

A Benchmark of Automated Multivariate Time Series Forecasting Tools for Smart Cities

Pedro José Pereira^{1,2}, Nuno Costa², Pedro Mestre^{3,4}, and Paulo Cortez^{1,2}

¹ EPMQ - IT Engineering Maturity and Quality Lab, CCG ZGDV Institute,
Guimarães, Portugal

`pedro.pereira@ccg.pt`

² ALGORITMI Center/LASI, Dep. Information Systems, University of Minho,
Guimarães, Portugal

`a89167@alunos.uminho.pt`, `pcortez@dsi.uminho.pt`

³ CEIIA – Centre of Engineering and Product Development, Matosinhos, Portugal

⁴ ALGORITMI Center, Department of Engineering, School of Sciences and
Technology, University of Trás-os-Montes e Alto Douro, Vila Real, Portugal

`pedro.mestre@ceiia.com`

Abstract. Most Smart Cities data come from multiple related sensors. Within this context, multivariate Automated Time Series Forecasting (AutoTSF) tools are valuable for providing predictive analytics for citizens and city rulers. In this paper, we benchmark seven multivariate open-source AutoTSF tools (AutoARIMAX, AutoGluon, FlaML, AutoTS, MFEDOT and HyperTS) and one univariate AutoTSF tool (FE-DOT), measuring both their predictive performances, as well as their computational effort. The tools were evaluated by using four real-world multivariate datasets that were recently collected from a Portuguese city under a realistic rolling window scheme. Overall, the AutoGluon and AutoTS tools presented the best predictive performances, with AutoGluon requiring a substantially reduced training computational effort when compared with AutoTS.

Keywords: Automated Machine Learning · Multivariate Time Series Forecasting · Smart Cities

1 Introduction

Due to the growth of urban populations and recent advances in Information Technology, city rulers are increasingly adopting digital solutions to support their decision-making process, in what is termed as Smart Cities. Recent advances in Information Technology (IT), Artificial Intelligence (AI) and Machine Learning (ML), are leveraging the Smart Cities concept, making it easier to collect, store, process and analyze data. In effect, multiple research studies have been recently published and that address data-driven ML-based Decision Support Systems (DSS) to solve several Smart Cities challenges (e.g., predict energy consumption or bike sharing usage) [6, 18, 19]. Often, Smart Cities data is regularly collected over time (e.g., atmospheric sensors, traffic cameras), therefore

assuming a time series format [14]. Within this context, Time Series Forecasting (TSF) is a valuable component for Smart Cities DSS.

A time series consists of a sequence of continuous observations over time, chronologically ordered, and can be classified into two main types: univariate and multivariate. The former assumes that a single event is observed (e.g., number of vehicles intersecting a crossroad), while the latter considers that multiple events are observed simultaneously and may influence each other (e.g., parking lots occupation located at the same street) [21]. Several TSF algorithms have been proposed for each type of time series and they can be grouped into two main classes: conventional statistical methods and ML algorithms [5, 10]. The most popular statistical method is the Auto-Regressive Integrated Moving Average (ARIMA) [3]. Proposed in the 70s, it has been widely used for TSF tasks. Yet, it is rather rigid and presents limitations when modeling complex nonlinear relationships. More recently, several more flexible ML models have been successfully applied to TSF tasks, such as Deep Learning (DL) architectures [19, 25] and Automated ML (AutoML) tools [2, 15, 18].

With the current growing demand for ML, the need for easily and rapidly implementable AI solutions increases. Furthermore, since the development and implementation of such solutions often involves specific ML knowledge and a time-consuming empirical process (e.g., for hyperparameter tuning), the need of ML experts also increases. In particular, DL models often require a complex modeling phase, requiring the setup of a large number of hyperparameters (e.g., number of hidden layers, optimization function), requiring the training of high number of candidate deep architectures. Within this context, the AutoML emerges as an interesting approach, by allowing the automation of several steps of the ML process, such as the model selection and hyperparameter optimization steps. When applied to DL, the term is often known as AutoDL or Neural Architecture Search (NAS). Moreover, AutoML tools are more accessible to be used by non-expert users [15].

