Sn		TS	ST	Hg	Fd		Vc	ST	Sn	DP
[140]	Pt		Ub	Pt	Mt		T	ST	Md	Fd		Vc	ST	Sn	-*
[135]	Pt		Fw	Pt	Sn		T + ND	ST	Hg	Fd		Vc	ST	Mt	DP + DR
[161]	Pt		Fw	Pt	Mt		TS	ST	Hg	Fd + TT		Vc	**MT**	Sn	DP + ID
[141]	Pt		Fw	Pt	Mt		T	LT	Md	Fd		Vc	ST	Mt	DR
[162]	Pt		Fw	Nw	Sn		TS	ST	Hg	TT		Vc	ST	Mt	DP
[144]	Pt		Fw	Sg	Sn		T + ND	LT	Hg	Fd		Vc	ST	Mt	DP
[145]	Pt		Fw	Pt	Sn		T + ND	LT	Md	Fd		Vc	ST	Mt	DP
[175]	Pt		Fw	Sg	Sn		T	ST	Md	Fd		Vc	ST	Sn	ID
[176]	Pt		Fw	Pt	Sn		T	ST	Md	Fd		Vc	ST	Sn	-*
[177]	Pt		Fw	Pt	Sn		TS	ST	Md	Fd		Vc	ST	Sn	DP
[178]	Pt		Fw	Sg	Sn		TS + ND	ST	Md	Fd		Vc	ST	Mt	DP
[179]	Pt		Fw	Pt	Sn		T	ST	Md	Fd		Vc	ST	Sn	DP
[180]	Pt		Fw	Pt	Sn		T	ST	Md	Fd		Vc	ST	Mt	DP
[181]	Mv		Ub	Nw	Sn		T	ST	Md	Fd		Vc	ST	Mt	DR
[182]	Mv		Fw	Sg	Sn		TS	ST	Hg	Fd		Vc	ST	Sn	-*
[183]	Mv		Ub	Pt	Sn		T	ST	Md	Fd		Vc	ST	Sn	DP
[163]	It		Ub	Nw	Sn		TS	ST	Md	TT		Vc	ST	Mt	ID
[167]	Mv		Fw	Sg	Sn		TS + ND	ST	Md	TT		Vc	ST	Mt	ID
[150]	Pt		Ub	Pt	Sn		T	LT	Md	Fd		Vc	ST	Mt	DP
[143]	**Pt + It + Mv**		Fw	Nw	Sn		T + ND	ST	Md	Fd		Vc	ST	Sn	DP
[168]	Pt		Ub	Pt	Mt		T	LT	Md	Fd		Vc	ST	Mt	DR
[184]	Pt		Ub	Pt	Mt		T	ST	Md	Fd		**HT**	ST	St	-*
[185]	Pt		Ub	Nw	Sn		TS	ST	Md	Fd		Vc	ST	Mt	DP + ID

Table 3.3: Transportation literature, published between 2000 and 2013, categorised by the taxonomy

References	1. Data source	2. Scope	2.1 Context	2.2 Spatial coverage	2.3 Target	3. Problem definition	3.1 Input definition	3.2 Time horizon	3.3 Time step	3.4 Output definition	4. Modelling	4.1 Input modelling	4.2 Output modelling	4.3 Step prediction	4.4 Data pre-processing
[186]	Pt		Fw	Sg	Sn		TS	ST	Hg	TT		Vc	ST	Mt	DP + ID
[187]	Pt		Fw	Pt	Sn		T	ST	Hg	Fd		Vc	ST	Mt	-*
[188]	Pt		Fw	Sg	Sn		T	ST	Md	Fd		Vc	ST	Sn	DP + DR
[189]	Pt		Fw	Pt	Sn		T	ST	Md	Fd		Vc	ST	Sn	DR
[190]	Pt		Fw	Pt	Sn		TS	ST	Hg	Fd		Vc	ST	Mt	DP + ID
[191]	Pt		Fw	Sg	Sn		T	ST	Md	Fd		Vc	ST	Mt	ID
[192]	Pt		Fw	Pt	Sn		T	LT	Sm	TT		Vc	ST	Mt	-*
[111]	Mv		Ub	Sg	Sn		TS	ST	Hg	Fd		Vc	ST	Sn	-*
[193]	Pt		Fw	Pt	Sn		T	ST	Md	Fd		Vc	ST	Sn	-*
[194]	Pt		Fw	Pt	Mt		T	ST + LT	Hg + Md + Sm	Fd		Vc	MT	Mt	-*
[195]	Pt		Fw	Pt	Mt		T	ST + LT	Sm	Fd		Vc	ST	Mt	-*
[196]	Pt		Ub	Pt	Sn		T	ST	Md	Fd		Vc	ST	Sn	-*
[197]	Pt		Fw	Pt	Mt		T	ST	Md	Fd		Vc	ST	Sn	DP + ID
[198]	Pt		Ub	Pt	Sn		T	ST	Md	Fd		Vc	ST	Mt	-*
[199]	Pt		Fw	Pt	Sn		TS	ST	Md	Fd		Vc	ST	Sn	DP
[200]	Pt		Ub	Pt	Mt		T	ST	Hg + Md	Fd		Vc	ST	Mt	-*
[201]	Pt		Fw	Pt	Sn		TS	ST	Md	Fd		Vc	MT	Sn	DP + ID
[202]	Pt		Fw	Pt	Sn		T	LT	Sm	TT		Vc	ST	Mt	-*
[203]	Pt		Fw	Sg	Sn		T	LT	Md	TT		Vc	ST	Mt	-*
[204]	Pt		Fw	Sg	Sn		TS	ST	Md	Fd		Vc	ST	Sn	DP + ID + DR

tion), choosing any of the fundamental macroscopic variables is the predominant approach despite the relevance of forecasting time travel being previously stated in [5, 13, 16]. From the perspective of how many traffic parameters are predicted, the taxonomy demonstrates that most TF problems are focused on single-target output predictions (stacked bar **4.2 Output Modelling**). The latter opens the possibility to explore the benefits and challenges of multiple output targets, which according to Tables 3.1 and 3.2 has been approached by very few studies [119, 161, 169].

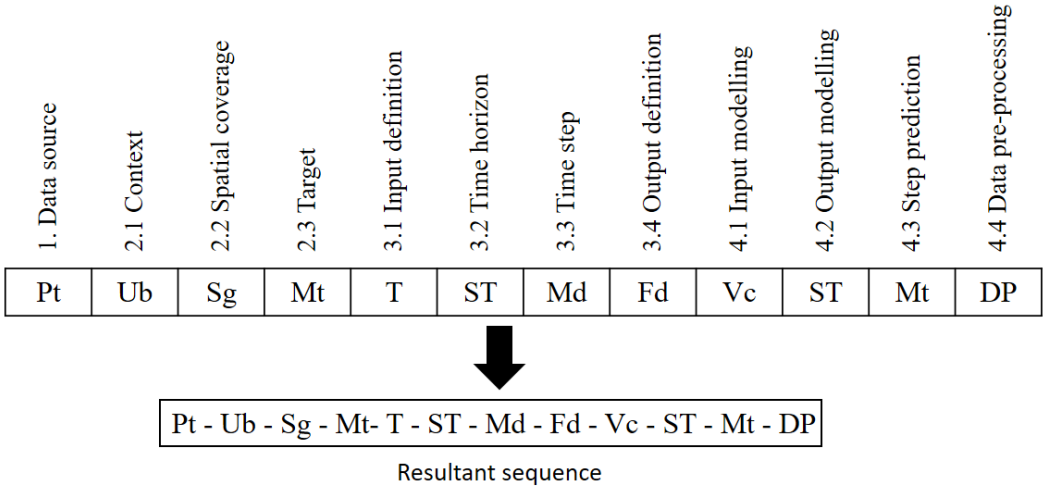
Finally, the DPP in TF problems (attribute **4.4 Data Pre-processing**) is mostly focused on data preparation tasks, particularly, data cleaning, data integration and data transformation. Notwithstanding, Tables 3.1, 3.2 and 3.3 show that many papers reviewed do not include any DPP. These cases are highlighted with the symbol

(-*) in the column **4.4 Data Pre-processing** of the tables, and although the authors performed the aggregation of input data into small, medium, or high time steps, this is not considered to be DPP because is an implicit task for all TF problems. The absence of DPP could be due to the fact that many articles use data sources sensed by third parties, which preprocess the traffic data before making it available.

3.4 Families of Traffic Forecasting Problems

Based on the literature categorisation, this section presents families of TF problems, that is, groups of problems that share common attributes of the taxonomy. To accomplish this purpose, first, we generated a unique sequence for each reference in Tables 3.1, 3.2 and 3.3. The sequence identifies the TF problem approached within every article. As can be seen in the example of Figure 3.12, the structure of each sequence is determined by both the traffic- and the modelling-related attributes assigned with the proposed taxonomy. We obtained 76 sequences that correspond to the total of papers reviewed between 2000 and 2019.

Figure 3.12: Example of sequence structure for a given TF problem

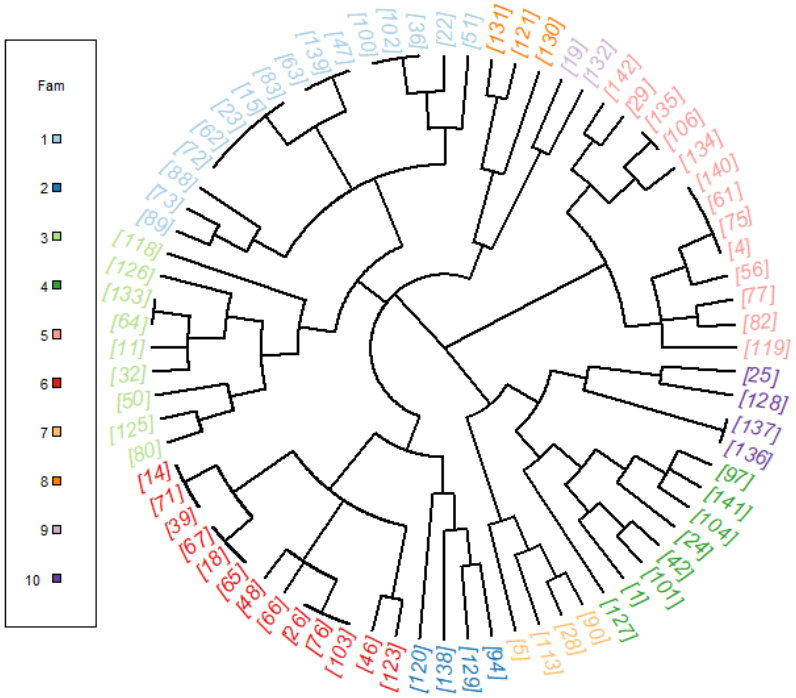


In the next step, we carried out a hierarchical clustering analysis considering only the traffic-related attributes. The clustering process was done comparing the 76 generated sequences, and those with up to 4 different transportation attributes were grouped in the same family. The modelling attributes were excluded in the

generation of the families because we wanted to highlight common groups of TF problems from a transportation perspective regardless of how they have been modelled in the literature.

We identified ten families of TF problems that are shown in the dendrogram of Figure 3.13. The branches represent how close one paper is to other studies in terms of familiarity determined by the transportation attributes, and the colours depict the families. The leaf nodes of the tree structure contain the papers categorised in Tables 3.1, 3.2 and 3.3.

Figure 3.13: Hierarchical clustering analysis to extract families of TF problems



The ten families of problems are presented with more details in Table [1] 3.4.

[1]Guide of acronyms: ***1. Data Source***: Point (Pt), Interval (It), Moving (Mv); ***2.1 Context***: Freeway (Fw), Urban (Ub); ***2.2 Spatial Coverage***: Point (Pt), Segment (Sg), Network (Nw); ***2.3***

The Table shows what traffic attributes characterise the families and how many of the papers reviewed have been approached within each of them. The missing cells along a single-family indicate that those transportation attributes, arranged in the columns of Table 3.4, do not belong to the family under consideration.

Table 3.4: Families of TF problems with their main traffic attributes together with the number of works classified within each family

Family	Family's main traffic attributes								Total of
	1.	**2.1**	**2.2**	**2.3**	**3.1**	**3.2**	**3.3**	**3.4**	**works**
Family 1	Pt	-	-	Sn	T	ST	Md	Fd	16
Family 2	Pt + It + Mv	Fw	Sg	Sn	-	ST	Md	-	4
Family 3	Mv, It	Ub	Nw	Sn	TS + ND	ST	-	Fd	9
Family 4	Pt	Fw	Sg	Sn	-	ST, LT	-	Fd	8
Family 5	Pt	-	Pt	Mt	T	ST, LT	-	Fd	13
Family 6	Pt	Fw	Pt	-	T + ND, TS	ST	-	Fd	13
Family 7	Pt	Fw	-	Sn	TS	ST	Hg	-	4
Family 8	Mv	Ub	Sg	-	TS	ST-	Md	Fd	3
Family 9	Pt + Mv	Fw + Ub	Pt + Sg	Mt	TS	ST	Md	Fd	2
Family 10	Pt	Fw	-	Sn	T	LT	-	TT	4

According to Table 3.4, the families that have more complex characteristics, in terms of the integration of multiple data sources and the prediction of traffic at the network level, are those that represent traffic conditions in a more realistic way (for instance, families 2, 3 and 9). However, these families have been approached by very few papers, which means that they are still open issues within the transportation literature because of the computational challenges that they bring with them. In contrast, the most straightforward families (numbers 1,5 and 6) are those that contain the highest number of papers together with the more traditional transportation scenarios (point data sources and prediction both at the segment level and at single points on the roads). How ML methods can successfully deal with these families is well documented in the literature but, at the same time, it is still necessary to determine the most suitable ML methods, in terms of computational cost and efficiency, to approach these families of problems.

Finally, the influence of data preprocessing on the families of TF problems is also discussed. Figure 3.14 shows how many papers within every family of

Target: Single (Sn), Multiple (Mt); ***3.1 Input Definition***: Temporal (T), Temporal + Spatial (TS), Non-traffic Data (ND); ***3.2 Time Horizon***: Short-term (ST), Long-term (LT); ***3.3 Time step***: High (Hg), Medium (Md), Small (Sm); ***3.4 Output Definition***: Fundamental (Fd), Travel Time (TT).

problems have included data preprocessing. As shown below, there are 9 families in which there is, at least, one paper that does not incorporate DPP.

Figure 3.14: Number of papers that include data preprocessing techniques within the families of TF problems

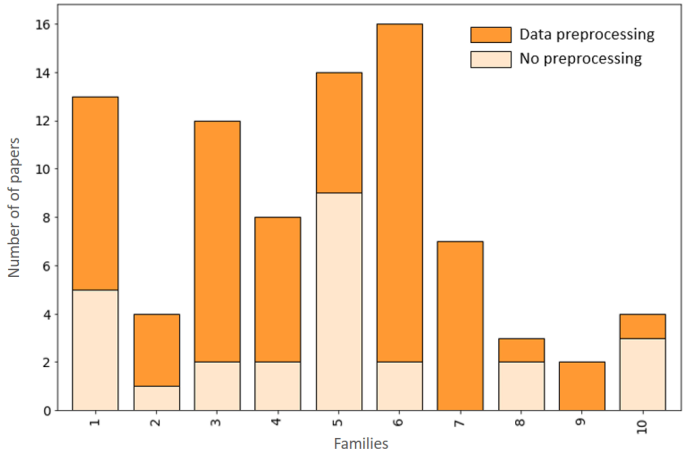
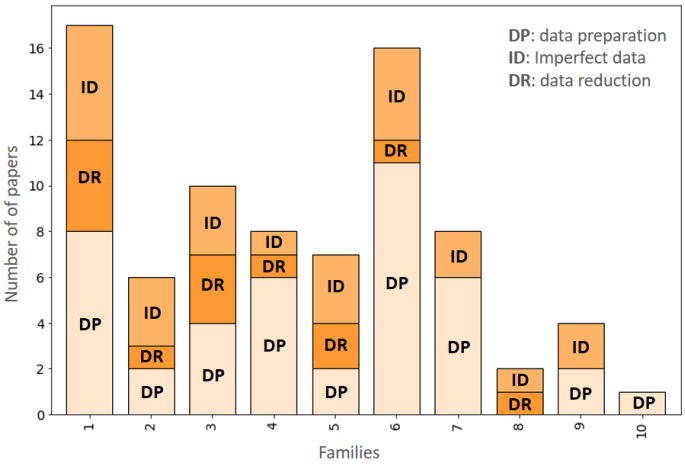


Figure 3.15: DPP approaches within each family of TF problems



More concretely, Figure 3.15 presents how many times the data preprocessing attributes of the taxonomy (Data Preparation - DP, Imperfect Data - ID, Data Reduction - DR) have been used in the studies classified through the ten families. As can be seen, DP is the most common approach followed by ID and DR. In this sense, although the preprocessing of input data is a fundamental stage in the modelling process of ML methods, in the transportation literature it has been scarcely

studied. Therefore, the influence of data preprocessing in the model selection problem, given the characteristics of a particular TF problem, is still an open issue.

3.5 Conclusions

In this chapter, we have proposed a taxonomy to categorise families of TF problems from a supervised learning perspective. Concretely, the taxonomy is built based on two types of attributes to categorise the problems: traffic and modelling specifications. The first includes transportation-related attributes that are the type of data source, the context and spatial coverage of predictions, the input and output variables considered, the time horizon of predictions, and the time step of data. The second set of attributes introduces specification about how the input and output data and the steps of forecasts can be modelled from a supervised learning perspective.

To check the robustness of the taxonomy, we categorised research studies published in the transportation literature from 2000 to 2019. As a result, the taxonomy analysis allowed the extraction of 10 families of TF problems whose complexity change mainly depending on the number and type of data sources used and on the scale of predictions. The families of TF problems most commonly approached are those that incorporate point detectors as the primary data source.

Given the categorisation of TF problems identified, it is possible to point out the lack of guidelines to determine the most suitable approaches to address each problem. It also is essential to highlight the absence of data prepossessing techniques in the modelling process of TF families. The current trend is focused only on the integration of traffic data coming from different data sources and the improvements of incomplete data (missing values and noisy data). The scarce inclusion of data preprocessing techniques in TF is a critical issue as ITSs are continually measuring and storing raw traffic data that needs to be preprocessed before being fed into ML methods. However, being ML knowledge an expensive asset not always affordable, it is hard to work on the synergies between preprocessing techniques and ML algorithms.

Determining the best combination of preprocessing techniques and ML method in the absence of ML knowledge is not an easy task, which implies human effort and times. Therefore, AutoML is a promising strategy to approach the MSP in TF.

It allows us to automatise the complete ML workflow (from data preprocessing to model validation) to reduce human effort and bias at the moment of deciding what model to use to approach a given TF problem while keeping the performance of predictions. Thus, in the following chapter, we present an analysis of the strengths and weaknesses that AutoML has when it deals with different supervised TF problems.

Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less.

Marie Curie

CHAPTER

4

General-purpose AutoML in Traffic Forecasting

4.1 Introduction

As it was seen in the previous chapter, different ML algorithms and preprocessing techniques may be more appropriate for different TF problems. Determining the best pipeline (sequence of data preprocessing techniques and a learning algorithm) for making traffic predictions is not a trivial task, especially when ML expert analyses are not available. As discussed in Sections 2.3.4 and 2.5.3, this challenge is known as the MSP and AutoML has been one of the most successful approaches addressing it so far. AutoML aims at automatically finding competitive combinations of preprocessing techniques, ML algorithm and hyperparameters for given data without being specialised in the domain-knowledge wherein the data comes from (general-purpose) [19]. This automation provides robust automated methods that enable people, with either little or no specialised knowledge, to integrate ML solutions into daily activities of research and business organisations.

Although recent literature [41, 80, 205] reports a variety of AutoML methods, there are only a few that automatise the construction of a complete pipeline. Those methods usually generate ML pipelines using an online search strategy, that is,

a search strategy that takes place after the input dataset has been provided. In some cases, this online search can be purely based on optimisation approaches that test different promising combinations of pipelines structures from a predefined base of preprocessing and ML algorithms to minimise or maximise a predefined performance measure [20, 21].

Alternatively, there are AutoML methods whose online search is complemented with learning strategies [1, 97, 98] typically focused on meta-learning [206]. These techniques first extract meta-features of the input dataset at hand providing information such as the number of instances, features, classes or entropy, among others [98]. From these meta-features, meta-learning identifies good candidates of pipeline structures from a predefined knowledge base that stores meta-features for different datasets and pipelines that are likely to perform well on them. Then, the candidate pipelines are used for a warm-start of the optimisation approach. Besides, other AutoML approaches use the best-performing pipelines found during the search to build an ensemble of models [1, 24]. This ensemble approach has proven to be more robust than only using the best pipeline found via optimisation.

