General-Purpose Automated Machine Learning for Transportation: A Case Study of Auto-sklearn for Traffic Forecasting

Juan S. Angarita-Zapata[1](✉), Antonio D. Masegosa[1,2], and Isaac Triguero[3]

[1] DeustoTech, Faculty of Engineering, University of Deusto, Av. Universidades, 24, 48007 Bilbao, Spain
{js.angarita,ad.masegosa}@deusto.es

[2] IKERBASQUE, Basque Foundation for Science, 48011 Bilbao, Spain

[3] Computational Optimisation and Learning (COL) Lab, School of Computer Science, University of Nottingham, Nottingham, UK
Isaac.Triguero@nottingham.ac.uk

Abstract. Currently, there are no guidelines to determine what are the most suitable machine learning pipelines (i.e. the workflow from data preprocessing to model selection and validation) to approach Traffic Forecasting (TF) problems. Although automated machine learning (AutoML) has proved to be successful dealing with the model selection problem in other applications areas, only a few papers have explored the performance of general-purpose AutoML methods, purely based on optimisation, when tackling TF. In this paper, we provide a thorough exploration of the benefits of Auto-sklearn for TF, as a general-purpose AutoML method that follows a hybrid search strategy combining optimisation with meta-learning and ensemble learning. Particularly, we focus on how well Auto-sklearn is able to recommend competitive machine learning pipelines to forecast traffic, modelled as a TF multi-class imbalanced classification problem, along different time horizons at two spatial scales (point and road segment) and two environments (freeway and urban). Concretely, we test the following scenarios: I) a hybrid search strategy with the three components (optimisation, meta-learning, ensemble learning), II) a strategy based on meta-learning and ensemble learning, and III) a strategy based on the estimation of the best performing pipeline from those suggested by the meta-learning. Experimental results show that the meta-learning component of Auto-sklearn does not work properly on TF problems, and on the other hand, that the optimisation does not contribute too much to the final performance of predictions.

Keywords: Traffic Forecasting · Transportation · Supervised learning · Machine learning · Automated machine learning · Computational intelligence

M.-J. Lesot et al. (Eds.): IPMU 2020, CCIS 1238, pp. 728–744, 2020.
https://doi.org/10.1007/978-3-030-50143-3_57

1 Introduction

A well-established strategy to tackle congestion is the design, development and implementation of TF systems. TF can be defined as the prediction of near-future traffic conditions (e.g. travel time) [16]. The recent emergence of telecommunications technologies integrated into transportation infrastructure generates vast volumes of traffic data. This unprecedented data availability and growing computational capacities have incremented the use of Machine Learning (ML) to address TF. From a ML perspective, TF is focused on building a predictive model using historical data to make predictions of traffic measures based on new and unseen data.

In spite of the aforementioned progress, different ML algorithms and preprocessing approaches may be more appropriate for different kinds of traffic data. Determining the best pipeline (sequence of data preprocessing techniques and a learning algorithm) for making traffic predictions is not a trivial task. In the ML area, this challenge is known as the Model Selection Problem (MSP) and Automated Machine Learning (AutoML) has been one of the most successful approaches addressing it so far. AutoML aims at automatically finding the best combination of preprocessing techniques, ML algorithm and hyperparameters that maximise a performance measure on given data without being specialized in the problem domain where this data comes from. The search strategy to find the mentioned combination can be based either on a "pure" optimisation process that tests different promising combinations from a predefined base of preprocessing and learning algorithms [10]; or it can be based on a hybrid search where the optimisation is complemented with learning strategies such as meta-learning [3]. In the 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 [14]. Roughly speaking, it can be seen as using ML for designing ML algorithms.

AutoML methods have successfully approached the MSP in other areas [8,18], however, it has hardly been explored in TF [1]. In the latter area, the current progress is focused only on AutoML methods designed purely on optimisation approaches; thus, leaving aside the study of AutoML methods that have hybrid search strategies. Having this idea in mind, the contribution of this paper is to study the benefits in terms of performance and computational cost of hybrid AutoML for TF. We use Auto-sklearn [3], a state-of-the-art hybrid AutoML method whose search strategy of pipelines uses bayesian optimisation, meta-learning and ensemble learning. To accomplish this objective, we use as a benchmark a multi-class imbalanced classification problem for different time horizons and for freeway and urban environments. Under these traffic forecasting settings, we explore the performance of the Auto-sklearn's components through three scenarios: I) a hybrid search strategy that uses its three components (optimisation, meta-learning, ensemble learning), II) a meta-learning strategy combined with ensemble learning, and III) a strategy based on the estimation of the best performing pipeline from those suggested by the meta-learning.