Several AutoML tools have been proposed focusing on different types of tasks and algorithms, such as: classification, regression and AutoDL [7]; and Automated TSF (AutoTSF) [15]. Given this recent proliferation of AutoML technology, it becomes relevant to empirically compare these tools, aiming to evaluate which tool tends to perform better in one or more real-world domains. In recent previous work [14], we performed such empirical study by comparing 8 univariate open-source AutoTSF tools (Pmdarima, Prophet, Ludwig, DeepAR, TFT, FEDOT, AutoTs and Sktime) using nine public domain time series that could be associated with a Smart Cities context (e.g., temperature, energy consumption and city traffic data). In this paper, we extend this work by benchmarking multivariate AutoTSF tools for the Smart Cities domain. In particular, we present a thorough comparison of 7 multivariate open-source AutoTSF tools, namely: AutoARIMAX, AutoGluon, AutoTS, FlaML, MFEDOT, HyperTS and VAR. We note that MFEDOT method corresponds to a novel multivariate adaptation (performed in this work) to the univariate TSF FEDOT tool [11]. Moreover, we compare the 7 multivariate tools with the FEDOT univariate AutoTSF ap-

proach, since it obtained the best univariate AutoTSF results in [14]. As for the benchmark study, it assumes 4 multivariate datasets collected between 2022 and 2023 from a Portuguese city, aiming to contribute for a green transition. The comparison assumes a robust and realistic rolling window evaluation [4], considering both the predictive performance and the computational effort.

This paper is organized as follows. Section 2 presents the related work. Then, the Section 3 details the time series data used, the AutoTSF methods and the comparison procedure. Next, in Section 4, we present and analyze the obtained results. Finally, Section 5 discusses the final conclusions and future work.

2 Related Work

Table 1 summarizes the related works, considering the following dimensions: **Study** - reference of the study; **Scope** - if the study presented a new AutoTSF tool or performed a benchmark of existing ones; **Best Model** - the best performing model; **SC** - if at least one Smart Cities dataset was used; **Uni.** - if univariate TSF tasks were addressed; **Multi.** - if multivariate TSF tasks were addressed; **CE** - if the Computational Effort was considered; **#tools** - the number of compared AutoTSF tools.

Table 1. Summary of related works involving AutoTSF tools.

Study	Scope	Best Model	SC	Uni.	Multi.	CE	#autoTSF tools
[27]	New tool	VAR			✓		0
[20]	New tool	Prophet		✓			1
[8]	New tool	DeepAR	✓	✓			0
[22]	Benchmark	Random Forest		✓			1
[9]	New tool	TFT	✓	✓			1
[16]	New tool	AutoAI-TS	✓	✓	✓	✓	4
[13]	Benchmark	Traditional models	✓	✓	✓	✓	0
[24]	New tool	AutoTS	✓		✓		0
[12]	New tool	FEDOT		✓		✓	2
[17]	New tool	AutoGluon	✓	✓		✓	6
[14]	Benchmark	FEDOT	✓	✓		✓	7
This work	Benchmark	AutoGluon/AutoTS	✓		✓	✓	7

Some research studies proposed new multivariate AutoTSF tools without comparing them with other AutoTSF tools, such as: Vector Auto-Regressive (VAR) model [27], DeepAR [8], Fast and Lightweighted AutoML Library (FLAML) [23], Automatic Time Series (AutoTS) [24] and HyperTS [26]. There are also research works that present novel AutoTSF tools and perform a comparison with

other AutoTSF approaches, namely: Prophet [20], Temporal Fusion Transformers (TFT) [9], AutoAI-TS [16], FEDOT tool [11] and AutoGluon [17].

As for pure benchmark TSF studies, an univariate comparison including one AutoTSF tool (AutoARIMA) was performed in [22] for conjunctivitis disease data. In 2021, traditional TSF methods were compared with regression AutoML frameworks (e.g., H2O) [13]. Yet, the AutoML usage requires a prior manual data processing step, where the time series are transformed into tabular regression data by using lagged entries. Thus, no direct AutoTSF tool (in the sense that it automatically creates a TSF model and forecasts from time series data) was used in the comparison. More recently, we proposed a benchmark study with a comparison of 7 AutoTSF tools, using public datasets related to Smart Cities [14], but it was limited only to univariate TSF tasks. Within our knowledge, there are no benchmarks comparing AutoTSF methods on multivariate TSF tasks. In this work, we address this research gap, focusing on the Smart Cities domain. Similarly to [13], we analyze both the predictive and computational effort performance of the compared methods.