AutoML methods have been successfully used in other areas [41, 205]; however, an extensive analysis to determine the strengths and weaknesses of the search strategies mentioned strategies has not been carried out in diverse learning tasks such as TF. To the best of our knowledge, only one paper out of this thesis has used concepts of AutoML in TF [23]. Vlahogianni et al. [23] proposed a meta-modelling technique that based on surrogate modelling and a genetic algorithm optimises both the algorithm selection and the hyperparameter setting. The AutoML task is performed from a search space base of three ML methods (NN, SVM and radial base function) to forecast average speed in a time horizon of 5 minutes. Although this study showed promising results, the approach used by [23] is an approximation to AutoML that does not consider the current advances of AutoML for ML pipelines. In this context, determining to what extent general-purpose AutoML can be competitive in TF is yet to be fully answered. Therefore, in-depth research efforts are required to investigate whether current AutoML approaches can be competitive against ad hoc methods on supervised learning tasks such as TF.

With these ideas in mind, in this chapter, we present a thorough exploration of the strengths and weaknesses of general-purpose AutoML when dealing with

supervised TF. Concretely, we test two well-established AutoML approaches based on pure optimisation and meta-learning with optimisation in a sample of supervised TF problems extracted from the taxonomy presented in Chapter 3. On the one hand, we test Auto-WEKA (the pioneer AutoML method based on pure optimisation) on a set of TF supervised regression problems characterised by having scales of predictions at a single location and road segment within the freeway and urban environments. We compare this AutoML method versus the general approach in TF that consists of selecting the best of a set of commonly used ML algorithms.

Additionally, we also test Auto-sklearn (the pioneer AutoML method based on meta-learning and optimisation) to approach the same TF problems mentioned above but in this case modelled as supervised classification. We explore the performance of the Auto-sklearn's components through four scenarios: 1) a complete hybrid search strategy that uses its three components (meta-learning, optimisation, ensemble learning); 2) a meta-learning strategy combined with ensemble learning; 3) the knowledge base of the meta-learning component combined with ensemble learning; 4) an approach based on the estimation of the best performing pipeline from the knowledge base of meta-learning.

This chapter is organised as follows. First, Section 4.2 presents the methodology that depicts the experimental framework proposed to explore the performance of general-purpose AutoML in TF. Next, Section 4.3 analyses the main results obtained. Finally, Section 4.4 summarises the main conclusions of the experimentation and this chapter.

4.2 Methodology

This chapter seeks to answer if general-purpose AutoML based on pure optimisation and meta-learning with optimisation can adequately work in supervised TF. To accomplish such purpose, we analyse to what extent Auto-WEKA and Auto-sklearn, two state-of-the-art AutoML methods, can be competitive in TF. In this context, we divided the experimentation into two parts, wherein each of them is devoted to every one of the AutoML methods mentioned. Thus, the following parts of this section are centred in giving more details about the supervised TF problems considered and the experimental set-up.

Supervised Traffic Forecasting Problems

As it was discussed in Chapter 3, there are approximately ten generic families of TF problems. For each of those families, there are plenty of instances in which traffic can be predicted under different transportation circumstances. In this sense, it would be almost impossible to test the performance of AutoML across all the TF families. The latter has a twofold justification. First, data availability is a common issue in data-driven problems, and TF is not excluded from it. There are a few traffic data sources available on the internet with high-quality standards to make predictions, and they represent only a small percentage of the problems that can be categorised by the proposed taxonomy. Secondly, as this is a research thesis, it is needed to narrow the research field wherein knowledge contributions are intended to be done. Such contributions must be conceived under time and resources constraints that make hard to evaluate the performance of AutoML along with a higher number of TF families.

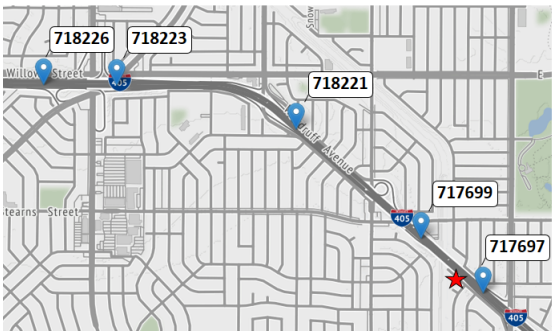
Considering the context above, we decided to approach a sample of two TF supervised families of problems with different instances of them. The first family corresponds to the prediction of traffic at a target location in a freeway environment. In the first place, using only past traffic data of this location (temporal data, T), and then considering historical traffic data coming from the target location and four downstream positions (temporal and spatial data, TS). Besides, in both instances, the input is enriched with calendar data (CD).

The second family type is focused on forecasting traffic within an urban context. Repeatedly, the predictions are made for a single target location considering exclusively historical data of this spot; and on the other hand, taking into account past traffic data of the target location together with four downstream positions. Again, the input data in both instances is complemented with calendar data.

Freeway data is provided by the Caltrans Performance Measurement System [1]. This data is collected in real-time every 30 seconds from nearly 40,000 individual detectors spanning the freeway system across the metropolitan area of California (USA).

[1] http://pems.dot.ca.gov

Figure 4.1: Location of 5 freeway sensors in California State (USA). The detector marked with a ⋆ symbol represents the forecast target location



The route selected for our experiments is the California Interstate I405-S. It is a heavily trafficked freeway by commuters along its entire length [207]. Mainly, we focus on the loop detectors shown in Figure 4.1, wherein the detector marked with a ⋆ symbol represents the forecast target location. The traffic measure collected from the detectors is the speed in an aggregation time of 5 minutes within the time window from March 1, 2019, to April 7, 2019 (38 days of data).

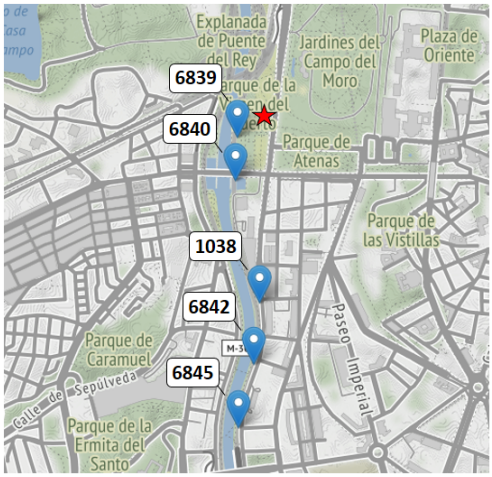
Contrarily, the urban traffic data is obtained from the Madrid Open Data Portal [1]. The Madrid City Council provides through this website access to traffic data around the whole city, publishing 5 - and 15-minutes aggregates of flow, occupancy and speed data in more than 3600 measuring stations.

We chose the M-30 motorway that circles the central districts of Madrid, and that is considered the busiest Spanish road because of its traffic jams. On this route, we focus on the loop detectors depicted in Figure 4.2 where again the ⋆ symbol represents the forecast target location. From them, we extract traffic speed data, in an aggregation time of 15 minutes, for the period between February 2, 2019, and February 28, 2019 (27 days of data).

Based on the raw data presented above, we generate multiple datasets to predict traffic using supervised regression and supervised classification approaches. In the former modelling perspective, the traffic measure to be predicted is averaged speed, a continuous traffic measure. For the latter modelling approach, the predicted traffic

[1] https://datos.madrid.es/portal/site/egob/

Figure 4.2: Location of 5 urban sensors in Madrid city (Spain). The detector marked with a ⋆ symbol represents the forecast target location



measure is LoS that is a categorical variable calculated using traffic speed. The following sections introduce with more details the datasets generated.

1. **Datasets for supervised regression with Auto-WEKA:** For the two families of TF problems described above, we generate 18 datasets in which speed is the traffic measure to be predicted. In the freeway case, the time horizons of predictions are 5, 15, 30, 45, and 60 minutes using data granularity of 5 minutes (granularity means how often the traffic measure is aggregated). Differently, for the urban TF problems, the forecasting time steps are 15, 30, 45, and 60 minutes with data granularity of 15 minutes. To better identify the datasets, they are named following the next structure: *Context_InputData_-Granularity_TimeHorizon*.

   Attributes of freeway datasets where the input is composed of only traffic data from the target location together with calendar data are *Day of the week; Minute of the day; Traffic speed of the objective spot at past 5, 10, 15, 20, 25, 30, 35, 40, and 45 minutes; and Current traffic speed in such point*. In the case of freeway datasets where the input consists of historical speed taken from the target location and four downstream detectors, the attributes are *Day of the week; Minute of the day; Traffic speed of the target position and four*

downstream locations at past 5, 10, 15, and 20 minutes; and Current speed of these five spots.

Attributes of urban datasets in which the input comprises traffic data of the target spot and calendar information are *Day of the week; Minute of the day; Traffic speed of the objective place at past 15, 30, 45, 60, 75, 90, 105, 120, and 135 minutes; and Current traffic speed in this point of interest*. Contrarily, urban datasets with past traffic speed from the target location and four downstream positions in addition to calendar data, have the following attributes: *Day of the week; Minute of the day; Traffic speed of the target position and four downstream locations at 15, 30, 45, and 60 minutes in the past; and Current speed of this five positions*.

2. **Datasets for supervised classification with Auto-sklearn:** In this case, we model the two families of TF problems as supervised classification. Concretely, the objective is to predict a categorical measure named LoS as a multi-class classification problem based on continuous traffic speed. LoS is used to categorise the quality levels of traffic through letters from A to E in a gradual way. Category *A* indicates light to moderate traffic, whereas a category *E* means extended delays [99].

   For the two TF problems described above, we generated 36 datasets (20 for freeway data and 16 for urban data). In the freeway case, the time horizons wherein LoS is predicted are 5, 15, 30, 45, and 60 minutes using data granularity of 5 minutes. Unlike the freeway case, for the urban TF problem, the forecasting time steps are 15, 30, 45, and 60 minutes with data granularity of 15 minutes. To better identify the datasets, they are named following the next structure: *Context_InputData_TimeHorizon*.

   Attributes of the freeway and urban datasets where the input is composed of only temporal traffic data from the target location and calendar data are *Day of the week; Minute of the day; Traffic speed of the objective spot at past 5, 10, 15, 20, 25, 30, 35, 40, 45 minutes for freeway and 15, 30, 45, 60, 75, 90, 105, 120, 135 minutes for urban; and LoS in the target location*. In the case of the freeway and urban datasets where the input consists of historical speed

taken from the target location and four downstream detectors, the attributes are the same mentioned above for the target location and also include traffic speed for the four downstream sites at the same past times.

Table 4.1 presents a summary of the 36 datasets that include the number of instances, the number of attributes, the number of instances per class and the Imbalance Ratios (IRs) of each dataset. The IR is calculated by dividing the number of instances of the majority class over the instances of each of all the other classes. IR values show that the generated datasets have a different imbalance degree. Some datasets do not contain all the possible classes because, on some occasions, some of the classes had a low presence (e.g., 20 samples), which introduces noise in the results. In such cases, the underrepresented classes were tagged as classes of the closest label with the lowest number of samples.

Moreover, the differences between freeway and urban datasets of Type I and Type II are their class distributions. Within each Type, the class distribution is the same for all time horizons. In this sense, we can explore the capacity of Auto-sklearn when approaching different degrees of imbalanced data.

Experimental set-up

In this section, we present the experimental set-up proposed to test the performance of Auto-WEKA and Auto-sklearn. Explicitly, we define a set of performance metrics and statistical tests that are commonly used in the design of experiments in computational intelligence and data mining [55, 208, 209]. We also specify the ETs of the AutoML methods and present the ML competitors that are well-known methods in TF literature [9].

Experimental set-up for Auto-WEKA

- **Metrics:** We evaluated the performance of the AutoML method and the BAs using the metric *Root-Mean-Square Error (RMSE)*. It is applied for regression problems to measure the average magnitude of the error between the predictions of a learning model and the actual values extracted from the raw

Table 4.1: Freeway and urban datasets

Type	Datasets	# Instances	# Attributes	# Instances per class	Imbalance ratios
Type I	Fw_T+CD in time horizons of 5, 15, 30, 45, 60 minutes	[10927 - 9906]	13	A = 4533, B = 3640 C = 893, D = 850	IR (A/D) = 5,07 IR (A/B) = 1,24 IR (A/C) = 5,33
	Fw_TS+CD in time horizons of 5, 15, 30, 45, 60 minutes	[10927 - 9906]	28	A = 5983, B = 4580, C = 363	IR (A/B) = 1,30 IR (A/C) = 16,48
Type II	Fw_T+CD in time horizons of 5, 15, 30, 45, 60 minutes	[10927 - 9906]	13	A = 4533, B = 4023, C = 136	IR (A/B) = 1,12 IR (A/C) = 3,33
	Fw_TS+CD in time horizons of 5, 15, 30, 45, 60 minutes	[10927 - 9906]	28	B = 7782, C = 2125, A = 101	IR (B/C) = 3,66 IR (B/A) = 7,63
Type I	Ub_T+CD in time horizons of 15, 30, 45, 60 minutes	[2684-2634]	13	A = 1337, B = 1188, C = 111	IR (A/B) = 1,12 IR (A/C) = 12,04
	Ub_TS+CD in time horizons of 15, 30, 45, 60 minutes	[2684-2634]	28	B = 1659, A = 691, C = 33	IR (B/A) = 2,40 IR (B/C) = 4
Type II	Ub_T+CD in time horizons of 15, 30, 45, 60 minutes	[2684-2634]	13	A = 1337 B = 1299	IR (A/B) = 1,02
	Ub_TS+CD in time horizons of 15, 30, 45, 60 minutes	[2684-2634]	28	B = 1561 A = 1122	IR (B/A) = 1,39

data [210]. Its calculation is expressed as $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y_i})^2}$ wherein n corresponds to the number of samples in the dataset.

- **Competitors and Baseline:** Here we compare the AutoML method versus the general approach in TF, which consists of selecting the best of a set of commonly used ML algorithms. Concretely, we contrast Auto-WEKA results with four state-of-the-art ML algorithms (NN, SVM, kNN, and RF) in the task of forecasting traffic speed [5, 9]. The latter methods are named Base Algorithms (BAs). For the experimentation with Auto-WEKA, three Execution Times (ETs) were considered: 15, 150, and 300 minutes. They correspond to the time that the method takes to find the best ML algorithm and its hyperparameter configuration for a given dataset. The assumption about ETs is that longer time budgets lead to better results; therefore, ETs under a progressive increase should exemplify this, such as is stated in [20]. Furthermore, five repetitions with different initial seeds were carried out for each ET.

The process of evaluating every BA over a dataset was done with five repetitions with different initial seeds and using their default hyperparameter setting. We have not performed any optimisation or extra-adjustment of the BAs' hyperparameters because we aim to compare the performance of AutoML versus BAs using the same human effort for both of them to make a fairer comparison. Besides, to assess the performance of Auto-WEKA and the BAs, the datasets are partitioned in training (70%) and test (30%).

- **Statistical tests:** We made use of non-parametric statistical tests to assess the differences in the performance of the methods. Two statistical analyses are used following the guidelines proposed in [208]. First, Friedman's test for multiple comparisons is applied to check whether there are differences among the methods. Then, the Holm's test is used to check whether the variations of the Friedman ranking are statistically significant or not.

Experimental set-up for Auto-sklearn

- **Metrics:** To evaluate the experimental set-up presented, we use the metric *G-measure (mGM)* that is applied for multi-class imbalanced data in classification problems [211]. Its calculation is expressed as $mGM = \sqrt[M]{\prod_{i=1}^{M} \text{specificity}_i \cdot \text{recall}_i}$ where M is the total number of classes.

- **Competitors and Baseline:** Unlike Auto-WEKA whose search strategy is only composed of optimisation, in this case, we explore the performance of the components that constitute the hybrid search strategy of Auto-sklearn through the following scenarios.

  - A default scenario in which the search strategy of the AutoML method uses its three components. In this case, we considered three ET for Auto-sklearn (15, 60 and 120 minutes) where meta-learning should enhance the efficiency of Bayesian optimisation by taking less time to find competitive pipelines [1]. For assessing the performance of this scenario, the datasets are partitioned in training (80%) and test (20%).

  - The recommendations done via meta-learning and pipelines of the knowledge base are combined in two ensembles based on weighted-voting without using the optimisation component. First, we extract the 25 ML pipelines (default value used by Auto-sklearn) suggested by the meta-learning part, which then are combined in the weighted-voting ensemble named MetaEns25. To test the latter ensemble, the datasets are partitioned in the same way as was described above. For the second ensemble, we extract the complete list of 121 ML pipelines of the knowledge base of the meta-learning component and choose again 25 best pipelines, based on their validation error, to generate the ensemble MetaEn25-121. In this case, we do the following procedure: the datasets are partitioned in training (60%), validation (20%) and test (20%). To select the 25 best pipelines, the 121 recommendations are trained on the training set, and their performance is assessed on the validation set. Then the ensemble is built with the 25 pipelines that have the best validation error. Finally, the ensemble is trained on training+validation partitions (same number of instances as previous strategies, that is, 80%) and assessed on the test set.
  - We consider the knowledge base of the meta-learning component in isolation. We follow a similar approach to that of MetaEn25-121, that is, we extract the 121 of the knowledge base and split the datasets in training (60%), validation (20%) and test (20%). This means that for every dataset, we train the 121 pipelines of the knowledge base on the training set and based on their validation error, we choose the best pipeline. The latter is then trained on training+validation (that is, over an 80% of the instances) and assessed over the test set.

- **Statistical tests:** We made use of the same non-parametric statistical tests defined for the experimentation with Auto-WEKA.

4.3 Results and Analysis

This section presents the results obtained by Auto-WEKA and Auto-sklearn when making traffic predictions for the families of problems selected.

Auto-WEKA results

This section analyses the results obtained from different angles. Specifically, our aims are:

- To compare the competitiveness and the significance of Auto-WEKA performance with respect to BA competitors in supervised regression problems such as TF.
- To investigate the main benefits and drawbacks of AutoML purely based on optimisation when dealing with supervised regression problems such as TF.

Table 4.2 shows the mean and standard deviation (between brackets) of the $RMSE$ values obtained by both Auto-WEKA and the BAs overall repetitions for each dataset. $RMSE$ values in bold indicate the best result in every dataset achieved by any of the BAs or the Auto-WEKA's execution times.

Observing Table 4.2, we can point out the following results:

- Auto-WEKA performs better than the BAs along with eight datasets. In all the other cases, *RF* or *NN* obtain better results than Auto-WEKA although with small improvements ranging from *0.01* to *1.31* in the $RMSE$ values. These results are interesting because to get the conclusion that *RF* and *NN* are the best BAs in those cases, the transportation user should run all BAs over all datasets and compare their performance among them, which is a time-consuming task. Contrary, running Auto-WEKA only once and therefore employing less human effort, the user can achieve similar or better results than those obtained with the best BAs.
- Regarding datasets characteristics, we can see that they do influence the differences between results of Auto-WEKA and *BAs*. Concretely, for all urban

Table 4.2: Mean $RMSE$ values and their standard deviations (in brackets) obtained by the Auto-WEKA and the BAs

Datasets	Auto-WEKA			Base Algorithms			
	15mET	**150mET**	**300mET**	**kNN**	**NN**	**RF**	**SVM**
Fw_T+CD_5m_5	2.87 (0.08)	2.87 (0.08)	2.91 (0.06)	4.25 (0.14)	2.93 (0.20)	**2.86 (0.06)**	2.90 (0.09)
Fw_T+CD_5m_15	5.81 (0.33)	5.80 (0.28)	5.82 (0.34)	6.66 (0.22)	5.90 (0.45)	**5.16 (0.19)**	5.68 (0.18)
Fw_T+CD_5m_30	7.35 (0.85)	**6.76 (0.41)**	6.99 (0.68)	8.30 (0.39)	9.05 (1.59)	7.06 (0.13)	8.19 (0.23)
Fw_T+CD_5m_45	8.30 (1.09)	7.83 (1.12)	8.53 (0.30)	8.72 (0.20)	10.26 (1.15)	**7.70 (0.25)**	9.65 (0.17)
Fw_T+CD_5m_60	9.12 (1.87)	9.01 (1.67)	9.61 (1.70)	9.01 (0.26)	10.90 (0.74)	**7.99 (0.08)**	10.56 (0.09)
Fw_TS+CD_5m_5	1.19 (0.05)	1.16 (0.01)	1.17 (0.03)	1.46 (0.03)	1.44 (0.29)	**1.13 (0.05)**	1.11 (0.03)
Fw_TS+CD_5m_15	1.92 (0.00)	2.00 (0.47)	2.01 (0.55)	1.78 (0.06)	2.16 (0.24)	**1.64 (0.03)**	1.86 (0.05)
Fw_TS+CD_5m_30	2.12 (0.37)	2.37 (0.47)	**1.90 (0.41)**	1.95 (0.13)	2.60 (0.26)	1.91 (0.08)	2.43 (0.05)
Fw_TS+CD_5m_45	2.50 (0.48)	2.33 (0.49)	2.14 (0.49)	**2.05 (0.09)**	2.92 (0.24)	2.06 (0.07)	2.82 (0.05)
Fw_TS+CD_5m_60	3.17 (0.63)	2.82 (0.69)	2.26 (0.49)	**2.16 (0.09)**	2.89 (0.15)	**2.16 (0.12)**	3.10 (0.11)
Ub_T+CD_15m_15	**5.62 (0.15)**	5.76 (0.26)	5.71 (0.36)	7.74 (0.40)	7.68 (1.27)	5.77 (0.03)	6.05 (0.11)
Ub_T+CD_15m_30	**5.71 (0.29)**	5.97 (0.57)	5.74 (0.35)	8.20 (0.37)	8.02 (1.03)	5.80 (0.23)	6.33 (0.25)
Ub_T+CD_15m_45	5.68 (0.14)	5.73 (0.15)	**5.65 (0.03)**	8.45 (0.20)	8.25 (1.88)	6.16 (0.26)	6.80 (0.37)
Ub_T+CD_15m_60	5.91 (0.12)	**5.85 (0.13)**	5.88 (0.25)	8.52 (0.60)	7.25 (0.70)	5.98 (0.42)	7.05 (0.42)
Ub_TS+CD_15m_15	8.97 (0.46)	8.84 (0.38)	8.83 (0.18)	10.42 (0.72)	14.81 (0.93)	**7.92 (0.30)**	8.45 (0.35)
Ub_TS+CD_15m_30	7.91 (0.23)	7.80 (0.17)	**7.61 (0.23)**	12.95 (0.80)	17.18 (1.82)	9.34 (0.53)	10.75 (0.66)
Ub_TS+CD_15m_45	9.89 (0.18)	9.56 (0.23)	**9.54 (0.24)**	13.96 (0.79)	19.02 (2.94)	9.74 (0.51)	11.53 (0.41)
Ub_TS+CD_15m_60	9.25 (0.09)	9.07 (0.24)	**8.94 (0.11)**	13.07 (0.52)	17.09 (0.96)	9.77 (0.91)	11.94 (0.84)

datasets with a granularity of 15 minutes (except for the dataset *Ub_TS+CD_15m_15*), the AutoML method obtains the best $RMSE$ values. On the other hand, RA works exceptionally well on freeway datasets with the shortest and most prolonged time horizons; the latter excluding *Fw_T+CD_5m_30* and *Fw_TS+CD_5m_30* datasets in which Auto-WEKA gets the best $RMSE$ performance.