The rest of this paper is structured as follows. Sections 2 and 3 present background and related work about AutoML methods in TF. Section 4 exposes the methodology followed in this paper. Then, Sect. 5 analyzes the main results obtained. Finally, conclusions are discussed in Sect. 6.

2 Background

This section reviews literature related to AutoML in the context of TF. We start presenting the foundations of general-purpose AutoML methods and finally, Sect. 2.2 reviews Auto-sklearn, the state-of-the-art hybrid AutoML method used in this research.

2.1 Automated Machine Learning

According to [18], 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

$$A = \{A_{preprocessing} \cup A_{feature} \cup A_{algorithm}\} \tag{1}$$

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 $\lambda^i \in \Lambda$. In order to build a ML pipeline with this structure, human effort and high computational capacities are needed because there is no pipeline that can achieve good performance on every learning problem [6,17]. This usually is done by means of a trial and error approach in an iterative manner, which causes that the success of ML comes at a great price [17].

AutoML is an emerging sub-area in ML that seeks to automatise the ML workflow from data preprocessing to model validation [5]. It allows reducing human bias and improving computational costs by making the construction of ML applications more efficient. The process consists of identifying the most promising combination P_{A^i,λ^i} that satisfies a given performance metric or condition when P_{A^i,λ^i} is trained on training data $D^{(i)}_{train}$ and evaluated on test data $D^{(i)}_{test}$.

Current literature [5,18] reports a variety of general-purpose AutoML methods. According to Chen et al. [17], there are two types of taxonomies that can categorise these methods. First, a "what" taxonomy that determines which stages of a ML pipeline are going to be automated (e.g., data preprocessing and algorithm selection, algorithm selection and hyperparameters, or even the entire pipeline). Within this taxonomy, the most common case is the CASH [13] (Combined Algorithm Selection and Hyperparameter) problem wherein AutoML is focused on finding the best combination of ML algorithm and its hyperparameters setting, leaving the data preprocessing up to the human user. In this paper, we are focused on the automation of fixed-size ML pipelines composed of data preprocessing techniques and a classifier algorithm with their respective hyperparameters configurations.

In contrast, the second taxonomy proposed by [17] classifies how the automation process to find the most promising pipeline is done. On the one hand, some AutoML methods use only an optimisation strategy wherein the ML pipeline is built testing multiple possible combinations from a predefined search space of preprocessing methods, ML algorithms and hyperparameters configurations. From this perspective, the ML pipeline building problem consists of finding a pipeline structure $P_{A,\lambda}$ that minimises a cross-validation loss function

$$P_{A^*,\lambda^*} = \underset{A^{(i)} \in A, \lambda^{(i)} \in \Lambda}{argmin} \frac{1}{K} \sum_{i=1}^{k} \gamma \left(P_{A^{(i)},\lambda^{(i)}}, D_{train}, D_{test} \right) \tag{2}$$

As shown in Eq. 2, this search process can be considered as a black-box optimisation problem that is not easily solvable as the search space can be large and complex. This equation is usually non-smooth and derivative-free, and convergence speed is a critical problem for building ML pipelines. Some methods to solve this equation are grid search, random search, bayesian optimisation and sequential Model-Based Optimisation.

On the other hand, regarding the second taxonomy proposed by [17], there are other AutoML methods that use the aforementioned optimisation in combination with learning strategies to constitute a hybrid search strategy with the purpose of reducing computational costs. In this case, the focus is on applying a ML algorithm at the meta-level to learn meta-knowledge that guides the AutoML process; this approach is known as meta-learning [7,14]. Meta-learning is the data-driven task of systematically observing how different learning algorithms or pipelines perform on different learning tasks and then learning from this experience to warm-start the optimisation process in a new and unknown ML task. This warm-start consists of promising pipelines that are used by the optimisation as starting points to be evaluated in the first place before trying pipelines extracted from a predefined search space.