3 Materials and Methods

3.1 Multivariate Time Series Data

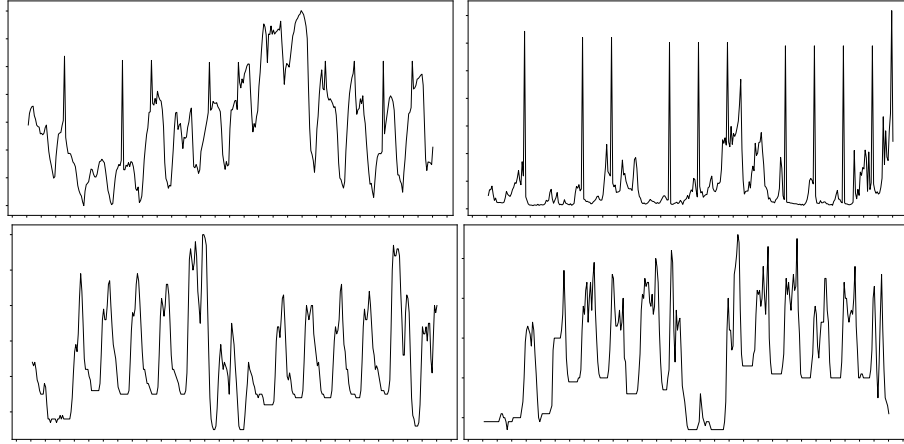
A multivariate time series corresponds to a set of time ordered simultaneous observations of interrelated events (e.g., atmosphere temperature and humidity). Thus, the multivariate TSF task assumes that $e \times H$ predictions are performed at time t (last known value), with e being the number of events and H the forecasting horizon.

In our experimentation, we used 4 datasets from the Smart Cities domain. The first three are about atmospheric conditions measured at three different city locations, while the last dataset concerns the parking lots occupation. Each atmospheric dataset (atm.) results from the aggregation of data from multiple sensors closely positioned, each containing different attributes from the following list: **T** - temperature; **RH** - relative humidity; **P** - pressure; **PM_x** - particulate matter, with x in $\{1.0, 2.5, 10\}$; and **noise** - regarding audio noise, measured in dB. As for the parking lots occupation dataset (parks), it results from the aggregation of data from 4 different parking lots located at the city center. Table 2 describes the details of each multivariate dataset, including its time series length (L), seasonal period (K) and the adopted rolling window parameters, namely: training window size (W), step (S) and horizon (H).

All the data were collected on an hourly basis, between 2022 and 2023. Thus, we defined $H = 24$, which corresponds to hourly predictions for one day ahead. Similarly to [14], we performed the visual inspection of the observed values and its autocorrelations, confirming the seasonal periods of one day ($K = 24$ hours) for all the time series. For demonstration purposes, Fig. 1 plots some of the analyzed time series. As the number of observed events, it corresponds to: $e = 6$ for atm.1, $e = 3$ for atm.2 and $e = 4$ for both atm.3 and parks.

Table 2. Description of the time series datasets.

Context	Time Series	Time period	Target	<i>L</i>	<i>K</i>	<i>W</i>	<i>S</i>	<i>H</i>
Atmosphere	atm.1	Jan. to Feb. 2022	T	953	24	360	28	24
			RH					
			P					
Atmosphere	atm.2	Jan. to Apr. 2022	PM1.0	2610	24	720	93	24
			PM2.5					
			PM10					
Atmosphere	atm.3	Jan. to Apr. 2022	T	2394	24	720	82	24
			RH					
			P					
Parking Lots Occupation	parks	Sep. 2022 to Jan. 2023	Noise	3216	24	1440	87	24
			park 1					
			park 2					
Parking Lots Occupation	parks	Sep. 2022 to Jan. 2023	park 3	3216	24	1440	87	24
			park 4					

**Fig. 1.** Examples of time series data: top left – atm.1 RH; top right – atm.2 PM2.5; bottom left – park2; and bottom right – park3.