- Another interesting aspect is the relation between the execution time and the performance of the models provided by Auto-WEKA. Longer execution times contribute to obtaining better results, mainly, in urban datasets with longer time horizons. In the case of freeway datasets where *RF* and *NN* algorithms are the best ones, the results improve when the Auto-WEKA's execution time increases from 15 to 150 minutes, but they are worse when we pass from 150 to 300 minutes. We observed that this worsening is due to the overfitting produced by the hyperparameters selected by Auto-WEKA. This result indicates that it is necessary to introduce mechanisms in the AutoML method to deal with overfitting, especially when execution times are high.

To assess whether the differences in performance observed in Table 4.2 are significant or not, we made use of non-parametric statistical tests. First, Friedman's test for multiple comparisons has been applied to check whether there are substantial differences among the three execution times of Auto-WEKA. Given that the p-value returned by this test is *0.35*, the null hypothesis cannot be rejected. Therefore, there are no significant statistical differences between the three execution times of Auto-WEKA. This result challenges the assumption of optimisation within AutoML in a sense that longer execution times do not strictly lead to better performance.

Table 4.3: Friedman's average ranking and adjusted p-values for Auto-WEKA ETs

Auto-WEKA mET	**Avg. Ranking**	**Adj. p-values**
300mET	1.8333	-
150mET	1.8611	0.9335
15mET	2.3056	0.3131

To assess if the differences observed between Auto-WEKA and the BAs are significant or not, we also used Friedman's non-parametric test. For this comparison we selected *300mET* as representative approach of Auto-WEKA for being in the first position of the ranking in Table 4.2, although without significant differences against *150mET* and *15mET*.

Considering that the p-value returned by Friedman's test was 0, the null hypothesis can be rejected. The mean ranking returned by the test is displayed in Table 4.4, showing a better global result of *RF* against the others BAs. At the same time, Table 4.4 also exposes the better global results of the AutoML method versus *kNN*, *NN* and *SVM*.

Holm post-hoc test has also been applied using RF as control algorithm (because it was the method with the lowest ranking) to assess the significance of the differences in performance concerning the other methods. Table 4.4 presents the adjusted p-values returned by this test. To highlight significant differences, those p-values lower than 0.05 are shown in bold. Looking at Table 4.4, there are important differences in the test's outcomes. It can be said that RF results improve the rest of BAs significantly, but not the $300mET$ of Auto-WEKA.

Table 4.4: Friedman's average ranking and adjusted p-Values obtained through Holm post-hoc test using RF as control algorithm

Algorithms	Avg. Ranking	Adj. p-values
RF	1.6111	-
Auto-WEKA (300mET)	2	0.4605
SVM	3.0556	**0.0122**
kNN	3.7778	**0.0001**
NN	4.5556	**0**

Auto-sklearn results

This section analyses the results obtained by Auto-sklearn from different perspectives. Specifically, our aims are:

- To compare the competitiveness and the significance of AutoML based on meta-learning and optimisation in supervised classification problems such as TF.
- To investigate the strengths and weaknesses of AutoML based on a search strategy composed of meta-learning and optimisation when dealing with supervised classification problems such as TF.

Table 4.5 shows the mean mGM values obtained by the three ET of Auto-sklearn (AutoS_ET), the two voting ensembles (MetaEns25 and MetaEn25-121) and the best pipeline in the validation stage from the knowledge base of the meta-learning component (BestPipe_Val). mGM values in bold indicate the best result achieved in every dataset. Besides, the last column of Table 4.5 shows what is the winner approach in terms of performance on each dataset.

In the cases wherein the $BestPipe\_Val$ approach obtains the best performance, we indicate the following information in two pairs of brackets. The first pair indicates the ranking position of the winner pipeline (according to the similarity metric used by Auto-sklearn) and the difference between the metric value of this pipeline and the metric value of the pipeline located in the first position of the ranking. The

second tuple between brackets shows the metric values for the pipelines in positions 1, 25 and 121 of the ranking. In the latter case, we can observe whether there is a positive correlation between the ranking positions and the actual performance of the pipelines. The assumption of Auto-sklearn is that pipelines closer to position 1 (distances near 0) are likely to perform better on the input data.

Table 4.5: Mean mGM valuesand their standard deviations (in brackets) obtained by the three Auto-sklearn's ET, two weighted-voting ensembles and the BestPipe_Val approach

Type	Data-set	BestPipe_Val	MetaEn25	MetaEn25-121	AutoS_15ET	AutoS_60ET	AutoS_120ET	Winner
Freeway Type I	T+CD_5	**0.70 (0.00)**	0.67 (0.00)	0.67 (0.00)	0.67 (0.01)	0.68 (0.01)	0.67 (0.01)	(Pipe 5, 2.4) - (0.9, 2.0, 11.6)
	T+CD_15	**0.48 (0.01)**	0.41 (0.01)	0.43 (0.01)	0.32 (0.01)	0.35 (0.02)	0.34 (0.01)	(Pipe 114, 5.2) - (0.9, 2.0, 11.6)
	T+CD_30	**0.29 (0.00)**	0.24 (0.01)	0.25 (0.01)	0.22 (0.01)	0.22 (0.01)	0.22 (0.02)	(Pipe 114, 5.2) - (0.9, 2.0, 11.6)
	T+CD_45	**0.85 (0.01)**	0.17 (0.01)	0.15 (0.00)	0.17 (0.01)	0.16 (0.01)	0.18 (0.00)	(Pipe 67, 2.8) - (0.8, 2.0, 11.6)
	T+CD_60	**0.84 (0.00)**	0.18 (0.01)	0.18 (0.01)	0.19 (0.01)	0.19 (0.00)	0.19 (0.01)	(Pipe 67, 2.8) - (0.8, 2.0, 11.6)
	TS+CD_5	**0.71 (0.00)**	0.70 (0.00)	0.70 (0.00)	0.54 (0.04)	0.60 (0.05)	0.61 (0.08)	(Pipe 72, 2.8) - (1.1, 1.9, 11.1)
	TS+CD_15	**0.50 (0.01)**	0.47 (0.01)	0.48 (0.01)	0.36 (0.03)	0.26 (0.14)	0.31 (0.08)	(Pipe 67, 2.7) - (1.1, 1.9, 11.1)
	TS+CD_30	**0.45 (0.00)**	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	(Pipe 72, 3.2) - (1.6, 1.9, 11.1)
	TS+CD_45	**0.35 (0.00)**	0.00 (0.00)	0.17 (0.30)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	(Pipe 72, 3.2) - (1.6, 1.9, 11.1)
	TS+CD_60	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	-
Freeway Type II	T+CD_5	**0.97 (0.00)**	0.87 (0.00)	0.88 (0.00)	0.87 (0.00)	0.88 (0.00)	0.88 (0.00)	(Pipe 77, 3.0) - (1.2, 1.8, 11.5)
	T+CD_15	**0.96 (0.00)**	0.76 (0.00)	0.76 (0.00)	0.75 (0.01)	0.75 (0.01)	0.76 (0.01)	(Pipe 23, 1-8) - (1.2, 1.8, 11.5)
	T+CD_30	**0.94 (0.00)**	0.66 (0.00)	0.66 (0.00)	0.67 (0.00)	0.67 (0.01)	0.67 (0.01)	(Pipe 58, 2.4) - (1.2, 1.8, 11.5)
	T+CD_45	0.60 (0.00)	0.61 (0.00)	**0.61 (0.00)**	0.60 (0.00)	0.61 (0.00)	0.61 (0.00)	MetaEns25-121
	T+CD_60	0.56 (0.00)	0.57 (0.00)	0.57 (0.00)	**0.58 (0.00)**	0.58 (0.00)	0.57 (0.00)	AutoS_15ET
	TS+CD_5	**0.91 (0.00)**	0.65 (0.00)	0.66 (0.01)	0.66 (0.01)	0.65 (0.02)	0.65 (0.01)	(Pipe 2, 1.0) - (0.9, 1.9, 10.9)
	TS+CD_15	**0.47 (0.00)**	0.41 (0.00)	0.42 (0.00)	0.41 (0.03)	0.42 (0.01)	0.42 (0.02)	(Pipe 41, 2.2) - (0.9, 1.9, 11.0)
	TS+CD_30	**0.46 (0.00)**	0.31 (0.00)	0.34 (0.00)	0.32 (0.02)	0.32 (0.01)	0.31 (0.02)	(Pipe 84, 3.5) - (0.9, 1.9, 11.0)
	TS+CD_45	**0.44 (0.00)**	0.25 (0.00)	0.39 (0.25)	0.27 (0.01)	0.28 (0.03)	0.29 (0.01)	(Pipe 84, 3.4) - (0.9, 1.9, 11.0)
	TS+CD_60	**0.41 (0.00)**	0.22 (0.00)	0.23 (0.01)	0.27 (0.02)	0.25 (0.02)	0.27 (0.02)	(Pipe 83, 3.4) - (0.9, 1.9, 11.0)
Urban Type I	T+CD_15	**0.78 (0.03)**	0.24 (0.00)	0.24 (0.00)	0.32 (0.17)	0.33 (0.11)	0.35 (0.07)	(Pipe 4, 1.3) - (0.8, 2.0, 11.7)
	T+CD_30	**0.70 (0.00)**	0.15 (0.02)	0.20 (0.02)	0.25 (0.09)	0.15 (0.14)	0.12 (0.10)	(Pipe 85, 3.3) - (0.8, 2.0, 11.7)
	T+CD_45	0.71 (0.01)	0.12 (0.07)	0.18 (0.02)	0.30 (0.05)	0.29 (0.06)	0.21 (0.05)	(Pipe 101, 4-1) - (0.9, 2.0, 11.7)
	T+CD_60	**0.40 (0.03)**	0.19 (0.00)	0.21 (0.02)	0.23 (0.06)	0.27 (0.06)	0.26 (0.02)	(Pipe 108, 4.3) - (0.9, 2.0, 11.7)
	TS+CD_15	**0.88 (0.01)**	0.54 (0.00)	0.55 (0.01)	0.56 (0.03)	0.54 (0.01)	0.53 (0.02)	(Pipe 107, 4.3) - (1.1. 1.9, 11.9)
	TS+CD_30	**0.67 (0.01)**	0.50 (0.01)	0.53 (0.02)	0.56 (0.03)	0.58 (0.04)	0.54 (0.05)	(Pipe 94, 3.7) - (1.1, 1.9, 11.8)
	TS+CD_45	**0.69 (0.01)**	0.47 (0.00)	0.51 (0.01)	0.56 (0.02)	0.57 (0.02)	0.56 (0.01)	(Pipe 98, 4.0) - (1.1, 1.9, 11.8)
	TS+CD_60	**0.70 (0.00)**	0.50 (0.00)	0.54 (0.01)	0.57 (0.02)	0.61 (0.04)	0.58 (0.03)	(Pipe 73, 2.8) - (1.1, 1.9, 11.9)
Urban Type II	T+CD_15	**0.91 (0.01)**	0.69 (0.01)	0.69 (0.01)	0.69 (0.00)	0.69 (0.01)	0.68 (0.01)	(Pipe 1, 0.3) - (0.3, 1.3, 12.6)
	T+CD_30	**0.92 (0.01)**	0.66 (0.00)	0.68 (0.01)	0.68 (0.01)	0.68 (0.01)	0.69 (0.01)	(Pipe 48, 2.4) - (0.3, 1.3, 12.6)
	T+CD_45	**0.79 (0.01)**	0.69 (0.00)	0.70 (0.01)	0.69 (0.0)	0.69 (0.01)	0.68 (0.01)	(Pipe 39, 1.9) - (0.3, 1.3, 12.6)
	T+CD_60	**0.89 (0.00)**	0.68 (0.00)	0.68 (0.01)	0.69 (0.013)	0.69 (0.00)	0.70 (0.01)	(Pipe 18, 1.0) - (0.3, 1.3, 12.6)
	TS+CD_15	**0.92 (0.00)**	0.66 (0.00)	0.66 (0.00)	0.66 (0.00)	0.67 (0.01)	0.67 (0.01)	(Pipe 109, 5.0) - (0.5, 1.2, 12.3)
	TS+CD_30	**0.91 (0.00)**	0.64 (0.01)	0.64 (0.01)	0.62 (0.00)	0.64 (0.01)	0.64 (0.01)	(Pipe 109, 5.1) - (0.6, 1.2, 12.4)
	TS+CD_45	**0.90 (0.01)**	0.64 (0.01)	0.63 (0.01)	0.62 (0.01)	0.62 (0.15)	0.64 (0.01)	(Pipe 109, 5.1) - (0.6, 1.2, 12.4)
	TS+CD_60	**0.92 (0.00)**	0.67 (0.01)	0.70 (0.01)	0.63 (0.01)	0.65 (0.03)	0.68 (0.02)	(Pipe 2, 0.6) - (0.6, 1.2, 12.4)

Observing Table 4.5, we can point out the following results:

- The BestPipe_Val component is by far the best performing approach when making traffic predictions. Concretely, it can suggest the best pipeline in 33

out of 36 datasets, performing even better than the longer ET (120 minutes) of Auto-sklearn when it considers its complete search strategy.

- If we check carefully the winner pipelines in the last column of Table 4.5, only in 5 cases (datasets: *Fw_TS+CD_5 - Type II*, *Ub_T+CD_15 - Type I*, *Ub_T+CD_15 - Type II*, *Ub_T+CD_60 - Type II*, *Ub_TS+CD_60 - Type II*) the pipelines are located in a position lower than 25. As it was stated before, 25 is the default value that Auto-sklearn uses to recommend the 25 pipelines that are more likely to perform well on the input data. Such recommendation is made by the similarity metric that compares the meta-features of the input datasets against the meta-features stored in the meta-knowledge base. Considering such comparison, the similarity metric chooses the best pipelines, found in the offline Auto-sklearn's phase, for the 25 most similar datasets with respect to the input data.

- The meta-features used for the comparison are not working correctly, and they are providing information to the similarity metric that makes it leaving out competitive pipelines located beyond position 25. In conclusion, the majority of pipelines in the column winner of Table 4.5 are associated to datasets, that with the current Auto-sklearn's meta-features comparison, are no being categorised as similar with regard to the TF datasets.

- For the default scenario in which Auto-sklearn uses its three components, longer ET are supposed to improve the final results of predictions. However, the improvements only rank approximately from *0.01* to *0.07* in the best of the cases (e.g., *Fw_TS+CD_5 - Type I*). This could be because the meta-learning component is suggesting low-performance pipelines for the warm-start process of the optimisation component. Opposite to this tendency are datasets *Fw_TS+CD_15 - Type I*, *Fw_T+CD_15 - Type II* and *Ub_TS+CD_15 - Type I* wherein the best mGM value is found by ET shorter than 120 minutes. We observed that this worsening is due to the overfitting produced by the hyperparameters tuning of Auto-sklearn on the recommended pipelines. Therefore, it is also necessary to introduce mechanisms in the hybrid search

strategy of AutoML to deal with overfitting, especially when execution times of the optimisation are high.

- Regarding the performance of the two ensembles approaches based on weighted-voting (MetaEns25 and MetaEns25-121), the results of MetaEns25-121 are quite similar with respecto to the results obtained when the optimisation component is taken into account. Concretely, in datasets of *freeway Types I-II* and *urban Type II*, MetaEns25-121 outperforms Auto-sklearn in multiple cases. Particularly, in datasets *Fw_TS+CD_45 - Type II*, *Fw_TS+CD_45- Type I* and *Fw_TS+CD_5 - Type I* the performance of MetaEns25-121 is better than any of the Auto-sklearn's ET. This can be explained because this ensemble is built using already optimised pipelines located beyond position 25 of the ranking. As it was stated before, in those positions are competitive pipelines whose performance is boosted by the ensemble without the need of doing optimisation.

- In the case of datasets *freeway Types I-II* and *urban Type I*, as the time horizon of predictions increases, the performance of all approaches decreases. For these datasets, the ones that have a time horizon of five minutes are the TF problems in which the six methods perform better. Besides, in datasets *Fw_TS+CD_30* and *Fw_TS+CD_60*, almost all Auto-sklearn components have problems predicting the minorities classes, and therefore their mGM values in these cases are equal to zero.

- Regarding urban datasets of *Type I* with only temporal traffic data (T), the three ET of Auto-sklearn and the two ensembles have the lowest performance. This is because these datasets have the highest IRs (IR(A/B) = 1.12 IR(A/C) = 12.04). The latter demonstrates that Auto-sklearn does not incorporate in its inner structure mechanisms to deal with high imbalanced classification datasets. Meanwhile, in the case of urban datasets of *Type I with* spatial and temporal data (TS) and all urban datasets of *Type II*, the performance of the six approaches is quite acceptable and homogeneous across them. This behaviour can be argued as these 12 datasets are the most bal-

Table 4.6: Execution times in minutes of the BestPipe_Val approach and the two weighted voting ensembles. Values in bold indicates an execution time that is between 60 and 120 minutes which are the two longer execution times of Auto-sklearn

Type	Datasets	BestPipe_Val	MetaEns25	MetaEns25-121	Type	BestPipe_Val	MetaEns25	MetaEns25-121
Freeway Type I	T+CD_5	9	1	6	Freeway Type II	9	1	5
	T+CD_15	**90**	25	76		**80**	17	52
	T+CD_30	**86**	26	77		**83**	18	45
	T+CD_45	8	2	10		8	1	9
	T+CD_60	8	1	10		8	1	6
	TS+CD_5	33	4	22		**129**	25	22
	TS+CD_15	38	4	23		31	8	23
	TS+CD_30	36	7	30		35	5	38
	TS+CD_45	39	7	23		37	6	28
	TS+CD_60	40	7	22		37	5	5
Urban Type I	T+CD_15	11	7	13	Urban Type II	10	2	7
	T+CD_30	15	7	12		7	3	7
	T+CD_45	16	2	9		9	2	6
	T+CD_60	16	2	12		7	2	6
	TS+CD_15	46	7	43		38	15	20
	TS+CD_30	46	7	43		40	16	24
	TS+CD_45	47	7	30		38	15	27
	TS+CD_60	46	7	28		35	17	22

anced of the 36 datasets (IR(B/A) = 2.40, IR(B/C) = 4.98; IR(A/B) = 1.02, IR(B/A) = 1.39).

As the computational cost is a crucial factor in AutoML, Table 4.6 shows the execution times in minutes that the BestPipe_Val and the two meta-ensembles took to make predictions on every dataset. As can be seen, in the majority of the cases, the three approaches spent less than 60 minutes, which is the second longer ET of Auto-sklearn considering its three components.

Lastly, to evaluate if the differences in performance observed in Table 4.5 are significant or not, we used the same non-parametric statistical tests considered for Auto-sklearn results. In this case, the tests are not carried out for the 36 datasets at the same. Instead, the datasets are grouped according to their types; this means that freeway and urban datasets of Type I are considered together and the same occurs for datasets of Type II.

First, Friedman's test was applied to check whether there are significant differences among the three execution times of Auto-sklearn. Given that the p-values returned by this test are *0.5134* and *0.2391* for datasets of Type I and Type II, re-

spectively, the null hypothesis cannot be rejected in any of the two cases. In this sense, there is no substantial evidence to corroborate that longer ET for the optimisation component integrated with meta-learning can guarantee higher performance.