Meta-learning can extract meta-knowledge using three different strategies [14]. First, there is the "Learning from prior evaluations" strategy wherein a set of known-previous learning tasks $t_j \in T$, a set of configurations $\lambda_i \in \Lambda$ (e.g., hyperparameter settings), and a set of all prior evaluations $P_{i,j}$ coming from applying the configurations Λ over the tasks T, have to be given. Having this knowledge, the objective is to train a meta-learner L able to recommend promising hyperparameters configurations λ_i for an unseen and new task t_{new}. In contrast, the second approach is known as "Learning from task properties". It is based on characterising the known-previous learning tasks $t_j \in T$ using meta-features $m_j \in M$ (e.g, number of instances and features, class imbalance), then extracting the configurations $\lambda_i \in \Lambda$ of the learners associated with these prior tasks, and finally collecting the performance $P_{i,j}$ of the trained models given the meta-features m_j and the configurations λ_i. Having this meta-knowledge, the objective is to train a meta-leaner L that predicts the performance of pipelines or recommend them for an unseen and new task $task_{new}$.

Lastly, the third approach is "Learning from prior models". In this case, the focus is on training a meta-learner given the parameters $m_j \in M$ (e.g. model parameters, features) of prior learnt models, their configurations λ_i, and the performance $P_{i,j}$ of these learners over the previous and known tasks. Then, the objective is to train a meta-learner L that transfers trained models to save computational costs at the moment of approaching a new task t_{new}.

Within this paper, we focus on Auto-sklearn, the AutoML method with a hybrid search strategy that includes optimisation and meta-learning based on the approach "Learning from task properties". This method is presented with more details in the following section.

2.2 Auto-sklearn

Auto-sklearn is an AutoML method that uses meta-learning, bayesian optimisation and ensemble selection to find promising ML pipelines composed of preprocessing methods and ML classifiers. Here we provide a brief description of the method. The interested reader is referred to [3] for further details.

In an off-line phase, for a repository of 121 data-sets, bayesian optimisation is used to determine an optimised ML pipeline with high performance on every data-set. These pipelines are generated from a search space of 15 classifiers, 14 feature preprocessing methods, and 4 data preprocessing methods. Then, for each data-set, a set of 38 meta-features is extracted to characterise every set of data; these meta-features include simple, 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 data-sets, 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 data-set and the optimised pipeline that works well on it.

In the online phase, that is, when a new data-set D_{new} is given, Auto-sklearn computes its meta-features, ranks all the data-sets stored in the meta-knowledge base (stored in the form of meta-features and not the data itself) by their $L1$ distance w.r.t. D_{new}, and selects the stored ML pipelines for the k nearest data-sets (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 data-sets 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 aforementioned search space. 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 hyper-parameter setting, and it is more robust than only using the best pipeline found with the optimisation component.

3 Related Work

Within the most representative AutoML methods are Auto-WEKA [13], Auto-sklearn [3], TPOT [10], ATM [12], and ML-Plan [9]. In the case of the first two methods, they are focused on the construction of fixed ML pipelines in which the pipeline structure is a linear sequence of data preprocessing and algorithm learning. The other methods work by building pipeline structures that can be more complex and diverse. In the case of Auto-WEKA, TPOT, ATM and ML-Plan, they use an optimisation approach to find pipeline structures; meanwhile, Auto-sklearn is the state-of-the-art method to generate ML pipelines using a hybrid search strategy. As a common denominator, all these AutoML methods are agnostic w.r.t. the problem domains in which they have been applied; in this sense, they are general-purpose methods that have shown competitive performance in different applications areas [5].

In the transportation area, to the best authors' knowledge, only three papers have used AutoML methods for TF [1,2,15]. The first research carried out by Vlahogianni et al. [15] 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 hyper-parameter setting. The AutoML task is performed from an algorithms base of three ML methods (Neural Network, Support Vector Machine and Radial Base Function) that forecast average speed in a time horizon of 5 min, using a regression approach. After that, Angarita et al. in [1] and [2] used Auto-WEKA, an AutoML method that applies sequential model-based bayesian optimisation [4] to find optimal ML pipelines. Both papers compared the performance of Auto-WEKA w.r.t. the general approach, which consists of selecting by trial and error the best of a set of algorithms to predict traffic. In the case of [2], the paper was centred in forecasting traffic LoS at a fixed freeway location through multiple time horizons. On the other hand, in [1], the authors were focused on predicting traffic speed on a subset of families of TF regression problems focused on making predictions at the point and the road segment levels within the freeway and urban environments.