3.2 AutoTSF Methods

We consider a tool to perform an AutoTSF if it is capable of directly automating the creation of TSF models and forecasts from raw time series data. In this

work, we benchmark 7 open-source Python AutoTSF tools for multivariate TSF tasks. Table 3 summarizes the analyzed tools in terms of: **Name**, publication **Year**, **Type**, **Optimization** method used for TSF model search (when known), **License** and **Training Mode**. In terms of the tool type, we considered: one Auto-Regressive - VAR [27]; one Auto-Arima - AutoARIMAX [1]; and 5 other methods - AutoGluon [17], AutoTS [24], MFEDOT (multivariate extension of FEDOT [11]), HyperTS [26] and FlaML [23]. In order to maintain a fair comparison, we assumed the default parameters of each implementation. Yet, due to computational constraints, whenever possible, we limited the models execution time. To do so, we either set a time limit, or chose a fast execution mode option, or performed the process of model selection and hyperparameter tuning only once, during the first rolling window iteration, similarly to what was executed in [14]. This last option, which we termed as Update training mode, fixes the model and its hyperparameters after the first run of the rolling window evaluation, only fitting the model to newer data (updating it) in the remaining ones, thereby significantly reducing the computational effort.

Table 3. Description of the Multivariate AutoTSF algorithms.

Name	Year	Type	Opt.	License	Training Mode
VAR	2006	Auto-Regressive	-	BSD-3-Clause	Train
AutoARIMAX	2017	Auto-Arima	-	MIT	Train
AutoGluon	2021	AutoTSF	-	Apache 2.0	Train
FlaML	2021	AutoTSF	-	MIT	Train
AutoTS	2022	AutoTSF	GA	MIT	Train
MFEDOT	2022	AutoTSF	EA	BSD-3-Clause	Update
HyperTS	2022	AutoTSF	-	Apache 2.0	Train

Regarding the implementation and configuration of AutoTSF tools, we used:

- *AutoARIMAX*: the multivariate ARIMA version based on the Auto-Arima implementation of the `pmdarima` Python module. It predicts each target separately, using the remaining ones as the exogenous variables [1].
- *AutoGluon*: a tool that allows the simultaneous prediction of multiple time series, based on its previous values and other related co-variables [17]. It combines multiple state-of-the-art algorithms from the `gluonTS` package, such as ETS, ARIMA, TFT and DeepAR. For its implementation, we defined a time limitation of 900 s (15 min.).
- *AutoTS*: uses a Genetic Algorithm (GA) for the automatic design of TSF models [24]. We used `autots` python module for its implementation and set the list of used models and data transformations as “all”.
- *MFEDOT*: extension of the univariate FEDOT implementation [11], which uses an Evolutionary Algorithm for the design of ML pipelines. For the multivariate implementation, we used the `fedot` module and, similarly to AutoARIMAX, predicted each target separately, using the remaining ones

- as exogenous variables. The package default parameters were assumed, with the exception of a training time limitation of 15 min., as performed for AutoTSF. Finally, we used the update training mode, which builds and fits the pipeline only during the first rolling window iteration, fixing it and then updating it to newer data in the remaining iterations, as performed in [14].
- *FlaML*: based on the `flaml` python tool that automates several steps of the ML process (e.g., model selection and tuning) [23]. This method uses multiple exogenous variables to forecast each target individually.
 - *HyperTS*: it is a full pipeline automated time series toolkit, supporting forecasting, classification and anomaly detection implementations, automating multiple steps of the ML process (e.g., data preprocessing, feature engineering, model selection) [26]. We used the `hyperts` module with its default parameters.
 - *VAR*: it is a TSF model from the `statsmodels` python package that allows the analysis and forecasting of multivariate time series [27].

For comparison purposes, the FEDOT univariate AutoTSF tool is also included in the empirical results. The tool was configured similarly to MFEDOT, creating a distinct univariate TSF model for all e event targets of a multivariate dataset.