Table 4.7: Friedman's average ranking for Auto-sklearn ET and adjusted p-Values obtained through Holm post-hoc test using AutoS_60ET and AutoS_120ET as control algorithms

Fw and Ub datasets Type I		**Fw and Ub datasets Type II**	
Auto-sklearn ET	***Avg. Ranking***	***Auto-sklearn ET***	***Avg. Ranking***
AutoS_60ET	1.7778	AutoS_120ET	1.75
AutoS_15ET	2.1111	AutoS_60ET	1.9444
AutoS_120ET	2.1111	Auto_15ET	2.3056

Next, we evaluate whether the differences in performance of Auto-sklearn and the other approaches in Table 4.5. For this comparison we choose AutoS_60ET and AutoS_120ET as representative approaches of Auto-sklearn in datasets of *Type I* and *Type II*, respectively. This decision is based on the fact that, although without significance differences, they are the approaches in the first positions of Friedman's rankings in Table 4.7.

Table 4.8: Friedman's average ranking and Adjusted p-Values obtained through Holm post-hoc test using RF as control algorithm

Fw and Ub datasets Type I			**Fw and Ub datasets Type II**		
Method	***Avg. Ranking***	***Adj. p-values***	***Method***	***Avg. Ranking***	***Adj. p-values***
BestPipe_Val	1.0833	-	BestPipe_Val	1.3333	-
MetaEns25-121	2.75	**0.0001**	MetaEns25-121	2.6667	**0.0031**
AutoS_60ET	2.8333	**0.000095**	AutoS_120ET	2.6944	**0.0031**
MetaEns25	3.3333	**0.000001**	MetaEns25	3.3056	**0.000014**

Considering the p-values returned by this test are 0.000001 and 0.000053 for datasets of Type I and Type II, respectively, the null hypothesis can be rejected in both cases. The mean rankings returned by the test are displayed in Table 4.8 and p-values lower than 0.05 are shown in bold. It can be confirmed $BestPipe\_Val$ as the best approach against the two ensembles and the best Auto-sklearn ETs.

4.4 Conclusions

In this chapter, we focused on deepening into the strengths and drawbacks that general-purpose AutoML faces when dealing with supervised learning problems such as TF. To accomplish such purpose, we have done two independent studies of AutoML in regression and classification TF problems. First, we test Auto-WEKA (AutoML method based on a pure optimisation search strategy) on TF supervised regression problems wherein the target was predicting traffic speed. Later on, we assessed Auto-sklearn (AutoML method that uses a meta-learning and optimisation) in TF supervised classification problem to forecast LoS. For both cases, the data used was traffic speed collected by loop detectors within urban and freeway environments. Besides, the scales of predictions were focused on point and road segment levels along multiple time horizons.

Regarding a general comparison among Auto-WEKA and Auto-sklearn, both AutoML methods have issues dealing with high-imbalanced TF datasets. Therefore, current analyses and mechanisms already available in ML literature [52, 212] to address imbalanced data still need to be systematised and incorporated into the AutoML workflow. Besides, Auto-sklearn and Auto-WEKA also have difficulties with the time horizon of predictions. As the horizon increases, the performance of the methods tends to fall. The latter is a crucial issue within the transportation domain to be improved because policy-makers require information about the evolution of traffic over the short- and long-term to improve the management of traffic.

Concerning individual analysis carried out for Auto-WEKA and Auto-sklearn, we drew the following conclusions:

- **AutoML based on pure optimisation:** with a lower human effort, the transportation user can expect similar or even better results than the best BA. The latter is interesting because to figure out what the best ML methods are, the user should carry out thorough experimentation to determine the best performing algorithm and the most suitable configuration of hyperparameters. This is achieved using the optimisation search strategy whose focus is on generating, fine-tuning, and evaluating individual pipelines. However, we identified some drawbacks of this approach:

  - the optimisation approach is a demanding process because it involves complex search spaces and, therefore, the evaluation of the objective function usually is computationally expensive.
  - to establish a priori the best time budget for the optimisation process is a difficult task. As it was shown, longer execution times allocated for the search of ML models do not always imply higher final performance in specific supervised learning tasks such as TF. In big or complex datasets collecting good pipelines can require a long time, and the scalability of AutoML could come at a high computational cost. In contrast, in small or more straightforward datasets an excessive time budget is prone to overfitting.

- **AutoML based on meta-learning and optimisation:** although AutoML that combines meta-learning with optimisation can reduce the impact of the issues mentioned above, we identified the following drawbacks:

  - the definition of a set of meta-features able to characterise very diverse datasets and to predict pipelines' performance is a difficult task without substantial evidence to guide this design process. There is a risk that in very different supervised learning tasks to those included in the meta-knowledge base, for instance, TF as in this case, the meta-learning component may suggest pipelines that are not competitive to warm-start the optimisation process. The latter can cause a worsening in performance of the AutoML method.

The in-depth analyses of general-purpose AutoML based on optimisation and meta-learning with optimisation have allowed us to identify a set of strengths and weaknesses of AutoML when dealing with very diverse supervised learning problems such as TF. The knowledge extracted from the experimentation carried out guides us to improve general-purpose AutoML, which can also benefit its performance in supervised TF problems. Thus, in the following chapter, we introduce AutoEn. It is a new and novel AutoML method that overcomes some of the most common drawbacks mentioned above in supervised general-purpose and TF problems.

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Alan Turing

CHAPTER

5

AutoEn: a new AutoML method for supervised learning problems

5.1 Introduction

As it was presented in previous chapters, the core of existing AutoML methods is the optimisation of individual ML pipelines. The optimisation approach could be enhanced with a preliminary stage based on meta-learning. Specifically, the learning approach is in charge of systematically observing how different ML pipelines perform on a wide range of learning tasks to take advantage of this experience. Thus, when new input data comes in, meta-learning recommends competitive pipelines based on previous experience that are used then as a warm-start of the optimisation process.

However, as it was observed in Chapter 4, current AutoML approaches suffer from some issues that motivate the design of new AutoML methods to overcome such limitations. In the first place, longer execution times for the optimisation of pipelines may cause overfitting. Besides, sometimes for medium- and small-size datasets, competitive pipelines can be quickly found without the need of allocating

long time budgets for the optimisation search. Thus, the optimisation of pipelines can be an expensive procedure when the goal is to make good enough predictions. In the second place, as the size of input data grows, it is harder to generate, tune and test multiple pipelines because the evaluation of the objective function becomes expensive. Therefore, the number of candidate pipelines built during the optimisation could decrease, and it ends up affecting the performance of the final solutions.

Finally, the third issue identified in Chapter 4 about current AutoML methods is related with meta-learning. This learning approach has arisen as a promising strategy to deal with the two optimisation issues mentioned above. However, there is a risk that for supervised learning tasks not similar to the ones stored in the meta-knowledge base, meta-learning could recommend pipelines that do not perform as well as expected in specific supervised problems such as TF. Therefore, the optimisation process could be warm-started with pipelines that are not competitive on the input data.

Having in mind the motivations presented above, we want to conceive a competitive and straightforward AutoML method that adjusts to the complexity of the data without the need to define a time budget for its execution. This means that for small- and medium-size datasets, the method can find quick solutions; alternatively, for large datasets, the method may take a longer time to find competitive pipelines. The condition mentioned above could be satisfied not only suggesting single pipelines whose performance could vary drastically from one learning task to another. In this sense, we propose AutoEn, a new AutoML method based on the search and optimisation of ensembles of multi-classifiers that does not commit to a single ML workflow. Thus, we conceive a much simpler AutoML approach that can achieve better or competitive performance in the general-purpose domain as well as in TF.

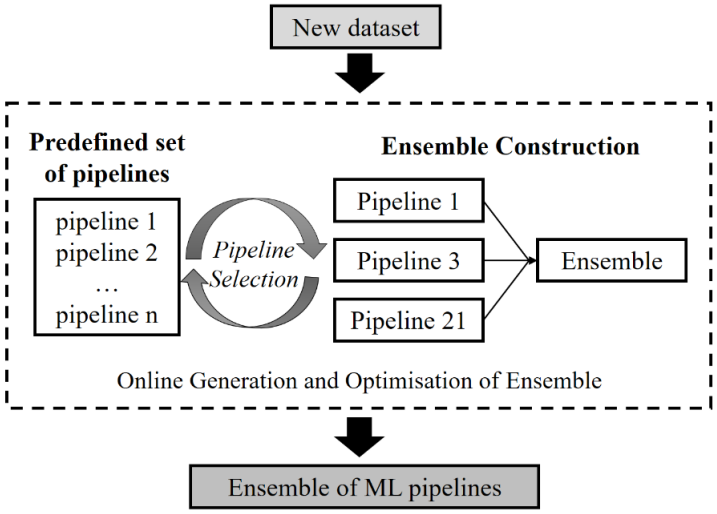
This chapter is organised as follows. First, Section 5.2 introduces AutoEn. Next, Section 5.3 exposes the methodology wherein the the experimental framework is summarised. Next, Section 5.4 analyses the results obtained. Afterwards, Section 5.5 presents a case study where the proposed method is tested on supervised TF problems following experimental set ups previously defined. Finally, conclusions are discussed in Section 5.6.

5.2 Proposed AutoML method

Based on the motivation presented above, in this section, we introduce AutoEn. It is a new AutoML method based on ensembles of pre-defined multi-classifiers composed of a sequence of preprocessing techniques plus one ML classifier.

Figure 5.1 introduces the architecture of AutoEn. This AutoML replaces the online search and optimisation of individual pipelines by the automated generation of ensembles. Similarly to Auto-sklearn, it has in its inner structure a base of diverse pipelines that were trained on different learning tasks during the construction of AutoEn. The pipelines within this base consist of a sequence of data preprocessing technique and a classifier algorithm. The underlying idea of using a set of pre-defined pipelines is offering for every learning task, ML workflows different in their nature that can be less prone to overfitting issues. Differently from Auto-sklearn, we will not use meta-features to decide what pipelines are part or not of the ensemble construction.

Figure 5.1: General workflow of AutoEn that is composed of two main blocks: 1) a set of predefined pipelines trained on different learning tasks and 2) an ensemble component that generates a multi-classifier system, from the collection of pipelines, when a new dataset comes

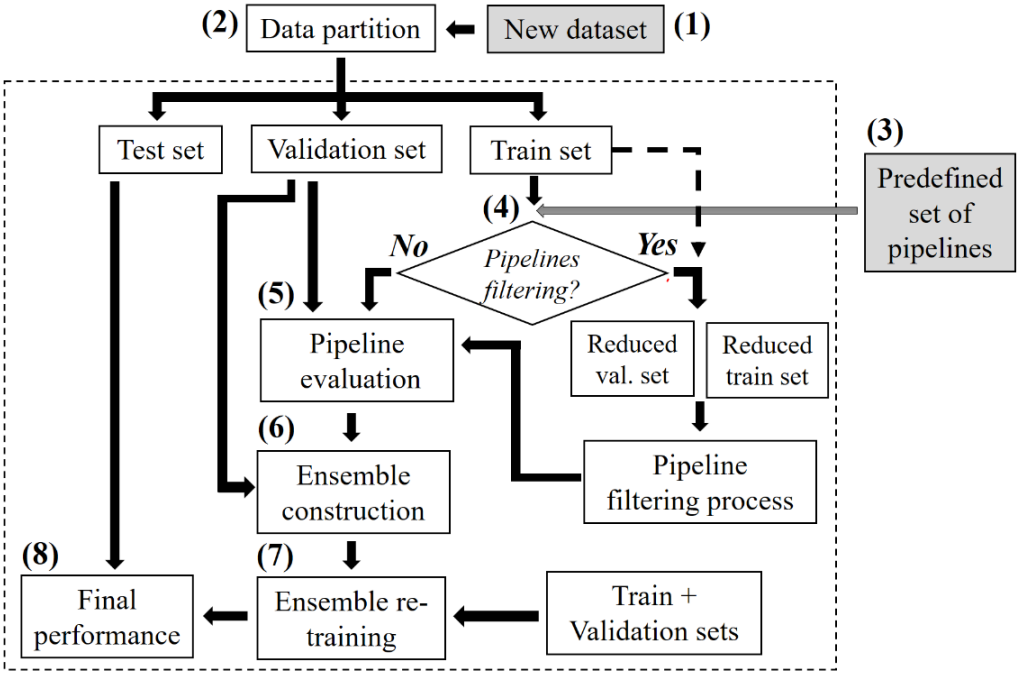


A meta-learning approach should be able to assist us in finding the most promising pipelines to be part of the ensemble. However, we observed in [213] that this

idea can only correctly work under a good enough set of meta-features able to characterise very different datasets, which unfortunately it is not always the case. To avoid relying on meta-features, when new input data comes in AutoEn, it assesses the performance of the base of pipelines on a validation set. Depending on their validation errors, the pipelines can or cannot be part of the ensemble construction. Thus, we avoid the drawbacks of meta-learning when facing very diverse learning tasks.

Details about the automatic selection and construction of the ensemble based on these pipelines are presented in Figure 5.2. In step (1), it receives a new and unseen dataset that later in step (2) is split into train, validation and test sets. Then, in step (3), we retrieve the set of available ML pipelines.

Figure 5.2: Automated selection and construction of the ensemble based on the set of pipelines



Having the list of optimised pipelines at hand, in step (4), it is possible to filter them to improve computational efficiency or not. The latter decision implies either to directly train the complete list of pipelines on the training set (”No” path) or to reduce the list to have a smaller set (”Yes” path) to enhance the computational

efficiency. The former option is the default mode of our AutoML method. At the same time, the "Yes" path implies selecting a sample of the training set (dashed line in Fig. 5.2), which in turn is divided into training and validation set (reduced train and validation set in Fig. 5.2) to filter the pipelines and choose the fastest ones.

After selecting the complete list of pipelines or a reduced version of it, the next step is training the available pipelines in the complete train set and ranking them according to their errors in the validation set (step 5). After that, we proceed to build an ensemble from this set of trained pipelines (step 6). We implemented the ensemble selection approach introduced by [214]. The latter is a greedy procedure that starts from an empty ensemble and then iteratively adds the model that maximises ensemble performance in the same validation set in which the pipelines were previously assessed.

For our AutoEn, we initialise the ensemble with one element, which is the pipeline with the highest performance of the validation set. From the one-element ensemble, we start to append new pipelines, with equal weights, to the one-element ensemble until reaching a predefined ensemble size. It is essential to note the inclusion of new pipelines to the ensemble allows repetitions; this means one high-performance pipeline can appear multiple times in the final ensemble. Finally, in step (7), the pipelines that composed the ensemble are re-trained on training+validation partitions to make classification on the unseen test set. For this final step, the pipelines that appear multiple times in the ensemble are only re-trained once; thus, we can decrease the computational cost of this last step.

5.3 Methodology

We compare the proposed method against the results of AutoML methods reported in the benchmark published by Gijsbers et al. [215]. The latter is an open-source AutoML framework that uses public datasets to conduct a thorough comparison of state-of-the-art AutoML methods.

In this section, we show the elements related to the experimental study. We provide details of the problems chosen for the experimentation. Then, we introduce the measures employed to evaluate the performance of the methods. Finally, we present the methods used for comparison.

Table 5.1: Binary datasets of general-purpose domains

Dataset	Features - Instanances	Numeric Features	Nominal Features	Missing Values
adult	15 - 48842	6	9	6465
amazon_employ	10 - 32769	0	10	0
albert	79 - 425240	26	53	2734000
apsfailure	171 - 76000	170	1	1078695
bank-mark	17 - 45211	7	10	0
blood-transf	5 - 748	4	1	0
christine	1637 - 5418	1599	38	0
credit-g	21 - 1000	7	14	0
higgs	29 - 98050	28	1	9
jasmine	145 - 2984	8	137	0
kc1	22 - 2109	21	1	0
kr-vs-kp	37 - 3196	0	37	0
nomao	119 - 34465	89	30	0
numerai28.6	22 - 92320	21	1	0
phoneme	6 - 5404	5	1	0
sylvine	21 - 5124	20	1	0

Binary and Multi-class supervised problems

We use 16 datasets of binary classification problems and 12 of multi-class problems from the AutoML benchmark [215]. The total of 28 datasets were used in previous AutoML papers [20], AutoML competitions [216] and ML benchmarks [217]. The datasets vary in the number of samples and features by orders of magnitude and vary in the occurrence of numeric features, categorical features and missing values. Table 5.1 presents a summary of binary datasets used for the experimentation. It shows the number of features and instances, the number of numeric and nominal features, and the number of missing values on each dataset.

Table 5.2 introduces an overview of multi-class datasets considered for the experimentation. It summarises the number of features and instances, the number of numeric and nominal features, and the number of classes. In this case, any of these datasets contain missing values.

Table 5.2: Multi-class datasets of general-purpose domains

Dataset	Features - Instances	Numeric Features	Nominal Features	Classes
car	7 - 1728	0	7	4
cnae-9	857 - 1080	856	0	9
connect-4	43 - 67557	0	43	3
dilbert	2001 - 10000	2000	1	5
fashion-mnist	785 - 70000	784	1	10
jannis	55 - 83733	54	1	4
jungle	7 - 44819	6	1	3
mfeat-factors	217 - 2000	216	1	10
segment	20 - 2310	19	1	7
shuttle	10 - 58000	9	1	7
vehicle	19 - 846	18	1	4
volkert	181 - 58310	180	1	10

Performance measures

In this section, we present the metrics used to measure the performance of methods in binary and multi-class problems, specifications about how we measure the computational time of AutoEn, and the statistical test considered to make the comparison between the performance of the methods.

- **Metrics:** We follow the same experimental set-up proposed in the AutoML benchmark to make fair comparisons, and therefore, we use the same performance measures. In particular, the area under the receiver operator curve (roc_auc_score) is used for binary classification problems, and log loss_score is used for multi-class problems. Besides, both measures are averages of ten-fold cross-validation.

- **Computational time:** To measure how much time AutEn takes since new input data comes in until it makes final predictions, we estimate the execution time in seconds extracted from the system clock. Concretely, we start to measure this time when AutoEn receives the input data until it outputs the final classification of the ensemble built during the online search of the method.

- **Statistical tests:** We use non-parametric statistical tests to assess the differences in the performance of the methods [208]. First, Friedman's test for multiple comparisons is applied to check whether there are differences among the methods. Then, the Holm's test is used to check whether the variations of the Friedman ranking are statistically significant or not.

Competitors and baselines

For the experimentation carried out in this chapter, AutoEn has used the pipelines list stored in Auto-sklearn's knowledge base. These pipelines were generated and tunes employing sequential model-based optimisation [91] using a search space of 15 classifiers and 18 preprocessing techniques, all of them implemented in scikit-learn ML library. The classifier can be categorised in linear models, support vector machines, discriminant analysis, nearest neighbours, naive Bayes, decision trees and ensembles. Additionally, data preprocessing techniques include rescaling, imputation of missing values, one-hot encoding, feature selection, kernel approximation, feature clustering and polynomial feature expansion. The interested reader can consult [218] to know more details about these classifiers and preprocessing techniques.

The referenced AutoML benchmark makes comparisons of 4 AutoML state-of-the-art methods: H2O, AutoWEKA, Auto-sklearn and TPOT. In this chapter, we make comparisons against the same AutoML method used in Chapter 4 for supervised TF problems: Auto-WEKA and Auto-sklearn. Additionally, we include the same baselines methods of the referenced AutoML benchmark. They are a constant predictor, which always predicts the class prior (CnstPrd), an untuned Random Forest (RF), and a tuned Random Forest (tuned_RF). We also include the winner approach (BestV_ML) of the experimentation carried out with Auto-sklearn in Chapter 4.

Auto-sklearn [1] and Auto-WEKA [20] were deployed with their default hyperparameter values and search spaces, since most users will use them in this way, and their time budget per fold was 1 and 4 hours. RF uses scikit-learn 0.20 default hyperparameters, and tuned_RF is built with 2000 estimators. For AutoEn and AutoEn_economy (AutoEn_ec), their main parameters are shown in Table 5.3.

Table 5.3: Initial hyperparameters of AutoEn for its two operative modes

Hyperparameters	AutoEn	AutoEn_ec
Ensemble size	50	50
Data partition per fold	train: 60%, validation: 20%, test: 20%	train: 60%, validation: 20%, test: 20%
Data partition to filter pipelines	-	10% of the original train set
Time left for a pipeline to be trained in the pipelines filtering phase	-	36 seconds

5.4 Results and Analysis

This section analyses the results obtained from different angles. Specifically, our aims are:

- To compare the competitiveness and the significance of AutoEn performance with respect to AutoML competitors and baselines in binary and multi-class learning tasks.
- To investigate the main benefits of an AutoML method without allocating any time budget when dealing with classification datasets of different sizes.
- To contrast the differences in performance and runtime between AutoEn and its economy mode.

Table 5.4 shows the mean roc_auc_score and log_loss_score values of our AutoML method in its default (AutoEn) and economy modes (AutoEn_ec), the AutoML competitors (AutoSkl_1h, AutoSkl_4h, AutoW_1h, AutoW_4h) and the baseline methods on the test phase. The best result for each dataset is highlighted in bold-face. Note that for binary datasets, the higher the performance value (roc_-auc), the better, while for multi-class datasets, the lower the loss, the better.