The main differences between this research and the three aforementioned papers lay on the typology of AutoML method used and the addressed TF problems. Whereas the previous three focused on "pure" AutoML optimization approaches, in this research, we centre on a hybrid strategy based on meta-Learning, optimization and ensemble learning with the purpose of evaluating the benefits that the former has on TF within three scenarios: by its own (without optimization and ensemble learning), in combination solely with ensemble learning, and integrated to optimization plus ensemble learning.

4 Methodology

This research seeks to keep exploring the benefits that meta-learning within AutoML of hybrid search strategies can bring to TF. To accomplish such purpose, we analyse to what extent Auto-sklearn, the state-of-the-art AutoML

Table 1. Data-sets

Type	Data-sets	# Instances	# Attributes	# Instances per class	Imbalance ratios
Type I	Fw_T+CD in time horizons of 5, 15, 30, 45, 60 min	[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 min	[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 min	[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 min	[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 min	[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 min	[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 min	[2684-2634]	13	A = 1337 B = 1299	IR (A/B) = 1,02
	Ub_TS+CD in time horizons of 15, 30, 45, 60 min	[2684-2634]	28	B = 1561 A = 1122	IR (B/A) = 1,39

method for this category of search strategies, is able to recommend competitive ML pipelines for TF. In this context, the following parts of these sections are devoted to giving more details about the data-sets used for the experimentation (Sect. 4.1); and the experimental set-up of this study (Sect. 4.2).

4.1 Data-Sets

For experimentation, we considered two TF environments: freeway and urban. For the freeway environment, the data was collected from the Caltrans Performance Measurement System[1] whereas for the urban one, the data was collected from the Madrid Open Data Portal[2]. In both cases, the traffic measure used was thee months of speed in aggregation times of 5 and 15 min, respectively. For more details about the raw data used to generate the data-sets employed in this research, the interested reader is referred to [1].

Concretely, we approach two types of TF classification problems with two problem instances for each of them. In both problems, the objective is to predict

[1] http://pems.dot.ca.gov.
[2] https://datos.madrid.es/portal/site/egob/.

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[3] [11].

The first TF problem corresponds to the prediction of LoS at a target location in a freeway environment. The first instance of this problem is based only on past traffic speed data of the target location (temporal data, T); meanwhile, the second instance considers historical traffic data coming from the target location and from four downstream positions (temporal and spatial, TS). It is important to clarify that these two instances of the first TF problem are correlated because they share the same target location.

The second kind of TF problem is focused on forecasting LoS within an urban context independent of the freeway data described above. Repeatedly, the two correlated instances of this problem are: predict LoS for a single target location considering exclusively historical data of this spot, and forecasting LoS taking into account past traffic speed of the target location together with other four downstream positions.

For the two TF problems described above, we generated 36 data-sets (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 min using data granularity of 5 min (granularity means how often and how long the traffic measure is aggregated). Unlike the previous one, for the urban TF problem, the forecasting time steps are 15, 30, 45, and 60 min with data granularity of 15 min. To better identify the data-sets, they are named following the next structure: *Context_InputData_TimeHorizon*.

Attributes of the freeway and urban data-sets where the input is composed of only temporal traffic data from the target location and calendar data are: 1) Day of the week; Minute of the day, 2) Traffic speed of the objective spot at past 5, 10, 15, 20, 25, 30, 35, 40, 45 min for freeway and 15, 30, 45, 60, 75, 90, 105, 120, 135 min for urban, and 3) LoS in the target location. In the case of the freeway and urban data-sets where the input consists of historical speed taken from the target location and from four downstream detectors, the attributes are the same mentioned above for the target location and include also attributes of traffic speed of the four downstream locations at the same past times.

Table 1 presents a summary of the 36 data-sets that includes the number of instances, the number of attributes, the number of instances per class and the Imbalance Ratios (IRs) of each data-set. 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 data-sets have a different imbalance degree. Some data-sets do not contain all the possible classes because on some occasions some of the classes had an extremely low presence (e.g. 20 samples) which introduced noise in the results. Samples of these classes where tagged as classes of the closest label with the lowest number of samples. Moreover, the differences between freeway and urban data-sets of Type I and Type II

[3] Category A indicates light to moderate traffic, whereas a category E means extended delays.

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-sets.