3.3 Evaluation

To evaluate the AutoTSF methods, we used a robust and realistic Rolling Window (RW) procedure with a total of $U = 20$ training and testing iterations over time, similarly to what was executed in [14]. In each iteration, the data is chronologically divided into a training set of fixed size W , which was used to fit the AutoTSF models, and a test set of size H , for which forecasts were performed. The first RW iteration uses the oldest data to train the model and the following H -ahead observations to evaluate it. In the next iteration, the window rolls a step of S values, discarding the oldest S observations from the training data and updating it with the S new records. This process continues until the maximum of $U = 20$ iterations are reached. Table 2 presents the selected values for L , W , S and H . Considering each series length (L), we set $W = 360$ hours (15 days) for atm.1, $W = 720$ hours (30 days) for atm.2 and atm.3, and $W = 1440$ (60 days) for parks. These values mean that the model will learn from 15, 30 or 60 days of data, depending on the dataset, and forecast the following $H=24$ observations. The S values were defined according to the formula $S = (L - (W + H - 1))/U$ [14], in order to achieve the $U = 20$ maximum value.

We considered two aspects when evaluating the tools: the predictive performance and the computational effort. For the predictive performance, we used the Normalized Mean Absolute Error (NMAE) metric: $NMAE = \frac{MAE}{y_{max} - y_{min}}$, with $MAE = \frac{\sum_{i=1}^H |y_{t+i} - \hat{y}_{t+i}|}{H}$, where y_{t+i} denotes the target values, $\hat{y}_{t+i}$ the predictions (made at time t for the i -th ahead step), and y_{min} and y_{max} the series minimum and maximum values, respectively. The NMAE has the advantage of being scale independent, being also expressed as a percentage error. The

lower the value, the better the forecasts. The NMAE measure was individually computed for each predicted event (target). In terms of computational effort, we measured both the training and inference (prediction) time for each tool.

4 Results

All the experiments conducted were developed using Python code that executed in a Linux Intel Xeon 2.0 Ghz server. Table 4 presents the obtained median NMAE values regarding the $U=20$ RW iterations and all multivariate targets (e.g., T and RH for atm.1). For each dataset, we computed two overall measures (for all e events): the average and the median NMAE values for each tool and dataset. For instance, for the atm.1 dataset, the univariate implementation of FEDOT presented the best e median NMAE value (3.88), followed by HyperTS (4.03). As for the average e NMAE values, the HyperTS presented the best score (5.26), followed by AutoTS (5.41). For this dataset, the AutoTS tool obtained 3 wins (best results for the RH, PM1.0 and PM2.5 targets).

Concerning the atm.2 dataset results, AutoGluon achieved 2 wins (T, RH). Yet, the best median e NMAE results were obtained by AutoTS (6.45), followed by FEDOT (6.49) and AutoGluon (6.90). The univariate version of FEDOT has the lowest averaged NMAE for these series (7.55), AutoTS has the second lowest (7.58) and AutoGluon the third (7.78). As for the worst results, VAR achieve both the average (12.85) and the median (15.05) higher NMAE values.

Regarding the atm.3 dataset, the univariate FEDOT achieved the best median NMAE value (8.31), followed by FlaML (8.64) and HyperTS (8.64), while the best average NMAE were obtained by AutoTS, FlaML and MFEDOT with 7.59, 7.78 and 7.83, respectively. Again, VAR obtained the last place for both aggregated NMAE results. In terms of wins, each of the following models achieved 1: FEDOT (T), MFEDOT (RH), AutoTS (P) and FlaML (Noise).

Finally, for the parks dataset, AutoGluon achieved the best predictive performance in both the aggregated values and a total of 3 (out of 4) wins (parks 1, 2 and 3), while AutoARIMAX accomplished the remaining win (park 4). AutoTS and HyperTS achieved second and third places, respectively, in both aggregated metrics. As for the worst performance, in terms of average NMAE, it belongs to MFEDOT (15.20), and in terms of median NMAE it belongs to VAR (17.88).

The aggregated AutoTSF tool results are presented in Table 5, in terms of: the **Median** and Average (**Avg.**) **NMAE (%)** value, as computed when considering all individual time series targets (total of 17 series); the number of “wins” (**#wins**), the number of times a tool outperformed all the remaining ones for a given target when considering the median NMAE RW values; the computational effort, computed in terms of the averaged training (**Train**) and forecasting (**Inference**) times, measured in seconds and in milliseconds, respectively. The top 3 tools, in terms of best overall median NMAE results, are AutoGluon (7.54), FEDOT (7.56) and AutoTS (8.26). On the other hand, if we consider the overall averaged NMAE values, AutoTS was the best method with 7.09, followed by AutoGluon (7.25) and HyperTS (7.51). In both scenarios, VAR is ranked

Table 4. Comparison of the median NMAE (%) RW results (best values are in **bold**).