Observing Table 5.4, we can point out the following:

Table 5.4: Mean roc_auc (binary problems) and log_loss (multi-class problems) values obtained by AutoEn, AutoEn_ec, the AutoML competitors and the baseline methods. Values highlighted in bold are the highest performance obtained by any of the methods on every dataset.

Type	Dataset	AutoSkl_1h	AutoSkl_4h	AutoW_1h	AutoW_4h	ConstPrd	RF	tuned_RF	BestV_ML	AutoEn	AutoEn_ec
Binary	adult	**0.930**	**0.930**	0.908	0.909	0.500	0.909	0.909	0.917	0.920	0.920
	amazon_employee	0.856	0.849	0.809	0.820	0.500	**0.864**	0.863	0.859	**0.864**	0.863
	albert	**0.748**	**0.748**	0.724	0.724	0.500	0.738	0.738	0.739	0.702	0.704
	apsfailure	0.991	**0.992**	0.965	0.984	0.500	0.991	0.991	0.990	0.991	0.989
	bank-marketing	**0.937**	**0.937**	0.827	0.909	0.500	0.931	0.931	0.931	0.934	0.935
	blood-transfusion	0.757	**0.763**	0.741	0.742	0.500	0.686	0.689	0.730	0.741	0.735
	christine	0.830	**0.831**	0.802	0.809	0.500	0.806	0.810	0.816	0.825	0.811
	credit-g	0.783	0.782	0.753	0.744	0.500	0.795	**0.796**	0.781	0.786	0.795
	higgs	0.793	**0.809**	0.677	0.757	0.500	0.803	0.803	0.798	0.807	0.807
	jasmine	0.884	0.883	0.861	0.865	0.500	0.888	**0.889**	0.878	0.881	0.883
	kc1	0.843	0.839	0.814	0.818	0.500	0.836	0.842	0.840	**0.852**	0.844
	kr-vs-kp	**1.000**	**1.000**	0.976	0.979	0.500	0.999	**1.000**	**1.000**	0.998	0.998
	nomao	**0.996**	**0.996**	0.984	0.982	0.500	0.995	0.995	0.995	**0.996**	0.969
	numerai28.6	0.529	0.530	0.520	0.528	0.500	0.520	0.521	0.529	**0.532**	0.530
	phoneme	0.963	0.962	0.957	0.965	0.500	0.965	0.966	0.970	**0.972**	0.970
	sylvine	0.990	**0.991**	0.975	0.977	0.500	0.983	0.984	0.989	0.989	0.990
Multi-class	car	**0.010**	**0.010**	0.243	0.122	0.836	0.144	0.047	0.105	0.065	0.070
	cnae-9	0.171	**0.168**	0.873	1.173	2.197	0.301	0.297	0.205	0.178	0.182
	connect-4	0.426	**0.387**	0.741	1.427	0.845	0.495	0.478	0.483	0.460	0.461
	dilbert	0.097	0.063	1.787	1.009	1.609	0.328	0.329	**0.033**	**0.033**	0.034
	fashion-mnist	0.354	0.358	0.581	0.902	2.303	0.361	0.362	0.345	**0.314**	0.315
	jannis	0.705	**0.685**	6.271	1.885	1.109	0.728	0.729	0.710	0.702	0.701
	jungle	0.234	0.223	1.559	2.695	0.935	0.438	0.402	0.216	**0.215**	**0.215**
	mfeat-factors	0.099	0.093	0.627	0.656	2.303	0.234	0.201	0.160	0.093	**0.088**
	segment	**0.060**	0.063	0.501	0.427	1.946	0.084	0.069	0.074	0.061	0.069
	shuttle	0.001	**0.000**	0.015	0.015	0.666	0.001	0.001	**0.000**	**0.000**	0.001
	vehicle	0.395	0.379	2.105	5.560	1.386	0.497	0.486	0.352	0.341	**0.332**
	volkert	0.945	0.925	1.110	8.329	2.053	0.980	0.979	0.925	**0.858**	**0.858**

- As general overview, in binary datasets there are ties of at least 2 methods in 5 datasets: *adult*, *albert*, *bank_marketing*, *kr-vs-kp*, and *nomao*. With respect to multi-class datasets, there are 5 ties of at least 2 methods in *car*, *dilbert*, *dilbert*, *jungle* and *volkert* datasets. Summarising , there is no AutoML method that consistently outperforms all AutoML competitors on binary and multi-class; and AutoML scores are relatively close to the performance of tuned_RF. Additionally, any of the AutoML methods outperforms the tuned_RF on the complete list of datasets.

- Another interesting aspect of results in Table 5.4 is Auto-sklearn and Auto-WEKA results are quite similar under the time budgets of 1 and 4 hours per fold. The latter means that the optimisation process carries out by those 2 methods only brings slight score improvements. Particularly for binary-

datasets, Auto-sklearn only achieves improvements with a longer execution time in *apsfailure*, *blood-transfusion*, *christine*, *numerai28.6* and *sylvine*. Those improvements range from 0.001 to 0.007. In *amazon_employee* , *credit-g*, *jasmine*, *kc-1* and *phoneme*, Auto-sklearn obtains worse performance with 4 hours execution time than the performance it obtains with only 1 hour. For the case of multi-class datasets, although the improvements are greater, they do not reach at least 10% of enhancement; datasets *cnae-9*, *connect-4*, *dilbert*, *jannis*, *vehicle* and *volkert* exemplified the aforementioned situation. As well as binary datasets, sometimes Auto-sklearn performance decreases with longer execution times in *fashion-mnist* and *segment* datasets. Such worsening under longer time budgets allocated for the its optimisation could be due to overfitting issues such as it has been corroborated by previous research [213, 215].

- For the case of Auto-WEKA, the behaviour of improvements from 1 to 4 hours are quite similar with respect to Auto-sklearn. In binary and multi-class datasets there are supervised problems wherein overfitting also seems to affect the performance of Auto-WEKA (*adult*, *albert*, *blood-transfusion*, *jasmine*, *kc1*, *kr-vs-kp*,*mefeat-factors*, *suttle* datasets). In only some datasets there are significant improvements with regard to the shortest execution time (*amazon-employee*, *apsfailure*, *bank-marketing*, and *higgs* dataset). Moreover, there is a considerable worsening in performance when Auto-WEKA receives a longer execution time in some multi-class problems: *cnae-9*, *connect-4*, *jungle*, *vehicle* and *volkert*.

To assess whether the differences in performance observed in Table 5.4 are significant or not, we used non-parametric statistical tests. Two statistical analyses have been applied following the guidelines proposed in [208]. First, Friedman's test for multiple comparisons has been applied to check whether there are significant differences among AutoEn in its two modes, the AutoML competitors and the baseline methods. Then, the Holm post-hoc test has also been applied to assess the significance of the differences in performance.

Table 5.5: Binary datasets: Friedman's average ranking and p-values obtained through Holm post-hoc test using AutoSkl_4h as control method

Methods	Av. Ranking	p-values
AutoSkl_4h	3	-
AutoSkl_1h	3.4062	1
AutoEn	3.5938	1
AutoEn_ec	4.4062	0.4391
tuned_RF	4.5938	0.3990
BestV_ML	5	0.1943
RF	5.4062	0.0776
AutoW_4h	7.25	**0.0008**
AutoW_1h	8.3438	**0**

Table 5.5 exposes the test outcomes for binary datasets and again p-values lower than 0.05 are shown in bold. In this case, AutoSkl_4h is the method in the first position of the ranking. However, it is interesting to note AutoSkl_4h is only statistical better than AutoW_1h and Auto_4h, which is the AutoML with a search strategy only based on optimisation. The rest of the methods, including RF, tuned_TF and BestV_ML, have better performance than the latter AutoML approach.

Table 5.6 summarises test results for multi-class datasets and significant differences with p-values lower than 0.05 are highlighted in bold. AutoEn is the method in the first position of Friedman's average ranking and such as binary problems, there are no significant statistical differences among AutoEn, Auto-sklearn and BestV_ML methods. Moreover, in multi-class datasets, AutoEn is better than Auto-WEKA and the two variants of RF.

From Tables 5.5 and 5.6, it is possible to observe that despite the Friedman test provides a ranking of the methods evaluated, there were no statistical differences among some of them, particularly between the AutoML methods. Therefore, to better assess the performance of AutoEn and AutoEn_ec, we introduce the following analyses.

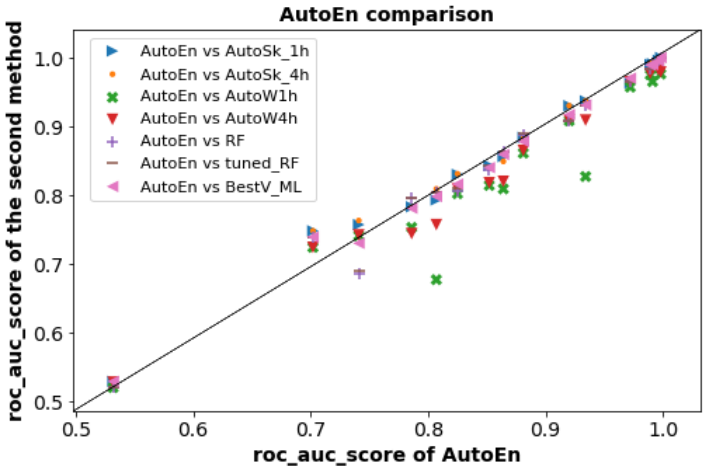
In Figures 5.3 and 5.4, each point compares AutoEn and AutoEn_ec to a second method on binary problems, respectively. In both Figures, the x-axis position of the points is the roc_auc_score of our method on binary datasets. The y-axis position represents the performance metric of the comparison algorithms. Points below the

Table 5.6: Multi-class datasets: Friedman's average ranking and p-values obtained through Holm post-hoc test using AutoEn as control method

Methods	Av. Ranking	p-values
AutoEn	2.25	-
AutoSkl_4h	2.7083	0.6818
AutoEn_ec	2.9167	0.5509
AutoSkl_1h	3.6667	0.2051
BestV_ML	4.0833	0.1010
tuned_RF	5.75	**0.0017**
RF	6.7083	**0.000067**
AutoW_1h	8.375	**0**
AutoW_4h	8.5417	**0**

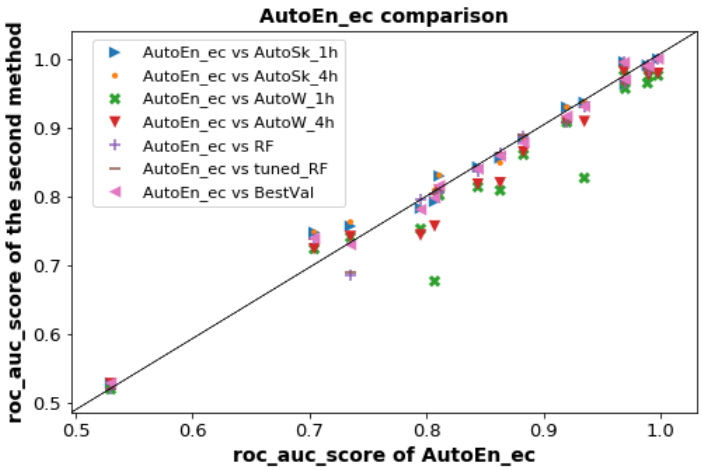
y=x line correspond to datasets for which our method performs better than a second method.

Figure 5.3: ROC-AUC score of AutoEn over 16 binary datasets



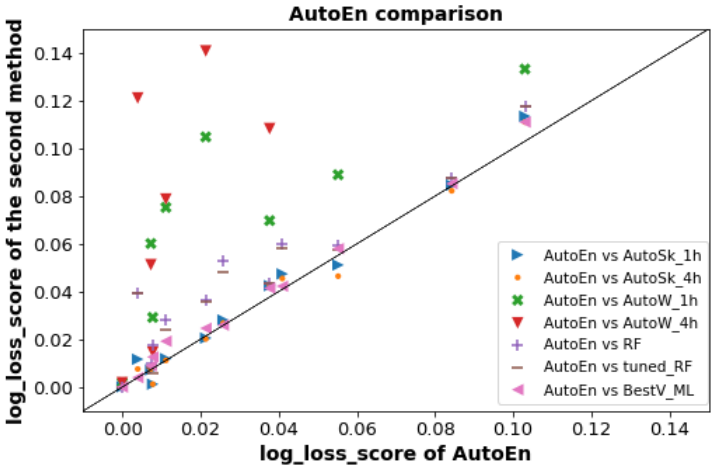
Similarly, Figures 5.5 and 5.6 compare AutoEn and AutoEn_ec to a second method in multi-class datasets. Note that log_loss_score is a minimisation metric; therefore, points above y=x are datasets wherein AutoEn and AutoEn_ec have a higher performance than a second method. For this latter case, the values are normalised between 0 and 1, and outliers were discarded. The reason to normalise the values is that, as was shown in Table 5.4, there are multi-class problems wherein

Figure 5.4: ROC-AUC score of AutoEn_ec over 16 binary datasets



some methods obtained values greater than 1.

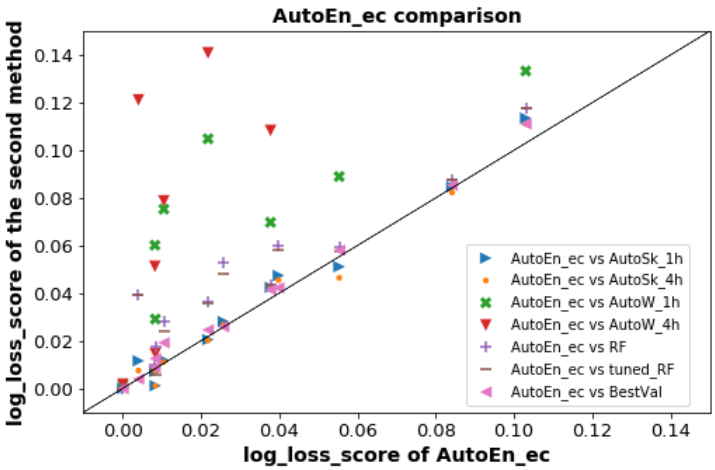
Figure 5.5: Log loss score of AutoEn over 12 multi-class datasets



In a general way, it can be observed for binary datasets, AutoEn and AutoEn_ec have in the majority of the cases similar results to the comparison methods. On the other hand, in multi-class problems, our two approaches have a performance generally better than the other methods.

Finally, as the computational cost is also a relevant factor in AutoML, Figure 5.7 shows the execution time per fold that AutoEn took to make classifications on

Figure 5.6: Log loss score of AutoEn_ec over 12 multi-class datasets

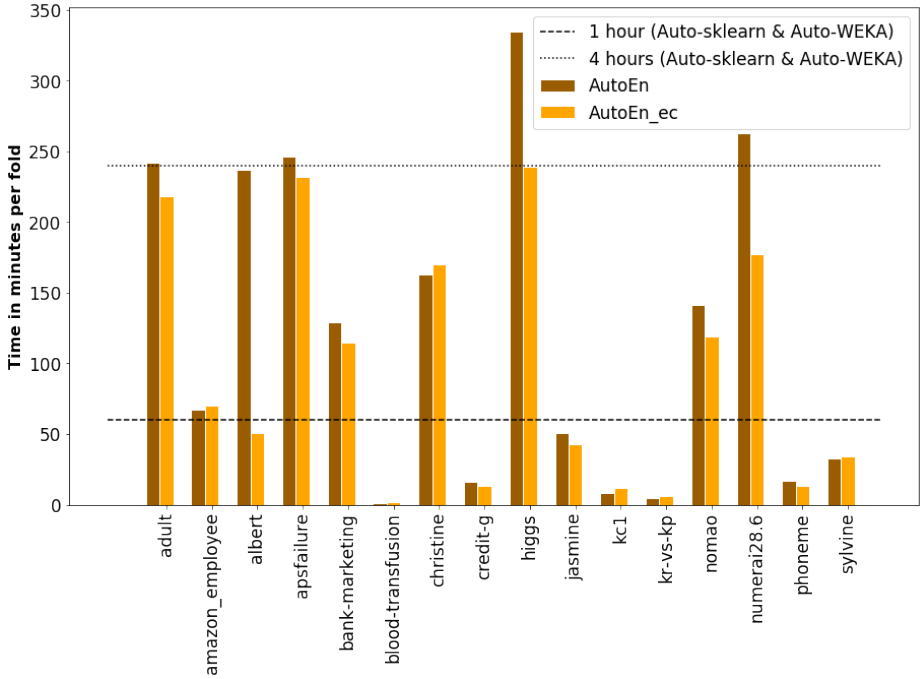


binary datasets. The Figure also plots the execution time of its economy mode. Figure 5.7 also has straight lines that represent the time thresholds of the two optimisation execution times associated with Auto-sklearn and Auto-WEKA. Thus, the purpose is to check whether the execution times of AutoEn can be in-between, below or beyond these two thresholds.

In Figure 5.7, the execution time of AutoEn in its default mode has three different behaviours. First, in *blood-transfusion*, *credit-g*, *jasmine*, *kc1*, *kr-vs-kp*, *phoneme* and *sylvine* datasets, AutoEn spends less than 60 minutes per fold that is the shortest execution time allocated for Auto-sklearn and Auto-WEKA. Secondly, for *adult*, *amazon_employee*, *albert*, *bank-marketing*, *christine* and *nomao* datasets, its execution time is in-between the the two thresholds. Finally, for the remaining datasets (*apsfailure*, *higgs*, *numerai28.6*), AutoEn takes more than 4 hours of time. With resgard to AutoEn_ec, its execution time is below 4 hours in all binary datasets, and as it was observed in Figure 5.4, its performance remains quite similar to AutoEn.

Figure 5.8 summarises AutoEn and AutoEn_ec execution times per fold of in multi-class datasets. Besides, it also shows the 2 time thresholds of the two optimisation execution times carried out by Auto-sklearn and Auto-WEKA. Significant reductions of time can be seen in *dilbert*, *fashion-mnist*, *jannis* and *volkert* datasets, without affecting the performance as was observed in Figure 5.6.

Figure 5.7: AutoEn execution times in binary datasets under its default and economy modes



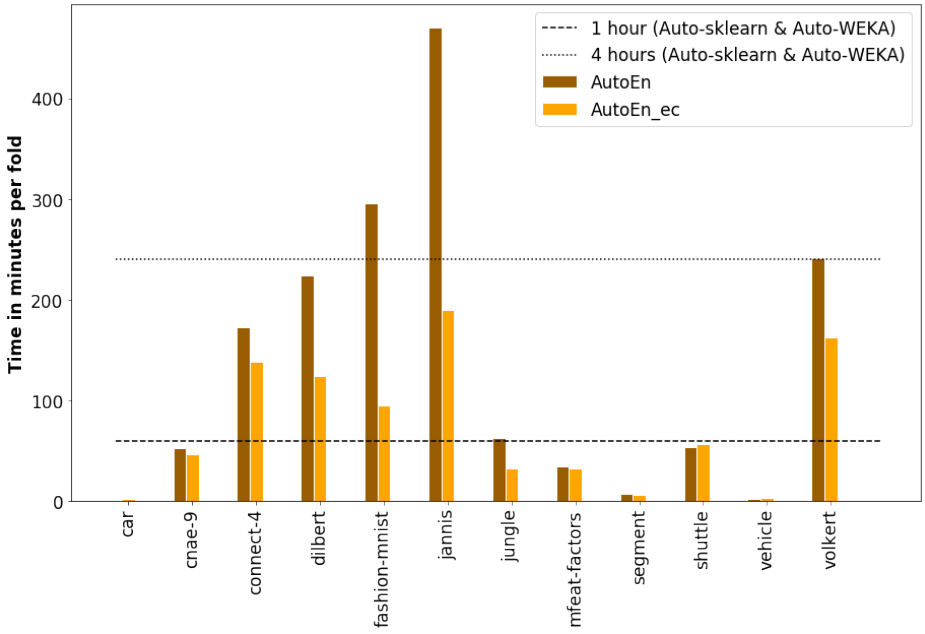
5.5 Case Study: Improving Traffic Forecasting with AutoEn

In this section, we introduce a case study of AutoEn in TF. We aim to demonstrate that this improved general-purpose AutoML approach can enhance the performance of AutoML in the TF domain. To this end, we test AutoEn against Auto-sklearn in various multi-class TF problems that were previously used in Chapter 4. The rest of this section is organised as follow. First, Section 5.5.1 presents the experimental framework of this case study. Then, Section 5.5.2 introduces and analyses the main results obtained.

5.5.1 Experimental framework

1. **Datasets:** For this case study, we approach TF as a supervised classification problem. The objective is to predict LoS. We have used 18 freeway (10) and

Figure 5.8: AutoEn execution times in multi-class datasets under its default and economy modes



urban (8) datasets of Type I used in Chapter 4.

2. **Performance measures**

   - **Metrics:** We use for this case study the same experimental framework for multi-class problems proposed in Section 5.3. This means that we assess the performance of results using loss_score averaged over ten-fold cross-validation per dataset.
   - **Statistical tests:** We also made use of the same two non-parametric statistical tests to assess the differences in the performance of the methods: Friedman's and Holm's tests.