4.2 Experimental Set-Up

Considering the traffic forecasting setting presented, we explore the performance of the Auto-sklearn's components through three scenarios using the data-sets presented above. First, a default scenario in which the hybrid search strategy of the AutoML method uses its three components to find pipelines. In this case, we considered three execution times for Auto-sklearn (ET): 15, 60 and 120 min. They correspond to the time that the bayesian optimisation can take to find the best pipelines and their hyper-parameter configuration for a given data-set. For assessing the performance of this scenario, the data-sets are partitioned in training (80%) and test (20%), keeping the chronological order of the data.

In the second scenario, we probe an alternative approach in which the recommendations done via meta-learning are combined in two ensembles based on weighted-voting without using the optimisation process. First, we extract the 25 best ML pipelines (default value used by Auto-sklearn) suggested by the meta-learning component, which then are combined in the weighted-voting ensemble named MetaEns25. To test this ensemble, the data-sets are partitioned in the same way as Auto-sklearn, such as was described above. For the second ensemble, we extract the complete list of 121 ML pipelines that can be suggested by 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 data-sets are partitioned in training (60%), validation (20%) and test (20%). To select the 25 best pipelines, these are trained on the training set and their performance is assessed on the validation set. Then, the ensemble is built with the 25 pipelines with the best validation error. Finally, the ensemble is trained on training+validation partitions (same number of instances as previous strategies, that is, 80%) and validated on the test set.

Lastly, for the third test scenario, we consider the meta-learning component in isolation. We follow a similar approach to that of MetaEn25-121, that is, we split the data-sets in training (60%), validation (20%) and test (20%). This means that for every data-set, we train the 121 pipelines suggested by meta-learning on the training set and based on their error on the validation set, 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.

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. 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.

Table 2. 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)

(*continued*)

Table 2. (*continued*)

Type	Data-set	BestPipe_Val	MetaEn25	MetaEn25-121	AutoS_15ET	AutoS_60ET	AutoS_120ET	Winner
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)

5 Results

This section presents the results obtained with the experimental set-up proposed in the previous section. Table 2 shows the mean *mGM* values obtained by the three execution times (ET) of Auto-sklearn (AutoS_ET), the two voting ensembles (MetaEns25 and MetaEn25-121) and the best pipeline in validation from the meta-learning component (BestPipe_Val). These *mGM* values were calculated by carrying out five repetitions for each approach on every data-set. *mGM* values in bold indicate the best result achieved in every data-set. Besides, the last column of Table 2 shows which is the winner approach in terms of performance on each data-set.

In the cases wherein the best performing is obtained by the *BestPipe_Val* approach, we indicate the following information: the first pair between brackets indicates the ranking position of the winner pipeline according to the similarity metric used by Auto-sklearn, and the difference between the value of this metric and the one from the pipeline in the first position of the ranking, whereas in the second tuple between brackets, the value of the metric for the pipelines in positions 1, 25 and 121 (this information appears in the same order in the column named "Winner" of Table 2). In this way, 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.

From Table 2, the following highlights can be extracted regarding the behaviour of the methods AutoML methods compared. The BestPipe_Val component is by far the best performing approach when making traffic predictions. Concretely, it is able to suggest the best pipeline in 33 out of 36 data-sets, performing even better than the longer ET (120 min) of Auto-sklearn. However, these results also show that the distance measure in which Auto-sklearn is based it is not well correlated with performance as we explain below.

If we check carefully the winner pipelines in the last column of Table 2, only in 5 cases (data-sets: 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 higher than 25. As it was stated in Sect. 2.1, this 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 data-sets against the meta-features stored in the meta-knowledge base. Considering such comparison, the similarity metric chooses the best pipelines found in the off-line Auto-sklearn's phase for each of the 25 most similar data-sets w.r.t. the one at hand. Based on results of Table 2, the meta-features used for the comparison are not working properly 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 2 are associated to data-sets, that with the current Auto-sklearn's meta-features comparison, are no being categorised as similar w.r.t the TF data-sets.