Dataset	Target	Auto ARIMAX	Auto Gluon	Auto TS	MFedot	FlaML	Hyper TS	VAR	Fedot
atm.1	T	20.70	10.00	11.43	8.41	9.29	10.74	18.06	8.96
	RH	15.78	11.90	9.84	13.36	12.49	11.59	13.90	15.35
	P	6.99	9.65	7.76	6.14	8.20	5.28	4.39	4.82
	PM1.0	4.02	2.76	2.51	3.65	3.14	2.77	3.78	2.93
	PM2.5	1.34	0.79	0.76	1.64	0.76	0.97	1.29	0.78
	PM10	0.45	0.14	0.16	0.32	0.17	0.19	0.26	0.21
	Median	5.51	6.21	5.14	4.90	5.67	4.03	4.09	3.88
	Avg.	8.21	5.87	5.41	5.59	5.68	5.26	6.95	5.51
atm.2	T	13.57	6.19	6.45	10.53	9.02	8.02	15.05	6.49
	RH	12.82	10.25	11.15	11.31	15.64	12.84	16.10	11.92
	P	4.56	6.90	5.14	5.75	6.86	4.56	7.40	4.24
	Median	12.82	6.90	6.45	10.53	9.02	8.02	15.05	6.49
	Avg.	10.32	7.78	7.58	9.20	10.51	8.47	12.85	7.55
atm.3	T	10.99	10.28	8.26	9.17	9.75	8.29	11.31	7.56
	RH	13.48	12.07	11.97	11.08	13.67	13.38	15.71	17.88
	P	0.58	0.12	0.11	0.63	0.18	0.31	0.33	0.26
	Noise	26.12	14.03	10.00	10.43	7.52	9.38	26.34	9.06
	Median	12.24	11.18	9.13	9.80	8.64	8.84	13.51	8.31
	Avg.	12.79	9.13	7.59	7.83	7.78	7.84	13.42	8.69
parks	park1	19.86	7.54	9.54	27.85	17.18	15.59	19.21	15.54
	park2	17.03	7.69	10.10	16.43	12.44	10.40	16.92	11.04
	park3	17.53	7.50	12.34	11.66	12.43	10.20	18.83	11.04
	park4	1.80	5.42	2.98	4.86	2.36	3.11	4.65	2.07
	Median	17.28	7.52	9.82	14.05	12.44	10.30	17.88	11.04
	Avg.	14.06	7.04	8.74	15.20	11.10	9.83	14.90	9.92

at the last position, achieving the highest NMAE aggregated results. Yet, in terms of computational effort, VAR earned the first place with an impressive 0.004 s mean training time and 0.008 ms mean inference time. AutoTS achieved the worst training time, requiring on average almost 30 minutes to fit the TSF model, while AutoGluon is the slower to perform predictions, taking in average 131.545 ms per forecast. In terms of wins, AutoGluon has 6, followed by AutoTS with 4 wins. Lastly, the proposed MFEDOT variant has performed worst than the univariate one, both in terms of median and average NMAE, and in terms of computational effort.

5 Conclusions

In the last few years, there is a growing interest in the AutoML concept, due to its ability to automate several steps of the ML process (e.g., data preprocessing,

Table 5. Overall result comparison (best values in **bold**)

ML model	Median / Avg. NMAE (%)	#wins	Train time (s)	Inference time (ms)
AutoARIMAX	12.82 / 11.04	1	55.528	1.055
AutoGluon	7.54 / 7.25	6	356.430	131.545
AutoTS	8.26 / 7.09	4	1766.358	120.445
MFEDOT	9.17 / 9.01	2	269.620	19.798
FlaML	9.02 / 8.30	1	25.965	13.603
HyperTS	8.29 / 7.51	0	24.858	1.230
VAR	13.90 / 11.38	1	0.004	0.008
FEDOT (uni.)	7.56 / 7.66	2	219.59	11.76

model selection, hyperparameter optimization). However, several AutoML tools have been proposed for distinct ML tasks (e.g., classification, TSF), thus the selection of the best tool for a particular task is non trivial process.