3. **Competitors and baseline:** For the experimentation carried out in the case study, AutoEn used again the pipelines list stored in Auto-sklearn's knowledge base. Besides, we did not consider AutoEn_ec due to the size of the datasets considered. Freeway datasets are around 10.000 instances, and urban datasets have approximately 2.500 instances. These datasets are below

50.000 instances, and as it was observed in Section 5.4, the economy mode of AutoEn has a higher impact on bigger datasets to the ones used in the case study.

In the case of AutoML competitors, we have made comparisons against Auto-sklearn with its default hyperparameter values using three execution times (ET): 15, 60 and 150 minutes. Each ET is considered as individual AutoML competitor. Additionally, as baseline methods, we use a tuned RF built with 2000 estimators and BestV_ML, which was the winner approach in the experimentation carried out with Auto-sklearn in Chapter 4.

5.5.2 Results and Analysis

This section presents and analyses the results obtained for the case study of AutoEn in TF. Specifically, we aim to compare the competitiveness and the significance of AutoEn with regard to a general-purpose AutoML method as Auto-sklearn in supervised TF problems.

Table 5.7 shows the mean log_loss_score values of our AutoML method, the AutoML competitors (AutoSk_15m, AutoSk_60m, AutoSk_150m) and the baseline methods on the test phase. The best result for each dataset is highlighted in boldface, and note that the lower the loss value (log_loss), the better. Observing Table 5.7, we can point out the following:

- As a general overview of results, AutoEn is the most competitive learning approach. This can be seen in the wins distribution as follows: (11) AutoEn, (4) AutoSk_60m, (2) AutoSk_150m, and (1) AutoSk_15m. Moreover, AutoEn and the three ETs of AutoSk are better than RF over all datasets; while they are better than BestV_ML in almost all datasets.

- In freeway datasets, AutoEn is the better approach, especially in time horizons of predictions that range from 30 to 60 minutes in datasets with temporal-spatial (TS) traffic data. On the other hand, AutoEn is the better approach in all datasets that have only temporal (T) traffic data. Besides, AutoSk_15s and AutoSk_60m are more competitive in shorter time horizons (5 and 15 minutes) of freeway datasets with TS traffic data. In this sense, AutoEn achieves

Table 5.7: Mean log_loss (values obtained by AutoEn, the three ET of Auto-sklearn and the two baseline methods. Values highlighted in bold are the highest performance obtained by any of the methods on every dataset

Type	dataset	AutoEn	AutoSk_15m	AutoSk_60m	AutoSk_150m	BestV_ML	RF
Freeway	T_CD_5m	**0.220**	0.229	0.225	0.223	0.244	0.233
	T_CD_15m	**0.340**	0.360	0.346	0.348	0.371	0.366
	T_CD_30m	**0.381**	0.398	0.389	0.391	0.417	0.442
	T_CD_45m	**0.393**	0.438	0.429	0.420	0.439	0.472
	T_CD_60m	**0.401**	0.499	0.454	0.453	0.424	0.489
	TS_CD_5m	0.128	0.117	**0.114**	0.116	0.142	0.133
	TS_CD_15m	0.166	**0.163**	0.164	0.169	0.187	0.185
	TS_CD_30m	**0.186**	0.193	0.192	0.197	0.212	0.218
	TS_CD_45m	**0.180**	0.195	0.182	0.182	0.199	0.217
	TS_CD_60m	**0.182**	0.218	0.197	0.204	0.185	0.229
Urban	T_CD_15m	0.530	0.502	**0.500**	0.503	0.537	0.539
	T_CD_30m	0.538	0.516	0.510	**0.505**	0.536	0.568
	T_CD_45m	0.530	0.513	0.512	**0.495**	0.537	0.571
	T_CD_60m	0.534	0.508	**0.500**	0.508	0.553	0.569
	TS_CD_15m	0.406	0.390	**0.387**	0.389	0.412	0.420
	TS_CD_30m	**0.429**	0.435	0.434	0.437	0.468	0.461
	TS_CD_45m	**0.439**	0.447	0.453	0.452	0.494	0.482
	TS_CD_60m	**0.435**	0.437	0.438	0.437	0.457	0.485
Wins		*11*	*1*	*4*	*2*	*0*	*0*

to offer competitive results over long-term time horizons that were identified as a drawback of other AutoML approaches in Chapter 4.

- In urban datasets, AutoEn is more competitive in datasets that contain temporal-spatial (TS) data. Contrary, its performance is just behind Auto-sklearn in datasets with only temporal (T) traffic data.

- As it was previously discussed in results of Chapter 4, longer ETs allocated for the optimisation do not guarantee better performance. This assumption work properly in some datasets (freeway: *T_CD_5m*, *T_CD_45m*, *T_CD_60m*; urban: *T_CD_30m*, *T_CD_45m*). However, in some other cases, longer ETs end up with similar results to the ones obtained with shorter ETs (freeway: *TS_CD_5m*, *TS_CD_30m*; urban: *T_CD_15m*, *TS_CD_60m*); or in the worst of the cases, they tend to decrease the performance of Auto-sklearn (freeway: *TS_CD_15m*, *TS_CD_30m*; urban: *TS_CD_15m*, *TS_CD_30m*).

To assess whether the differences in performance observed in Table 5.7 are significant or not, we used non-parametric statistical tests. First, Friedman's test for multiple comparisons has been applied to check whether there are substantial differences among AutoEn, the AutoML competitors and the baseline methods. Then, Holm post-hoc test has also been used to assess the significance of the differences in performance.

Table 5.8 exposes the test outcomes and the p-values lower than 0.05 are shown in bold. In this case, AutoEn is the method in the first position of the ranking. However, it is only statistical better than RF and BestV_ML.

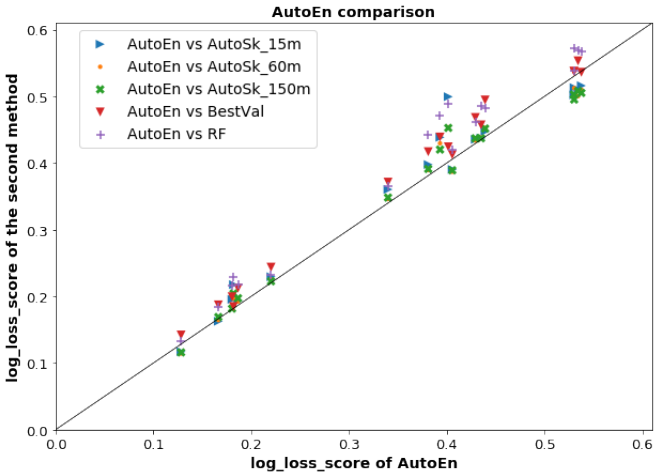
Table 5.8: Friedman's average ranking and p-values obtained through Holm post-hoc test using AutoEn as control method

Methods	Av. Ranking	p-values
AutoEn	2.1667	-
AutoSk_60m	2.3333	0.9520
AutoSk_150m	3.6111	0.9520
AutoSk_15m	3.3333	0.1841
BestV_ML	4.9444	**0**
RF	5.6111	**0**

From Tables 5.8, it is possible to observe that despite the Friedman's test provides a ranking of the methods evaluated, there were no statistical differences among some of them, particularly between the AutoML methods. Therefore, to better assess the performance, we introduce the following analyses of Figure 5.9. Each point compares AutoEn to a second method on the multi-class datasets. The x-axis position of the points is the log_loss_score of our method on, while the y-axis position represents the performance metric of the comparison algorithms. Points above the y=x line correspond to datasets for which AutoEn performs better than a second method. From Figure 5.9 it can be observed that AutoEn has in most of the cases better results than the comparison methods.

In summary, AutoEn can overcome the optimisation and meta-learning issues of AutoML in TF. A simple strategy based on the construction of multi-classifiers systems achieves better results in TF problems without pre-define time budgets for the optimisation process and without being limited by the representativeness of meta-learning. Although AutoSk_60m and AutoSk_150m achieved better results

Figure 5.9: Log loss score over 18 TF datasets. Points *above* the $y = x$ line correspond to datasets for which our method performs better than a comparison algorithm



than AutoEn in some particular cases, non-expert ML user must test Auto-sklearn multiple times using different ETs to find these competitive results.

5.6 Conclusions

In this chapter, we proposed a new AutoML method for supervised problems that has a search strategy supported on the construction of ensembles of multiple classifiers. The AutoML method was tested against Auto-sklearn and Auto-WEKA, two state-of-the-arts AutoML methods, in multiple binary and multi-class supervised problems of a well-established AutoML benchmark. Besides, we test AutoEn in a case study of TF considering diverse multi-class TF problems. The main conclusions drawn from the results are:

- The online phase of AutoML should be focused on building ensembles rather than individual ML pipelines. Thus, computational effort should be directed to generate high-performance pipelines in the offline period of AutoML, which later contribute to producing competitive and fast ensembles during the online stage of AutoML. These ensembles are less prone to overfitting and are not exposed to the meta-learning and meta-features drawbacks. Although this ensemble approach can also be optimised, its hyperparameters

are a reduced set of values. Such characteristic of AutoEn constitutes a much less complicated search space compared against the online optimisation of pipelines with a higher degree of hyperparameters.

- Ensembles of multi-classifiers can be more adaptable to datasets of different sizes, which lead to optimising computational efficiency of AutoML. On the one hand, high-performance solutions can be achieved for small- and medium-size dataset faster than traditional optimisation approaches (e.g., Bayesian optimisation, evolutionary programming). Contrary, for big datasets, although the time consumption could increase, the computation effort of the online phase is focused on finding competitive combinations of multi-classifiers. This approach is highly parallelisable compared to generating and fine-tuning individual pipelines that are even more costly to be evaluated in big datasets.

- AutoEn can be better and more competitive than state-of-the-art AutoML approaches in supervised TF problems. Notably, it is less prone to overfitting than Auto-sklearn because its ensemble strategy makes more robust predictions as multiple classifiers are considered at the same time. Besides, AutoEn can better adapt to different time horizons, specifically, in long-term time horizons that are a crucial element for transportation users when are managing traffic flows.

- Finally, it is relevant to note that in the TF domain, transportation users non-experts in ML could use other AutoML methods focused on optimisation of individual pipelines and achieve satisfactory results. However, as the core of other AutoML strategies is based on optimisation, the user should test multiple times different time budgets to deliver competitive results. This is a task that involves human effort and time costs, which does not have clear guidelines to decide the best ET depending on the size of the dataset at hand. Therefore, AutoEn arises a promising approach that overcomes these limitations while it can supply the demands of non-expert ML users in TF problems.

We can judge our progress by the courage of our questions and the depth of our answers, our willingness to embrace what is true rather than what feels good.

Carl Sagan

CHAPTER

6

Conclusions and Future Work

This final chapter summarises the main contributions, limitations, and future lines of research resulted of this thesis. First, a consolidated view of results and contributions is presented in Section 6.1. Afterward, limitations of these contributions are identified and future research are discussed in Section 6.2. Finally, Section 6.3 gives a list of publications carried out as co-author that derived from the knowledge gained in this research.

6.1 Summary of Contributions

TF, as well as other data-driven fields, is experiencing the generation of significant volumes of complex data. This data availability has promoted the use of ML to approach the prediction of traffic. Concretely, employing ML in TF represents a step forward in the development of more accurate and robust ITSs, as ML can mine intricate traffic patterns embedded in data without knowing traffic mechanisms. However, the success of ML in TF does not come from solely applying ML methods in isolation to make traffic predictions. Instead, complete ML pipelines (the workflow form data preprocessing to model assessment and selection) are required

to exploit raw traffic data fully. Searching for the most suitable data preprocessing approach that better fit the TF problem at hand and the ML method selected improves the performance of ML in TF. However, it is a complex problem that involves human effort and time costs that demand expert ML knowledge to deal with it.

In the context discussed above, AutoML is at the cutting edge of ML research as a valuable approach for fields exposed to the production of vast volumes of data for which a thorough ML expert analysis is scarce and not always affordable such as TF. However, as data production keeps increasing at rates never seen before, current AutoML approaches need to be reformulated to pursue scalability and simplicity while maintaining its competitiveness. The latter will allow AutoML to address the current and upcoming generation of big datasets in diverse specific problem domains (e.g.,. ITSs). This context demands research efforts to develop new and novel AutoML methods that overcome present AutoML drawbacks and take advantage of all available resources (raw data, preprocessing approaches, ML methods) to deal with different data-driven problems more effectively and efficiently.

The research developed in this dissertation has contributed to the improvement of AutoML for supervised learning problems. The experiments have been conducted employing TF as application area, which is a clear example of research fields with large amounts of data wherein expert ML knowledge is not always an affordable asset. A thorough review of the state-of-art about ML, AutoML and TF have been completed. After this, three research stages have been developed following a progressive order, in which early findings take part in the final research outcomes. The main contributions are revisited in the paragraphs presented below.

The first contribution has addressed the development of a novel taxonomy to categorise TF problems from a supervised learning perspective (Chapter 3). It allowed us to understand, organise and systematise the existing knowledge, trends and gaps of TF from a classical supervised learning perspective. The key novelty of the proposed taxonomy lies in its design based on core characteristics that may alter the complexity and the modelling process of TF problems. Based on this categorisation of TF problems, it is possible to identify both the most commonly ML methods used in a diversity of TF problems and the lack of guidelines to determine the most suitable approaches to address each problem. Moreover, the taxonomy

highlights the absence of data prepossessing techniques in the modelling process of TF problems. This is a crucial issue as ITS technologies are continually collecting raw and complex traffic data that needs to be preprocessed before being fed into ML methods. However, without expert ML knowledge available, it is difficult to exploit the synergies between preprocessing approaches and ML methods fully. Finding competitive combinations of preprocessing techniques and ML methods in the absence of ML knowledge is not a trivial task that involves human effort and times cost. Therefore, AutoML appears as a promising approach to solve these issues and play the role of ML experts focusing on generating competitive learning approaches that are able to deal with TF data.

After introducing the mentioned taxonomy and analysing its results, a second contribution has encompassed a comprehensive study of general-purpose AutoML in supervised learning (Chapter 4). We carried out a detailed analysis to figure out the strengths and weaknesses of the most commonly used AutoML approaches in supervised learning tasks. We used TF as an application area not previously addressed by AutoML. To accomplish this aim, we tested the performance of AutoML approaches based on pure optimisation and meta-learning with optimisation in different supervised TF problems. The key novelty of this contribution is an in-depth analysis of how the various components of AutoML (meta-learning, optimisation, ensemble learning) behave in supervised tasks such as TF, which allowed us to identify what are the main contributions of each component into the final performance of predictions. These analyses were not done before and allowed us to identify key points to improve the current state of AutoML in general-purpose domains as well as in specific problems such as TF.

The preceding investigation of AutoML approaches provided substantial evidence that supports the search of alternative strategies for the automatic generation of ML pipelines. Concretely, we realised that current AutoML strategies based on on-line optimisation of pipelines suffer from relevant issues such as overfitting, low scalability, and not a broad enough representativeness of meta-learning in some supervised learning tasks like TF. In this context, the final contribution of this thesis is a novel AutoML approach presented in Chapter 5. We introduce AutoEn, a new AutoML method that works based on the automated generation of ensembles from a predefined set of ML pipelines. This is a more straightforward strategy less prone

to overfitting that also is not exposed to meta-learning drawbacks; therefore, it can be more adaptable to datasets of different sizes and boost the computational efficiency of AutoML. We tested AutoEn against general-purpose AutoML competitors in very diverse binary and multi-class supervised problems, including TF. From our empirical results, AutoEn obtained better or competitive results with respect to the AutoML state-of-the-art approaches. Thus, AutoEn paths the way towards AutoML frameworks purely based on ensemble strategies that are competitive and efficient dealing with different data-driven problems, including TF.

6.2 Limitations and Future Work

In relation to the contributions summarised above, this thesis also presents limitations that need to be mentioned. In this section, we highlight some aspects that would enhance the research conducted and remark ideas of future work that potentially would overcome the limitations identified.

1. The proposed taxonomy to categorise TF problems from a supervised learning perspective (Chapter 3) has allowed identifying well-established trends and gaps of classical ML methods in TF. At the moment, they are the automated learning approaches most commonly used in this data-driven field. However, it is worth noting that ITS technologies are continually evolving and recently, new formats of traffic data are being collected [219]. These new formats correspond to video records and traffic images for which Deep Learning (DL) approaches are more appropriate than classical ML [220, 221]. Therefore, as future work, it would be interesting to address the study of DL methods and new traffic data sources such as the ones mentioned above. Taking as a starting point the proposed taxonomy, the inclusion of DL and video-image data sources could enhance the categorisation power of the taxonomy to cover a broader space of ML methods and TF problems. In this sense, new attributes may be added to the taxonomy as the preprocessing approach used by DL methods is embedded in their inner structure. Besides, new attributes may also be required to categorise the types of network architectures whose internal mechanisms can model input traffic data in different ways.

Additionally, apart from supervised regression and classification that are the most common modelling paradigms used in TF, the prediction of traffic can be also be tackled from other data-driven modelling approaches. Recently, unsupervised learning [74] and reinforcement learning [222] start to be used in the transportation literature [223, 224]. Thus, the consideration of these modelling approaches could potentially lead to new families of TF as these learning paradigms pose modelling constraints that may alter how TF problems are modelled.

2. The thorough study of AutoML, based on optimisation and meta-learning integrated with optimisation, in specific problems such as TF has allowed us to identify the strengths and weaknesses of general-purpose AutoML (Chapter 4). These findings offered insights to propose new AutoML approaches that overcome the most common and relevant issues of this type of automated methods. Therefore, these results are applicable to other specific domains, not limited to TF, wherein AutoML could play the role of ML experts and offers robust learning methods to approach data-driven problems. To accomplish this aim, we carried out the experiments using the two pioneer AutoML methods (Auto-WEKA and Auto-sklearn) that represent the two types of pipeline search strategies entirely: optimisation and meta-learning with optimisation. Currently, there is a bunch of AutoML methods (e.g.,. H2O [24], TPOT [21]) that automatise the complete ML workflow and that can be included in this in-depth analysis of AutoML. Their inclusion could offer further insights about how the general-purpose approach can be adaptable to diverse supervised learning tasks that have not been previously seen by these AutoML methods.

   Furthermore, as it was stated above, new ITS sensors are providing new traffic data formats (e.g.,. video, images). Based on the proposed taxonomy and on the families of TF problems selected to test the general-purpose approach of AutoML, we would like to study the performance of AutoEn and other AutoML methods in different transportation scenarios. Notably, it would be interesting to consider TF problems that include the data sources mentioned.

This research effort would allow mapping what are the best AutoML approaches in a broader set of TF families, which represents a valuable source of information for transportation managers for which expert ML knowledge is not always affordable.

3. The analysis of general-purpose AutoML in TF led us to develop AutoEn, a new AutoML method for supervised learning problems. This automated method has a strategy based on the construction of ensembles of multiple classifiers, which boosts its performance in specific problems such as TF with respect to the state-of-the-art. AutoEn is a more straightforward approach that overcomes the most common drawbacks of AutoML and improves the scalability of these learning approaches. Such as it was stated before, the core AutoEn's search strategy is a base of very diverse and predefined pipelines. Therefore, based on this fixed set of ML workflows, we would like to work on the new generation of AutoML systems. Concretely, we would like to develop AutoML approaches that autonomously are able to decide when to generate new ML workflows using the pre-defined set of pipelines at hand; and, on the other hand, that can recognise when its base of pipelines is enough to address learning tasks similar to the ones in which its pipelines are already high-performance ones. This approach would represent a step forward in the design of more autonomous AutoEn wherein ML experts decisions are organised and systematised to be incorporated within AutoML workflow.

   Finally, AutoEn contains pipelines of classical ML methods as the learning approaches most commonly used in specific problems such as TF. However, the constant generation of new data formats is motivating the use of other learning methods like DL. Drawing on its current design, AutoEn could be further expanded through the inclusion of DL methods within its base of pipelines. To accomplish this aim, our method could be adapted to work and research in AutoDL that is a ML field focused on the automatically generating of network architectures to deal with raw and high-dimension data [225]. Thus, AutoEn could have the opportunity to approach a broader set of

specific domains such as Citizen Science and image classification problems [226, 227].

Summarising, the limitations in the research conducted represent a set of improvements that may suggest immediate lines of work in the areas of AutoML and TF approached from a ML perspective.

6.3 Other Publications

Apart from the scientific dissemination that supports this research, this section presents other publications derived from the knowledge acquired during this PhD thesis. They were made in collaboration with other researchers and are shown below:

1. Title: A Graph CNN-LSTM Neural Network for Short and Long-term Traffic Forecasting based on trajectory data.

Authors: Bogaerts T, Masegosa AD, Angarita-Zapata JS, Onieva E.

Journal: Transportation Research Part C (Impact Factor = 6.077 $\rightarrow$ Q1).

Status: Published. Vol. 112, pp. 62-77, 2019.