Table 3. 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 min which are the two longer execution times of Auto-sklearn

Type	Data-sets	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

For the default scenario in which Auto-sklearn uses its three components to find competitive pipelines, longer ET are supposed to improve the final results of predictions. However, the improvements only rank from 0.01 to 0.07, approximately, in the best of the cases (e.g., Fw_TS+CD_5 - Type I). This could be due to the fact that the meta-learning component is suggesting low-performance pipelines for the warm-start process of the optimisation component. Opposite to this tendency are data-sets $Fw\_TS + CD\_15$ - Type I, $Fw\_T + CD\_15$ - Type II and $Ub\_TS + CD\_15$ - Type I wherein the best mGM value is found by an ET shorter than 120m ET. We observed that this worsening is due to the over-fitting produced by the hyperparameters tuning of Auto-sklearn on the recommended pipelines. This result indicates that it is necessary to introduce mechanisms in the hybrid search strategy of AutoML to deal with over-fitting, especially when execution times of the optimisation are higher.

Regarding the performance of the two ensembles approaches based on weighted-voting (MetaEns25 and MetaEns25-121), the results of MetaEns25-121 are quite similar w.r.t. the results obtained when the optimisation component is taken into account. Concretely, in data-sets of freeway Types I-II and urban Type II, the MetaEns25-121 is able to outperform Auto-sklearn in multiple cases. In particular, in the data-sets $Fw\_TS + CD\_45$ - 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 latter ensemble is built using already optimised pipelines located beyond position 25 of the ranking, and as it was stated before, in those positions are competitive pipelines whose performance is boosted by the ensemble without the need of doing optimisation. For the case of MetaEns25, its performance is lower than MetaEns25-121 and the three ET of Auto-sklearn. However, it is interesting to note that these ensembles are not better than the best pipeline suggested via meta-learning; in this sense, it would be interesting to explore why the ensembles obtains a performance worse than the best pipeline in isolation.

As the computational cost is a key factor in AutoML, Table 3 shows the execution times in minutes that the BestPipe_Val and the two meta-ensembles took to make predictions on every data-set. As can be seen, in the majority of the cases, the three approaches spent less than 60 min, which is the second longer ET of Auto-sklearn.

Finally, additional results that are observed regardless of what approach is the one with the highest mGM values are discussed below. In the cases of data-sets freeway Types I-II and urban Type I, as the time horizon of predictions increases, the performance of all approaches decreases. For these data-sets, the ones that have a time horizon of five minutes are the TF problems in which the six approaches perform better. Besides, in data-sets $Fw\_TS + CD\_30$ and $Fw\_TS + CD\_60$, most of them have problems predicting the minorities classes and therefore their mGM values in these cases are equal to zero.

Regarding urban data-set 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 due to the fact that these data-sets have the highest IRs (IR(A/B) = 1.12

IR(A/C) = 12.04). This demonstrates that Auto-sklearn does not incorporate in its inner structure mechanisms to deal with high imbalanced classification data-sets. Meanwhile, in the case of urban data-sets of Type I with spatial and temporal data (TS) and all urban data-sets of Type II, the performance of the six approaches is quite acceptable and homogeneous across them. This behaviour can be argued as these 12 data-sets are the most balanced of the 36 data-sets (IR(B/A) = 2.40, IR(B/C) = 4.98; IR(A/B) = 1.02, IR(B/A) = 1.39).

6 Conclusions

In this paper, we studied the benefits in terms of performance and computational cost of hybrid AutoML for TF. We use Auto-sklearn, a state-of-the-art hybrid AutoML method whose search strategy of pipelines uses bayesian optimisation, meta-learning and ensemble learning. We focused on how well Auto-sklearn is able to recommend competitive ML pipelines to forecast traffic, modelled as a multi-class imbalanced classification problem, along different time horizons in urban and freeway environments.

From the results, we drew interesting conclusions. A simple approach based on estimating the best pipeline from Auto-sklearn's meta-learning component is able to suggest competitive pipelines that perform better than the results obtained by the three ET of Auto-sklearn considered and the two weighted-voting ensembles. However, these winner pipelines usually were not included in the 25 suggestions done by default by the Auto-sklearn's meta-learning component. Instead, they were located in lower positions, which could lead to thinking that the meta-features and the similarity metric in charge of recommending pipelines are not performing as expected for these data-sets. As a result, the ranking positions are not directly related to the performance that the pipelines could have on the TF data-sets.