In this paper, we present an AutoTSF tool empirical benchmark study focused on multivariate time series datasets from the Smart Cities domain. In particular, we compare 7 open-source multivariate solutions: AutoARIMAX, AutoGluon, AutoTS, MFEDOT (a multivariate extension of FEDOT), FlaML, HyperTS and VAR. For comparison purposes, we also test the univariate FEDOT tool, which outperformed other univariate AutoTSF tools in [14].

The benchmark study included a large set of computational experiments, using 17 time series from four multivariate Smart Cities datasets and a realistic rolling window evaluation. The obtained results position the AutoGluon and AutoTS tools as the best options, with quite competitive results. In particular, from a predictive performance perspective, AutoGluon achieved the lowest overall median NMAE, while AutoTS obtained the lowest overall averaged NMAE, both ranging between 7% to 8%. In terms of computational effort, AutoTS requires a substantial increased training effort (around 5 times higher) when compared with the AutoGluon tool. Given that AutoGluon also obtained a higher number of wins (6 best individual time series results while AutoTs achieved 4 wins), we recommend AutoGluon as an interesting overall solution for Smart Cities multivariate AutoTSF.

In future work, we intend to enlarge the empirical comparison by considering more multivariate AutoTSF tools and datasets from the Smart Cities domain. Moreover, we wish to study ensemble approaches (e.g., Stacking), in which the best forecasts returned by 2 or more AutoTSF tools are combined.

Acknowledgments. This work has been supported by the European Union under the NextGenerationEU, through a grant of the Portuguese Republic’s Recovery and Resilience Plan (PRR) Partnership Agreement, within the scope of the project BE.NEUTRAL – “Agenda da Mobilidade para a neutralidade carbónica das cidades” (Project ref. nr. 35 - C644874240-00000016).

Bibliography

- [1] Alghamdi, T., Elgazzar, K., Bayoumi, M., Sharaf, T., Shah, S.: Forecasting traffic congestion using ARIMA modeling. In: IWCMC. pp. 1227–1232. IEEE (2019)
- [2] Alsharef, A., Aggarwal, K., Sonia, Kumar, M., Mishra, A.: Review of ml and automl solutions to forecast time-series data. *Archives of Computational Methods in Engineering* **29**(7), 5297–5311 (2022)
- [3] Box, G.E.: *Gm jenkins time series analysis: Forecasting and control*. San Francisco, Holden-Day (1970)
- [4] Cortez, P., Matos, L.M., Pereira, P.J., Santos, N., Duque, D.: Forecasting store foot traffic using facial recognition, time series and support vector machines. In: Graña, M., López-Guede, J.M., Etxaniz, O., Herrero, Á., Quintián, H., Corchado, E. (eds.) *International Joint Conference SOCO’16-CISIS’16-ICEUTE’16 - San Sebastián, Spain, October 19th-21st, 2016, Proceedings. Advances in Intelligent Systems and Computing*, vol. 527, pp. 267–276 (2016)
- [5] Divina, F., Garcia Torres, M., Gomez Vela, F.A., Vazquez Noguera, J.L.: A comparative study of time series forecasting methods for short term electric energy consumption prediction in smart buildings. *Energies* **12**(10), 1934 (2019)
- [6] Fernandes, G., Oliveira, N., Cortez, P., Mendes, R.: A realistic scooter rebalancing system via metaheuristics. In: Coello, C.A.C. (ed.) *GECCO ’20: Genetic and Evolutionary Computation Conference, Companion Volume, Cancún, Mexico, July 8-12, 2020*. pp. 265–266. ACM (2020). <https://doi.org/10.1145/3377929.3389905>, <https://doi.org/10.1145/3377929.3389905>
- [7] Ferreira, L., Cortez, P.: Autooc: Automated multi-objective design of deep autoencoders and one-class classifiers using grammatical evolution. *Appl. Soft Comput.* **144**, 110496 (2023)
- [8] Flunkert, V., Salinas, D., Gasthaus, J.: Deepar: Probabilistic forecasting with autoregressive recurrent networks. *CoRR* **abs/1704.04110** (2017)
- [9] Lim, B., Arik, S.Ö., Loeff, N., Pfister, T.: Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* **37**(4), 1748–1764 (2021)
- [10] Meisenbacher, S., Turowski, M., Phipps, K., Rätz, M., Müller, D., Hagenmeyer, V., Mikut, R.: Review of automated time series forecasting pipelines. *WIREs Data Mining Knowl. Discov.* **12**(6) (2022)
- [11] Nikitin, N.O., Vychuzhanin, P., Sarafanov, M., Polonskaia, I.S., Revin, I., Barabanova, I.V., Maximov, G., Kalyuzhnaya, A.V., Boukhanovsky, A.: Automated evolutionary approach for the design of composite machine learning pipelines. *Future Gener. Comput. Syst.* **127**, 109–125 (2022)
- [12] Nikitin, N.O., Vychuzhanin, P., Sarafanov, M., Polonskaia, I.S., Revin, I., Barabanova, I.V., Maximov, G., Kalyuzhnaya, A.V., Boukhanovsky, A.: Automated evolutionary approach for the design of composite machine learning pipelines. *Future Gener. Comput. Syst.* **127**, 109–125 (2022)