2. Title: Nature-Inspired Metaheuristics for optimising Information Dissemination in Vehicular Networks.

Authors: Masegosa AD, Osaba E, Angarita-Zapata JS, Laña I, Del Ser J.

Congress: The Genetic and Evolutionary Computation Conference (GECCO), 2019, Prague (Republic).

3. Title: White-box flight simulator built with system dynamics to support urban transportation decision-making and address induced travel demand.

Authors: Angarita-Zapata JS, Andrade Sosa HH, Masegosa AD.

Journal: Scientia Et Technica Journal.

Status: Published. Vol. 25, pp. 438-447, 2020.

Bibliography

[1] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and Robust Automated Machine Learning," in *Advances in Neural Information Processing Systems*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2962–2970.

[2] A. M. Nagy and V. Simon, "Survey on traffic prediction in smart cities," *Pervasive and Mobile Computing*, vol. 50, pp. 148–163, 2018.

[3] A. Perallos, U. Hernandez-Jayo, E. Onieva, and I. J. García-Zuazola, *Intelligent Transport Systems: Technologies and Applications*, 1st ed. Wiley Publishing, 2015.

[4] F. . Wang, P. B. Mirchandani, and N. Zheng, "Advances and trends in research and development of intelligent transportation systems: An introduction to the special issue," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 222–223, 2004.

[5] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 3–19, 2014.

[6] B. S. Kerner, "The physics of traffic," *Physics World*, vol. 12, no. 8, pp. 25–30, 1999.

[7] M. Karlaftis and E. Vlahogianni, "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 3, pp. 387 – 399, 2011.

[8] P. Lopez-Garcia, E. Onieva, E. Osaba, A. D. Masegosa, and A. Perallos, "A hybrid method for short-term traffic congestion forecasting using genetic algorithms and cross entropy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 557–569, 2016.

[9] I. Laña and J. Del Ser and M. Velez and E. I. Vlahogianni, "Road Traffic Forecasting: Recent Advances and New Challenges," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 2, pp. 93–109, 2018.

[10] J.-F. Chen, S.-K. Lo, and Q. H. Do, "Forecasting Short-Term Traffic Flow by Fuzzy Wavelet Neural Network with Parameters Optimized by Biogeography-Based Optimization Algorithm," in *Computational Intelligence and Neuroscience*, vol. 2018, 2018, pp. 1–13.

[11] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[12] S. Oh, Y.-J. Byon, K. Jang, and H. Yeo, "Short-term Travel-time Prediction on Highway: A Review of the Data-driven Approach," *Transport Reviews*, vol. 35, no. 1, pp. 4–32, 2015.

[13] E. I. Vlahogianni, J. C. Golias, and M. G. Karlaftis, "Short term traffic forecasting: Overview of objectives and methods," *Transport Reviews*, vol. 24, no. 5, pp. 533–557, 2004.

[14] C. P. van Hinsbergen, J. W. van Lint, and F. M. Sanders, "Short Term Traffic Prediction Models," *World Congress on Intelligent Transport Systems*, 2007.

[15] H. van Lint and C. van Hinsbergen, *Transportation research circular*. Transportation Research Board-National Research Council, 2012.

[16] U. Mori, A. Mendiburu, M. Álvarez, and J. A. Lozano, "A review of travel time estimation and forecasting for Advanced Traveller Information Systems," *Transportmetrica A: Transport Science*, vol. 11, no. 2, pp. 119–157, 2015.

[17] S. Garcia, J. Luengo, and F. Herrera, *Data preprocessing in data mining*. *Springer.* Cham, Switzerland, 2015.

[18] I. Triguero, D. García-Gil, J. Maillo, J. Luengo, S. García, and F. Herrera, "Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 2, p. e1289, 2019.

[19] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018.

[20] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA," in *Proceedings of the 19th International conference on Knowledge discovery and data mining*, 2013, pp. 847–855.

[21] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, pp. 485–492.

[22] J. Vanschoren, "Meta-Learning: A Survey," *Computing Research Repository - CoRR*, 2018.

[23] E. I. Vlahogianni, "Optimization of traffic forecasting: Intelligent surrogate modeling," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 14 – 23, 2015.

[24] H2O.ai, *H2O AutoML*, June 2017, h2O version 3.30.0.1. [Online]. Available: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

[25] J. Zhang, F. Wang, K. Wang, W. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624–1639, 2011.

[26] J. M. Sussman, *ITS: A Short History and a Perspective on the Future*. Boston, MA: Springer US, 2005, pp. 3–17.

[27] G. Dimitrakopoulos and P. Demestichas, "Intelligent transportation systems," *IEEE Vehicular Technology Magazine*, vol. 5, no. 1, pp. 77–84, 2010.

[28] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal Recurrent Convolutional Networks for Traffic Prediction in Transportation Networks," *Sensors*, vol. 17, no. 7, 2017.

[29] Z. Liu, J. Guo, J. Cao, Y. Wei, and W. Huang, "A Hybrid Short-term Traffic Flow Forecasting Method Based on Neural Networks Combined with K-Nearest Neighbor," *Journal on Traffic and Transportation Technology*, vol. 30, no. 4, pp. 445–456, 2018.

[30] N. Zarei, M. A. Ghayour, and S. Hashemi, "Road traffic prediction using context-aware random forest based on volatility nature of traffic flows," in *Intelligent Information and Database Systems*, A. Selamat, N. T. Nguyen, and H. Haron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 196–205.

[31] Y. Liu and H. Wu, "Prediction of road traffic congestion based on random forest," in *International Symposium on Computational Intelligence and Design*, vol. 2, 2017, pp. 361–364.

[32] D. Xu, Y. Wang, P. Peng, S. Beilun, Z. Deng, and H. Guo, "Real-time road traffic state prediction based on kernel-knn," *Transportmetrica A: Transport Science*, vol. 16, no. 1, pp. 104–118, 2020.

[33] C. Ding, J. Duan, Y. Zhang, X. Wu, and G. Yu, "Using an arima-garch modeling approach to improve subway short-term ridership forecasting accounting for dynamic volatility," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1054–1064, 2018.

[34] W. Chen, J. An, R. Li, L. Fu, G. Xie, M. Z. A. Bhuiyan, and K. Li, "A novel fuzzy deep-learning approach to traffic flow prediction with uncertain spatial-temporal data features," *Future Generation Computer Systems*, vol. 89, pp. 78 – 88, 2018.

[35] S. Rahimipour, R. Moeinfar, and S. M. Hashemi, "Traffic prediction using a self-adjusted evolutionary neural network," *Journal of Modern Transportation*, pp. 1–11, 2018.

[36] A. Ermagun and D. Levinson, "Spatiotemporal traffic forecasting: review and proposed directions," *Transport Reviews*, vol. 38, no. 6, pp. 786–814, 2018.

[37] M. Shahgholian and D. Gharavian, "Advanced Traffic Management Systems: An Overview and A Development Strategy," *Computing Research Repository - CoRR*, 2018.

[38] T. M. Mitchell, *Machine Learning*. USA: McGraw-Hill, Inc., 1997.

[39] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708 – 713, 2015.

[40] T. Vyas, P. Prajapati, and S. Gadhwal, "A survey and evaluation of supervised machine learning techniques for spam e-mail filtering," in *IEEE International Conference on Electrical, Computer and Communication Technologies*, 2015, pp. 1–7.

[41] M.-A. Zöller and M. F. Huber, "Survey on Automated Machine Learning," *Computing Research Repository - CoRR*, 2019.

[42] S. Garcia, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Springer Publishing Company, Incorporated, 2014.

[43] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, "Data cleaning: Overview and emerging challenges," in *Proceedings of the International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2016, p. 2201–2206.

[44] W. Y. Kim, B. Choi, E. K. Hong, S.-K. Kim, and D. Lee, "A taxonomy of dirty data," *Data Mining and Knowledge Discovery*, vol. 7, pp. 81–99, 2003.

[45] D. Pyle, *Data Preparation for Data Mining*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[46] T. Y. T. Lin, "Attribute transformations for data mining i: Theoretical explorations," *International Journal of Intelligent Systems*, vol. 17, no. 2, pp. 213–222, 2002.

[47] X. L. Dong and T. Rekatsinas, "Data integration and machine learning: A natural synergy," in *Proceedings of the 2018 International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2018, p. 1645–1650.

[48] J. Han, M. Kamber, and J. Pei. (2012) Data mining concepts and techniques. Waltham, Mass.

[49] H. Wang and S. Wang, "Mining incomplete survey data through classification," *Knowledge and Information Systems*, vol. 24, pp. 221–233, 2009.

[50] R. J. Mislevy, *Journal of Educational Statistics*, vol. 16, no. 2, pp. 150–155, 1991.

[51] J. Derrac, I. Triguero, S. Garcia, and F. Herrera, "Integrating instance selection, instance weighting, and feature weighting for nearest neighbor classifiers by coevolutionary algorithms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 5, pp. 1383–1397, 2012.

[52] S. García, J. Derrac, I. Triguero, C. J. Carmona, and F. Herrera, "Evolutionary-based selection of generalized instances for imbalanced classification," *Knowledge-Based Systems*, vol. 25, no. 1, pp. 3 – 12, 2012.

[53] Huan Liu and R. Setiono, "Feature selection via discretization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 4, pp. 642–645, 1997.

[54] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *International Conference on Discovery Science*, 1993.

[55] D. Charte, F. Charte, S. García, and F. Herrera, "A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations," *Progress in Artificial Intelligence*, pp. 1–14, 2018.

[56] T. J. S. Geoffrey Hinton, "Unsupervised Learning," in *Unsupervised Learning: Foundations of Neural Computation*. The MIT Press, 1999.

[57] D. Dhall, R. Kaur, and M. Juneja, "Machine learning: A review of the algorithms and its applications," in *Proceedings of International Conference on Recent Innovations in Computing*, P. K. Singh, A. K. Kar, Y. Singh, M. H. Kolekar, and S. Tanwar, Eds. Cham: Springer International Publishing, 2020, pp. 47–63.

[58] S. Marsland, *Machine Learning: An Algorithmic Perspective, Second Edition*. Chapman Hall/CRC, 2014.

[59] K. Fukunaga, *Introduction to Statistical Pattern Recognition (2nd Ed.)*. USA: Academic Press Professional, Inc., 1990.

[60] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transaction Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, p. 4–37, 2000.

[61] D. M. J. Tax and R. P. W. Duin, "Using two-class classifiers for multiclass classification," in *Object recognition supported by user interaction for service robots*, vol. 2, 2002, pp. 124–127 vol.2.

[62] P. H. R.O. Duda and D. Stork, *Pattern Classification*. Berlin, Heidelberg: Springer-Verlag, 2007.

[63] A. J. Smola and B. Schölkopf, "On a kernel-based method for pattern recognition, regression, approximation, and operator inversion," *Algorithmica*, vol. 22, pp. 211–231, 1998.

[64] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[65] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[66] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241 – 258, 2019.

[67] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[68] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761 – 1776, 2011.

[69] R. E. Schapire, "The strength of weak learnability," *Journal of Machine Learning*, vol. 5, no. 2, p. 197–227, 1990.

[70] L. Breiman, "Bagging predictors," *Journal of Machine Learning*, vol. 24, p. 123–140, 1996.

[71] R. Ranawana and V. Palade, "Multi-classifier systems: Review and a roadmap for developers," *International Journal of Hybrid Intelligent Systems*, vol. 3, no. 1, p. 35–61, 2006.

[72] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[73] S. Day, *Principles of statistical inference*. Cambridge University Press, 2007.

[74] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*, ser. Springer series in statistics. Springer, 2009.

[75] D. Allen, "The relationship between variable selection and data augmentation and a method for prediction," *Technometrics*, vol. 16, pp. 125–127, 1974.

[76] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 36, no. 2, pp. 111–147, 1974.

[77] ——, "An asymptotic equivalence of choice of model by cross-validation and akaike's criterion," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 44–47, 1977.

[78] J. Brownlee, *Introduction to Time Series Forecasting With Python*, 1st ed. Copyright 2018 Jason Brownlee, 2018, vol. 1.

[79] R. Kohavi and D. Wolpert, "Bias plus variance decomposition for zero-one loss functions," in *Proceedings of the International Conference on International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996, p. 275–283.

[80] Q. Yao, M. Wang, Y. Chen, W. Dai, H. Yi-Qi, L. Yu-Feng, T. Wei-Wei, Y. Qiang, and Y. Yang, "Taking Human out of Learning Applications: A Survey on Automated Machine Learning," *Computing Research Repository - CoRR*, 2018.

[81] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn," in *Proceedings of SciPy*, 2014, p. 33–39.

[82] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *2015 IEEE International Conference on Data Science and Advanced Analytics*, 2015, pp. 1–10.

[83] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. Turaga, "Learning feature engineering for classification," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 2529–2535.

[84] G. Katz, E. C. R. Shin, and D. Song, "Explorekit: Automatic feature generation and selection," in *2016 IEEE 16th International Conference on Data Mining*, 2016, pp. 979–984.

[85] L. Friedman and S. Markovitch, "Recursive feature generation for knowledge-based learning," *Computing Research Repository - CoRR*, 2018.

[86] M. J. Smith, R. Wedge, and K. Veeramachaneni, "Featurehub: Towards collaborative data science," in *2017 IEEE International Conference on Data Science and Advanced Analytics*, 2017, pp. 590–600.

[87] M. Martin Salvador, M. Budka, and B. Gabrys, "Towards automatic composition of multicomponent predictive systems," in *Hybrid Artificial Intelligent Systems*, F. Martínez-Álvarez, A. Troncoso, H. Quintián, and E. Corchado, Eds. Cham: Springer International Publishing, 2016, pp. 27–39.

[88] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, "Data cleaning: Overview and emerging challenges," in *Proceedings of the International Conference on Management of Data*. New York, NY, USA: ACM, 2016, pp. 2201–2206.

[89] X. Chu, J. Morcos, I. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye, "Katara: A data cleaning system powered by knowledge bases and crowdsourcing," in *Proceedings of the International Conference on Management of Data*, vol. 2015-May. Association for Computing Machinery, 2015, pp. 1247–1261.

[90] I. B. Messaoud, H. El Abed, V. Märgner, and H. Amiri, "A design of a preprocessing framework for large database of historical documents," in *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*. New York, NY, USA: ACM, 2011, pp. 177–183.

[91] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," in *Learning and Intelligent Optimization*, C. A. C. Coello, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523.

[92] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*. USA: Curran Associates Inc., 2011, pp. 2546–2554.

[93] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*. USA: Curran Associates Inc., 2012, pp. 2951–2959.

[94] M. Claesen, J. Simm, D. Popovic, Y. Moreau, and B. D. Moor, "Easy hyper-parameter search using optunity," *Computing Research Repository - CoRR*, 2014.

[95] T. Swearingen, W. Drevo, B. Cyphers, A. Cuesta-Infante, A. Ross, and K. Veeramachaneni, "Atm: A distributed, collaborative, scalable system for automated machine learning," in *2017 IEEE International Conference on Big Data*, 2017, pp. 151–162.

[96] F. Mohr, M. Wever, and E. Hüllermeier, "ML-Plan: Automated machine learning via hierarchical planning," *Machine Learning*, vol. 107, no. 8, pp. 1495–1515, 2018.

[97] P. Nguyen, M. Hilario, and A. Kalousis, "Using meta-mining to support data mining workflow planning and optimization," *Journal of Artificial Intelligence Research*, vol. 51, no. 1, p. 605–644, 2014.

[98] J. Vanschoren, "Meta-learning," F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Springer, 2018, pp. 39–68.

[99] I. B. M. Skycomp, *Major High- way Performance Ratings and Bottleneck Inventory*. State of Maryland: Maryland State Highway Administration, the Baltimore Metropolitan Council and Maryland Transportation Authority, 2009.

[100] Kerschke, Pascal and Hoos, Holger and Neumann, Frank and Trautmann, Heike, "Automated Algorithm Selection: Survey and Perspectives," *Computing Research Repository - CoRR*, 2018.

[101] J. S. Angarita-Zapata, I. Triguero, and A. D. Masegosa, "A Preliminary Study on Automatic Algorithm Selection for Short-Term Traffic Forecasting," in *Intelligent Distributed Computing XII*, J. Del Ser, E. Osaba, M. N. Bilbao, J. J. Sanchez-Medina, M. Vecchio, and X.-S. Yang, Eds. Springer International Publishing, 2018, pp. 204–214.

[102] J. Xia, W. Huang, and J. Guo, "A clustering approach to online freeway traffic state identification using ITS data," *Journal of Civil Engineering*, vol. 16, no. 3, pp. 426–432, 2012.

[103] Y. Gao, S. Sun, and D. Shi, "Network-scale traffic modeling and forecasting with graphical lasso," in *Advances in Neural Networks*, D. Liu, H. Zhang, M. Polycarpou, C. Alippi, and H. He, Eds. Springer Berlin Heidelberg, 2011, pp. 151–158.

[104] E. I. Vlahogianni, "Enhancing Predictions in Signalized Arterials with Information on Short-Term Traffic Flow Dynamics," *Journal of Intelligent Transportation Systems*, vol. 13, no. 2, pp. 73–84, 2009.

[105] J. Lopes, J. Bento, E. Huang, C. Antoniou, and M. Ben-Akiva, "Traffic and mobility data collection for real-time applications," in *IEEE Conference on Intelligent Transportation Systems*, 2010, pp. 216–223.

[106] W. Jin, Y. Lin, Z. Wu, and H. Wan, "Spatio-temporal recurrent convolutional networks for citywide short-term crowd flows prediction," in *International Conference on Compute and Data Analysis*. ACM, 2018, pp. 28–35.

[107] J. Longfoot, "An automatic network travel time system," *Vehicle Navigation and Information Systems Conference, 1991*, vol. 2, pp. 1053–1061, 1991.

[108] P. S. Castro, D. Zhang, and S. Li, "Urban traffic modelling and prediction using large scale taxi GPS traces," in *Pervasive Computing*, J. Kay, P. Lukowicz, H. Tokuda, P. Olivier, and A. Krüger, Eds. Springer Berlin Heidelberg, 2012, pp. 57–72.

[109] J. C. Herrera, D. B. Work, R. Herring, X. J. Ban, Q. Jacobson, and A. M. Bayen, "Evaluation of traffic data obtained via GPS-enabled mobile phones: The mobile century field experiment," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 4, pp. 568 – 583, 2010.

[110] R. Herring, A. Hofleitner, P. Abbeel, and A. Bayen, "Estimating arterial traffic conditions using sparse probe data," in *International IEEE Conference on Intelligent Transportation Systems*, 2010, pp. 929–936.

[111] Q. Ye, W. Y. Szeto, and S. C. Wong, "Short-term traffic speed forecasting based on data recorded at irregular intervals," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1727–1737, 2012.

[112] S. Taguchi, S. Koide, and T. Yoshimura, "Online map matching with route prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 338–347, 2019.

[113] J. Huang, C. Liu, and J. Qie, "Developing map matching algorithm for transportation data center," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2014, pp. 167–170.

[114] Y. Hu, M. Li, H. Liu, X. Guo, X. Wang, and T. Li, "City traffic forecasting using taxi GPS data: A coarse-grained cellular automata model," *Computing Research Repository*, pp. 1–30, 2016.

[115] H. Yang, T. S. Dillon, E. Chang, and Y. P. Chen, "Optimized configuration of exponential smoothing and extreme learning machine for traffic flow forecasting," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 23–34, 2019.

[116] C. Song, H. Lee, C. Kang, W. Lee, Y. B. Kim, and S. W. Cha, "Traffic speed prediction under weekday using convolutional neural networks concepts," in *IEEE Intelligent Vehicles Symposium*, 2017, pp. 1293–1298.

[117] X. Chen, Z. Wei, X. Liu, Y. Cai, Z. Li, and F. Zhao, "Spatiotemporal variable and parameter selection using sparse hybrid genetic algorithm for traffic flow forecasting," *International Journal of Distributed Sensor Networks*, vol. 13, no. 6, pp. 1–14, 2017.

[118] B. Sun, W. Cheng, P. Goswami, and G. Bai, "Flow-aware WPT k-nearest neighbours regression for short-term traffic prediction," in *IEEE Symposium on Computers and Communications*, 2017, pp. 48–53.

[119] ——, "Short-term traffic forecasting using self-adjusting k-nearest neighbours," *IET Intelligent Transport Systems*, vol. 12, no. 1, pp. 41–48, 2018.

[120] A. Attanasi, L. Meschini, M. Pezzulla, G. Fusco, G. Gentile, and N. Isaenko, "A hybrid method for real-time short-term predictions of traffic flows in urban areas," in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems*, 2017, pp. 878–883.

[121] H. Yang, Y. Zou, Z. Wang, and B. Wu, "A hybrid method for short-term freeway travel time prediction based on wavelet neural network and markov chain," *Canadian Journal of Civil Engineering*, vol. 45, no. 2, pp. 77–86, 2018.

[122] B. Sharma, S. Kumar, P. Tiwari, P. Yadav, and M. I. Nezhurina, "ANN based short-term traffic flow forecasting in undivided two lane highway," *Journal of Big Data*, vol. 5, no. 1, pp. 1–16, 2018.