Another interesting conclusion is that the optimisation component is not adding too much to the final *mGM* values. Higher execution times for Auto-sklearn not always lead to better results as we can expect; this was also corroborated by previous research that approached the use of Auto-WEKA (another AutoML method) for TF [1,2]. In spite of this, the performance of the optimisation process could be improved if the ranking recommended by the meta-learning component was re-organized using the validation error of these pipelines on the input data. Thus, the optimisation would only be fed by pipelines that already are corroborated for having high performance on the data-set at hand. However, caution needs to be taken to check the computational cost consumed when calculating the validation error of the 121 pipelines in the meta-knowledge base.

Further research lines that we aim to explore in the future are: I) improving the synergy between meta-learning and ensemble learning; II) determining the TF problems in which the optimisation is strictly necessary to improve the results obtained via meta-learning.

Acknowledgments. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No. 815069 and the Marie Sklodoska-Curie grant agreement No. 665959.

References

1. Angarita-Zapata, J.S., Masegosa, A.D., Triguero, I.: Evaluating automated machine learning on supervised regression traffic forecasting problems. In: Llanes Santiago, O., Cruz Corona, C., Silva Neto, A.J., Verdegay, J.L. (eds.) Computational Intelligence in Emerging Technologies for Engineering Applications. SCI, vol. 872, pp. 187–204. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-34409-2_11
2. Angarita-Zapata, J.S., Triguero, I., Masegosa, A.D.: A preliminary study on automatic algorithm selection for short-term traffic forecasting. In: Del Ser, J., Osaba, E., Bilbao, M.N., Sanchez-Medina, J.J., Vecchio, M., Yang, X.-S. (eds.) IDC 2018. SCI, vol. 798, pp. 204–214. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99626-4_18
3. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, pp. 2962–2970. Curran Associates, Inc. (2015)
4. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25566-3_40
5. Hutter, F., Kotthoff, L., Vanschoren, J. (eds.): Automated Machine Learning: Methods, Systems, Challenges. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5
6. Kerschke, P., Hoos, H., Neumann, F., Trautmann, H.: Automated algorithm selection: survey and perspectives. CoRR (2018)
7. Lemke, C., Budka, M., Gabrys, B.: Metalearning: a survey of trends and technologies. Artif. Intell. Rev. **44**(1), 117–130 (2013). https://doi.org/10.1007/s10462-013-9406-y
8. Luo, G.: A review of automatic selection methods for machine learning algorithms and hyper-parameter values. Netw. Model. Anal. Health Inform. Bioinform. **5**(1), 18 (2016). https://doi.org/10.1007/s13721-016-0125-6
9. Mohr, F., Wever, M., Hüllermeier, E.: ML-Plan: automated machine learning via hierarchical planning. Mach. Learn. **107**(8), 1495–1515 (2018). https://doi.org/10.1007/s10994-018-5735-z
10. Olson, R.S., Bartley, N., Urbanowicz, R.J., Moore, J.H.: Evaluation of a tree-based pipeline optimization tool for automating data science. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, pp. 485–492 (2016)
11. Skycomp, I.B.M.: Major High- way Performance Ratings and Bottleneck Inventory. Maryland State Highway Administration, the Baltimore Metropolitan Council and Maryland Transportation Authority, State of Maryland (2009)
12. Swearingen, T., Drevo, W., Cyphers, B., Cuesta-Infante, A., Ross, A., Veeramachaneni, K.: ATM: a distributed, collaborative, scalable system for automated machine learning. In: 2017 IEEE International Conference on Big Data, pp. 151–162 (2017)

13. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA. In: Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining, pp. 847–855 (2013)
14. Vanschoren, J.: Meta-learning: a survey. arXiv preprint arXiv:1810.03548 (2018)
15. Vlahogianni, E.I.: Optimization of traffic forecasting: intelligent surrogate modeling. Transp. Res. Part C Emerg. Technol. **55**, 14–23 (2015). http://www.sciencedirect.com/science/article/pii/S0968090X15000959. Engineering and Applied Sciences Optimization (OPT-i) - Professor Matthew G. Karlaftis Memorial Issue
16. Vlahogianni, E.I., Karlaftis, M.G., Golias, J.C.: Short-term traffic forecasting: where we are and where we're going. Transp. Res. Part C: Emerg. Technol. **43**, 3–19 (2014)
17. Yao, Q., et al.: Taking human out of learning applications: a survey on automated machine learning. CoRR (2018)
18. Zöller, M.A., Huber, M.F.: Survey on automated machine learning. CoRR (2019)