- [13] Paldino, G.M., De Stefani, J., De Caro, F., Bontempi, G.: Does automl outperform naive forecasting? *Engineering proceedings* **5**(1), 36 (2021)
- [14] Pereira, P.J., Costa, N., Barros, M., Cortez, P., Durães, D., Silva, A., Machado, J.: A comparison of automated time series forecasting tools for smart cities. In: *EPIA. Lecture Notes in Computer Science*, vol. 13566, pp. 551–562. Springer (2022)
- [15] Pereira, P.J., Gonçalves, C., Nunes, L.L., Cortez, P., Pilastri, A.L.: AI4CITY - an automated machine learning platform for smart cities. In: *SAC*. pp. 886–889. ACM (2023)
- [16] Shah, S.Y., Patel, D., Vu, L., Dang, X., Chen, B., Kirchner, P., Samulowitz, H., Wood, D., Bramble, G., Gifford, W.M., Ganapavarapu, G., Vaculín, R., Zerefos, P.: Autoai-ts: Autoai for time series forecasting. In: *SIGMOD Conference*. pp. 2584–2596. ACM (2021)
- [17] Shchur, O., Türkmen, A.C., Erickson, N., Shen, H., Shirkov, A., Hu, T., Wang, B.: Autogluon-timeseries: Automl for probabilistic time series forecasting. In: *AutoML. Proceedings of Machine Learning Research*, vol. 224, pp. 9/1–21. PMLR (2023)
- [18] Soares, D., Pereira, P.J., Cortez, P., Gonçalves, C.: A comparison of automated machine learning tools for predicting energy building consumption in smart cities. In: *EPIA (1). Lecture Notes in Computer Science*, vol. 14115, pp. 311–322. Springer (2023)
- [19] Sun, L., Zhong, Z., Zhang, C., Zhang, Y., Wu, D.: Tess: multivariate sensor time series prediction for building sustainable smart cities. *ACM Transactions on Sensor Networks* (2022)
- [20] Taylor, S.J., Letham, B.: Forecasting at scale. *PeerJ Prepr.* **5**, e3190 (2017)
- [21] Torres, J.F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., Troncoso, A.: Deep learning for time series forecasting: A survey. *Big Data* **9**(1), 3–21 (2021)
- [22] Verma, S., Sharma, N., Sharma, K.P.: Comparative analysis of time series models for the prediction of conjunctivitis disease. In: *Proceedings of the International Conference on Advances in Electronics, Electrical & Computational Intelligence (ICAEEC)* (2019)
- [23] Wang, C., Wu, Q., Weimer, M., Zhu, E.: FLAML: A fast and lightweight automl library. In: *MLSys. mlsys.org* (2021)
- [24] Wang, C., Chen, X., Wu, C., Wang, H.: Autots: Automatic time series forecasting model design based on two-stage pruning. *CoRR* **abs/2203.14169** (2022)
- [25] Wu, H., Xu, J., Wang, J., Long, M.: Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In: *NeurIPS*. pp. 22419–22430 (2021)
- [26] Zhang, X., Wu, H., Yang, J.: HyperTS: A Full-Pipeline Automated Time Series Analysis Toolkit. <https://github.com/DataCanvasIO/HyperTS> (2022), version 0.2.x
- [27] Zivot, E., Wang, J.: Vector autoregressive models for multivariate time series. *Modeling financial time series with S-PLUS®* pp. 385–429 (2006)