[123] A. Raza and M. Zhong, "Lane-based short-term urban traffic forecasting with GA designed ANN and LWR models," *Transportation Research Procedia*, vol. 25, pp. 1430 – 1443, 2017.

[124] B. Yao, C. Chen, Q. Cao, L. Jin, M. Zhang, H. Zhu, and B. Yu, "Short-term traffic speed prediction for an urban corridor," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 2, pp. 154–169, 2017.

[125] A. Raza and M. Zhong, "Hybrid lane-based short-term urban traffic speed forecasting: A genetic approach," in *International Conference on Transportation Information and Safety*, 2017, pp. 271–279.

[126] K. Liu, S. Gao, P. Qiu, X. Liu, B. Yan, and F. Lu, "Road2vec: Measuring traffic interactions in urban road system from massive travel routes," *International Journal of Geo-Information*, vol. 6, no. 11, pp. 1–14, 2017.

[127] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, pp. 216–233, 2015.

[128] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting," *Computing Research Repository - CoRR*, vol. abs/1709.04875, 2017.

[129] Q. Liu, B. Wang, and Y. Zhu, "Short-term traffic speed forecasting based on attention convolutional neural network for arterials," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 11, pp. 999–1016, 2018.

[130] E. Bolshinsky and R. Freidman, "Traffic Flow Forecast Survey," Technion - Israel Institute of Technology, Computer Science Department, Tech. Rep., 2012.

[131] F. Su, H. Dong, L. Jia, Y. Qin, and Z. Tian, "Long-term forecasting oriented to urban expressway traffic situation," *Advances in Mechanical Engineering*, vol. 8, no. 1, pp. 1–16, 2016.

[132] M. Meng, C.-f. Shao, Y.-d. Wong, B.-b. Wang, and H.-x. Li, "A two-stage short-term traffic flow prediction method based on avl and aknn techniques," *Journal of Central South University*, vol. 22, no. 2, pp. 779–786, 2015.

[133] M.-L. Huang, "Intersection traffic flow forecasting based on v-GSVR with a new hybrid evolutionary algorithm," *Neurocomputing*, vol. 147, pp. 343 – 349, 2015.

[134] J.-t. Zhong and S. Ling, "Key factors of k-nearest neighbours nonparametric regression in short-time traffic flow forecasting," in *International Conference on Industrial Engineering and Engineering Management 2014*, E. Qi, J. Shen, and R. Dou, Eds. Atlantis Press, 2015, pp. 9–12.

[135] S.-y. Liu, D.-w. Li, Y.-g. Xi, and Q.-f. Tang, "A short-term traffic flow forecasting method and its applications," *Journal of Shanghai Jiaotong University*, vol. 20, no. 2, pp. 156–163, 2015.

[136] H. Pan, J. Liu, S. Zhou, and Z. Niu, "A block regression model for short-term mobile traffic forecasting," in *International Conference on Communications in China*, 2015, pp. 1–5.

[137] L. Li, X. Su, Y. Wang, Y. Lin, Z. Li, and Y. Li, "Robust causal dependence mining in big data network and its application to traffic flow predictions," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 292 – 307, 2015.

[138] F. Moretti, S. Pizzuti, S. Panzieri, and M. Annunziato, "Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling," *Neurocomputing*, vol. 167, pp. 3 – 7, 2015.

[139] J. Abdi and B. Moshiri, "Application of temporal difference learning rules in short-term traffic flow prediction," *Expert Systems*, vol. 32, no. 1, pp. 49–64, 2015.

[140] J. Zhao and S. Sun, "High-order gaussian process dynamical models for traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 2014–2019, 2016.

[141] W. Yuankai, H. Tan, J. Peter, B. Shen, and B. Ran, "Short-term traffic flow prediction based on multilinear analysis and k-nearest neighbor regression," in *CICTP 2015*, 2015, pp. 556–569.

[142] P. Cai, Y. Wang, G. Lu, P. Chen, C. Ding, and J. Sun, "A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 62, pp. 21 – 34, 2016.

[143] A. Wibisono, W. Jatmiko, H. A. Wisesa, B. Hardjono, and P. Mursanto, "Traffic big data prediction and visualization using fast incremental model trees-drift detection," *Knowledge-Based Systems*, vol. 93, pp. 33 – 46, 2016.

[144] Y. Hou, P. Edara, and C. Sun, "Traffic Flow Forecasting for Urban Work Zones," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1761–1770, aug 2015.

[145] C. xia Yang, R. Fu, and Y. qin Fu, "Prediction of short-term traffic flow based on similarity," *Journal of Highway and Transportation Research and Development*, vol. 10, no. 1, pp. 92–97, 2016.

[146] J. Xu, D. Deng, U. Demiryurek, C. Shahabi, and M. v. d. Schaar, "Mining the situation: Spatiotemporal traffic prediction with big data," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 702–715, 2015.

[147] X. Niu, Y. Zhu, Q. Cao, X. Zhang, W. Xie, and K. Zheng, "An online-traffic-prediction based route finding mechanism for smart city," *International Journal of Distributed Sensor Networks*, vol. 11, no. 8, pp. 1–16, 2015.

[148] N. E. Faouzi, H. Leung, and A. Kurian, "Data fusion in intelligent transportation systems: Progress and challenges - a survey," *Information Fusion*, vol. 12, no. 1, pp. 4 – 10, 2011.

[149] T.-I. Theodorou, A. Salamanis, D. D. Kehagias, D. Tzovaras, and C. Tjortjis, "Short-term traffic prediction under both typical and atypical traffic conditions using a pattern transition model," in *International Conference on Vehicle Technology and Intelligent Transport Systems*, 2017.

[150] H. jun Yang and X. Hu, "Wavelet neural network with improved genetic algorithm for traffic flow time series prediction," *Optik*, vol. 127, no. 19, pp. 8103 – 8110, 2016.

[151] L. Do, N. Taherifar, and H. L. Vu, "Survey of neural network-based models for short-term traffic state prediction," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, p. e1285, 09 2018.

[152] C. Dong, Z. Xiong, C. Shao, and H. Zhang, "A spatial-temporal-based state space approach for freeway network traffic flow modelling and prediction," *Transportmetrica A: Transport Science*, vol. 11, no. 7, pp. 547–560, 2015.

[153] D. Boto-Giralda, F. J. Díaz-Pernas, D. González-Ortega, J. F. Díaz-Higuera, M. Antón-Rodríguez, M. Martínez-Zarzuela, and I. Torre-Díez, "Wavelet-based denoising for traffic volume time series forecasting with self-organizing neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 25, no. 7, pp. 530–545, 2010.

[154] X. Jiang and H. Adeli, "Wavelet packet-autocorrelation function method for traffic flow pattern analysis," *Computer-Aided Civil and Infrastructure Engineering*, vol. 19, no. 5, pp. 324–337, 2004.

[155] E. Vlahogianni and M. Karlaftis, "Temporal aggregation in traffic data: implications for statistical characteristics and model choice," *Transportation Letters*, vol. 3, no. 1, pp. 37–49, 2011.

[156] S. Turner, W. Eisele, R. J Benz, and D. J Holdener, *Travel Time Data Collection Handbook*. Federal Highway Administration - USA, 1998.

[157] G. Leduc, "Road traffic data: Collection methods and applications," *Energy, Transport and Climate Change*, pp. 1–51, 2008.

[158] F. Soriguera and F. Robusta, "Estimation of traffic stream space mean speed from time aggregations of double loop detector data," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 1, pp. 115 – 129, 2011.

[159] D. Satrinia and G. A. P. Saptawati, "Traffic speed prediction from GPS data of taxi trip using support vector regression," in *International Conference on Data and Software Engineering*, 2017, pp. 1–6.

[160] J. Grengs, X. Wang, and L. Kostyniuk, "Using GPS data to understand driving behavior," *Journal of Urban Technology*, vol. 15, no. 2, pp. 33–53, 2008.

[161] A. Bezuglov and G. Comert, "Short-term freeway traffic parameter prediction: Application of grey system theory models," *Expert Systems with Applications*, vol. 62, pp. 284 – 292, 2016.

[162] J. Rupnik, J. Davies, B. Fortuna, A. Duke, and S. S. Clarke, "Travel time prediction on highways," in *International Conference on Computer and Information Technology*, 2015, pp. 1435–1442.

[163] D. Wang, Q. Zhang, S. Wu, X. Li, and R. Wang, "Traffic flow forecast with urban transport network," in *2016 IEEE International Conference on Intelligent Transportation Engineering*. IEEE, 2016, pp. 139–143.

[164] H. Botma and P. Bovy, *The free speed functions of drivers*. Delft University Press, 2000, pp. 53–74.

[165] J. W. C. van Lint and N. J. van der Zijpp, "Improving a travel-time estimation algorithm by using dual loop detectors," *Transportation Research Record*, vol. 1855, no. 1, pp. 41–48, 2003.

[166] F. Soriguera and F. Robuste, "Requiem for freeway travel time estimation methods based on blind speed interpolations between point measurements," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 291–297, 2011.

[167] Y. Zhang and A. Haghani, "A gradient boosting method to improve travel time prediction," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 308 – 324, 2015.

[168] I. Laña, J. Del Ser, M. Vélez, and I. Oregi, "Joint feature selection and parameter tuning for short-term traffic flow forecasting based on heuristically optimized multi-layer neural networks," in *Harmony Search Algorithm*, J. Del Ser, Ed. Springer Singapore, 2017, pp. 91–100.

[169] H. Yang, T. S. Dillon, and Y.-P. P. Chen, "Evaluation of recent computational approaches in short-term traffic forecasting," in *Artificial Intelligence in Theory and Practice IV*, T. Dillon, Ed. Springer International Publishing, 2015, pp. 108–116.

[170] I. Laña, I. Olabarrieta, M. Velez, and J. D. Ser, "On the imputation of missing data for road traffic forecasting: New insights and novel techniques," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 18 – 33, 2018.

[171] J. Han, M. Kamber, and J. Pei, *Data mining concepts and techniques, third edition*. Waltham, Mass.: Morgan Kaufmann Publishers, 2012.

[172] D. Pavlyuk, "Feature selection and extraction in spatiotemporal traffic forecasting: a systematic literature review," *European Transport Research Review*, vol. 11, no. 1, p. 6, 2019.

[173] S. Cheng, F. Lu, P. Peng, and S. Wu, "A spatiotemporal multi-view-based learning method for short-term traffic forecasting," *International Society for Photogrammetry and Remote Sensing - Journal of Geo-Information*, vol. 7, pp. 1–21, 2018.

[174] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory." *PloS one*, vol. 10, no. 3, pp. 1–17, 2015.

[175] Y. Zhang and Y. Zhang, "A comparative study of three multivariate short-term freeway traffic flow forecasting methods with missing data," *Journal of Intelligent Transportation Systems*, vol. 20, no. 3, pp. 205–218, 2016.

[176] X. Li and W. Gao, "Prediction of traffic flow combination model based on data mining," *International Journal of Database Theory and Application*, vol. 8, pp. 303–312, 12 2015.

[177] H.-p. Lu, Z.-y. Sun, W.-c. Qu, and L. Wang, "Real-Time Corrected Traffic Correlation Model for Traffic Flow Forecasting," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–7, 2015.

[178] P. Dell'Acqua, F. Bellotti, R. Berta, and A. D. Gloria, "Time-aware multivariate nearest neighbor regression methods for traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3393–3402, 2015.

[179] W. Ma and R. Wang, "Traffic flow forecasting research based on bayesian normalized elman neural network," in *IEEE Signal Processing and Signal Processing Education Workshop*, 2015, pp. 426–430.

[180] R. T. Das, K. K. Ang, and C. Quek, "ierspop: A novel incremental rough set-based pseudo outer-product with ensemble learning," *Applied Soft Computing*, vol. 46, pp. 170 – 186, 2016.

[181] G. Fusco, C. Colombaroni, L. Comelli, and N. Isaenko, "Short-term traffic predictions on large urban traffic networks: Applications of network-based machine learning models and dynamic traffic assignment models," 2015, pp. 93–101.

[182] D. Xia, B. Wang, H. Li, Y. Li, and Z. Zhang, "A distributed spatial–temporal weighted model on MapReduce for short-term traffic flow forecasting," *Neurocomputing*, vol. 179, pp. 246–263, feb 2016.

[183] Y. Cong, J. Wang, and X. Li, "Traffic flow forecasting by a least squares support vector machine with a fruit fly optimization algorithm," *Green Intelligent Transportation System and Safety - Procedia Engineering*, vol. 137, pp. 59 – 68, 2016.

[184] Z. Ma, G. Luo, and D. Huang, "Short term traffic flow prediction based on on-line sequential extreme learning machine," in *International Conference on Advanced Computational Intelligence*, 2016, pp. 143–149.

[185] Y. Xu, H. Chen, Q.-J. Kong, X. Zhai, and Y. Liu, "Urban traffic flow prediction: a spatio-temporal variable selection-based approach," *Journal of Advanced Transportation*, vol. 50, no. 4, pp. 489–506, 2016.

[186] J. van Lint, S. Hoogendoorn, and H. van Zuylen, "Accurate freeway travel time prediction with state-space neural networks under missing data," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 5, pp. 347 – 369, 2005.

[187] H. Dia, "An object-oriented neural network approach to short-term traffic forecasting," *European Journal of Operational Research*, vol. 131, no. 2, pp. 253 – 261, 2001.

[188] M. Lippi, M. Bertini, and P. Frasconi, "Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 871–882, 2013.

[189] H. Chen, S. Grant-Muller, L. Mussone, and F. Montgomery, "A study of hybrid neural network approaches and the effects of missing data on traffic forecasting," *Neural Computing & Applications*, vol. 10, no. 3, pp. 277–286, 2001.

[190] S. Ishak and C. Alecsandru, "Optimizing traffic prediction performance of neural networks under various topological, input, and traffic condition settings," *Journal of Transportation Engineering*, vol. 130, no. 4, pp. 452–465, 2004.

[191] S. Huang and A. W. Sadek, "A novel forecasting approach inspired by human memory: The example of short-term traffic volume forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 5, pp. 510 – 525, 2009.

[192] Y. Zhang and Y. Liu, "Traffic forecasting using least squares support vector machines," *Transportmetrica*, vol. 5, no. 3, pp. 193–213, 2009.

[193] B. Park, "Hybrid neuro-fuzzy application in short-term freeway traffic volume forecasting," *Transportation Research Record*, vol. 1802, no. 1, pp. 190–196, 2002.

[194] S. Dunne and B. Ghosh, "Regime-based short-term multivariate traffic condition forecasting algorithm," *Journal of Transportation Engineering*, vol. 138, no. 4, pp. 455–466, 2012.

[195] M. Zhong, S. Sharma, and P. Lingras, "Short-term traffic prediction on different types of roads with genetically designed regression and time delay neural network models," *Journal of Computing in Civil Engineering*, vol. 19, no. 1, pp. 94–103, 2005.

[196] D. Srinivasan, C. W. Chan, and P. Balaji, "Computational intelligence-based congestion prediction for a dynamic urban street network," *Neurocomputing*, vol. 72, no. 10, pp. 2710 – 2716, 2009.

[197] N. Zhang, Y. Zhang, and H. Lu, "Seasonal autoregressive integrated moving average and support vector machine models: Prediction of short-term traffic flow on freeways," *Transportation Research Record*, vol. 2215, no. 1, pp. 85–92, 2011.

[198] B. Gültekin, M. Sari, and O. Borat, "A neural network based traffic-flow prediction model," *Mathematical and Computational Applications*, vol. 15, no. 2, pp. 269–278, 2010.

[199] S. He, C. Hu, G.-j. Song, K.-q. Xie, and Y.-z. Sun, "Real-time short-term traffic flow forecasting based on process neural network," in *Advances in Neural Networks*, F. Sun, J. Zhang, Y. Tan, J. Cao, and W. Yu, Eds. Springer Berlin Heidelberg, 2008, pp. 560–569.

[200] J. Wang, P. Shang, and X. Zhao, "A new traffic speed forecasting method based on bi-pattern recognition," *Fluctuation and Noise Letters*, vol. 10, no. 01, pp. 59–75, 2011.

[201] H. Chen, M. S. Dougherty, and H. R. Kirby, "The effects of detector spacing on traffic forecasting performance using neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 16, no. 6, pp. 422–430, 2001.

[202] Y. Zhang and M.-l. Wang, "Peak traffic forecasting using nonparametric approaches," *Journal of Shanghai Jiaotong University*, vol. 17, no. 1, pp. 76–81, 2012.

[203] A. Dharia and H. Adeli, "Neural network model for rapid forecasting of freeway link travel time," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 7, pp. 607 – 613, 2003.

[204] S. Sun, C. Zhang, G. Yu, N. Lu, and F. Xiao, "Bayesian network methods for traffic flow forecasting with incomplete data," in *European Conference on Machine Learning*, J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, Eds. Springer Berlin Heidelberg, 2004, pp. 419–428.

[205] G. Luo, "A review of automatic selection methods for machine learning algorithms and hyper-parameter values," *Network Modeling Analysis in Health Informatics and Bioinformatics*, vol. 5, no. 1, p. 18, 2016.

[206] J. R. Rice, "The algorithm selection problem," ser. Advances in Computers, M. Rubinoff and M. C. Yovits, Eds. Elsevier, 1976, vol. 15, pp. 65 – 118.

[207] J. Park, Y. L. Murphey, R. McGee, J. G. Kristinsson, M. L. Kuang, and A. M. Phillips, "Intelligent Trip Modeling for the Prediction of an Origin–Destination Traveling Speed Profile," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 1039–1053, 2014.

[208] S. Garcia, A. Fernandez, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044 – 2064, 2010.

[209] A. Tharwat, "Classification assessment methods," *Applied Computing and Informatics*, 2018.

[210] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679 – 688, 2006.

[211] B. Krawczyk, B. T. McInnes, and A. Cano, "Sentiment classification from multi-class imbalanced twitter data using binarization," in *Hybrid Artificial Intelligent Systems*, 2017, pp. 26–37.

[212] V. López, I. Triguero, C. J. Carmona, S. García, and F. Herrera, "Addressing imbalanced classification with instance generation techniques: Ipade-id," *Neurocomputing*, vol. 126, pp. 15 – 28, 2014.

[213] J. S. Angarita-Zapata, A. D. Masegosa, and I. Triguero, "General-purpose automated machine learning for transportation: A case study of auto-sklearn for traffic forecasting," in *Proceeding of the 18th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Cham: Springer International Publishing, 2020.

[214] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proceedings of the Twenty-First International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2004, p. 18.

[215] P. Gijsbers, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren, "An open source automl benchmark," *Computing Research Repository - CoRR*, 2019, work presented at AutoML Workshop at International Conference on Machine Learning 2019.

[216] I. Guyon, I. Chaabane, H. J. Escalante, S. Escalera, D. Jajetic, J. R. Lloyd, N. Macià, B. Ray, L. Romaszko, M. Sebag, A. R. Statnikov, S. Treguer, and E. Viegas, "A brief review of the chalearn automl challenge: Any-time any-dataset learning without human intervention," in *Proceedings of the Workshop on Automatic Machine Learning*. New York, USA: PMLR, 2016, pp. 21–30.

[217] B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, and J. Vanschoren, "Openml benchmarking suites," 2017.

[218] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," *Computing Research Repository - CoRR*, 2018.

[219] Z. Yang and L. S. Pun-Cheng, "Vehicle detection in intelligent transportation systems and its applications under varying environments: A review," *Image and Vision Computing*, vol. 69, pp. 143 – 154, 2018.

[220] W. Jiang and L. Zhang, "Geospatial data to images: A deep-learning framework for traffic forecasting," *Tsinghua Science and Technology*, vol. 24, no. 1, pp. 52–64, 2019.

[221] D. Pavlyuk, "Spatiotemporal traffic forecasting as a video prediction problem," in *6th International Conference on Models and Technologies for Intelligent Transportation Systems*, 2019, pp. 1–7.

[222] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, p. 237–285, 1996.

[223] T. H. H. Aldhyani and M. R. Joshi, "Clustering to enhance network traffic forecasting," in *Information and Communication Technology for Sustainable Development*, D. K. Mishra, M. K. Nayak, and A. Joshi, Eds. Springer Singapore, 2018, pp. 357–364.

[224] D. Nallaperuma, R. Nawaratne, T. Bandaragoda, A. Adikari, S. Nguyen, T. Kempitiya, D. De Silva, D. Alahakoon, and D. Pothuhera, "Online incremental machine learning platform for big data-driven smart traffic management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4679–4690, 2019.

[225] L. Zimmer, M. Lindauer, and F. Hutter, "Auto-pytorch tabular: Multi-fidelity metalearning for efficient and robust autodl," 2020.

[226] M. Jiménez, I. Triguero, and R. John, "Handling uncertainty in citizen science data: Towards an improved amateur-based large-scale classification," *Information Sciences*, vol. 479, pp. 301 – 320, 2019.

[227] Z.-Q. Zhao, S.-T. Xu, D. Liu, W.-D. Tian, and Z.-D. Jiang, "A review of image set classification," *Neurocomputing*, vol. 335, pp. 251 – 260, 2019.