Taylor & Francis
Taylor & Francis Group

Applied Artificial Intelligence
An International Journal

ISSN: 0883-9514 (Print) 1087-6545 (Online) Journal homepage: www.tandfonline.com/journals/uaai20

Generalized Performance of LSTM in Time-Series Forecasting

Ryan Prater, Thomas Hanne & Rolf Dornberger

To cite this article: Ryan Prater, Thomas Hanne & Rolf Dornberger (2024) Generalized Performance of LSTM in Time-Series Forecasting, Applied Artificial Intelligence, 38:1, 2377510, DOI: 10.1080/08839514.2024.2377510

To link to this article: https://doi.org/10.1080/08839514.2024.2377510

© 2024 The Author(s). Published with license by Taylor & Francis Group, LLC.

View supplementary material

Published online: 15 Jul 2024.

Submit your article to this journal

Article views: 1568

View related articles

View Crossmark data

Citing articles: 1 View citing articles

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=uaai20

APPLIED ARTIFICIAL INTELLIGENCE
2024, VOL. 38, NO. 1, e2377510 (28 pages)
https://doi.org/10.1080/08839514.2024.2377510

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS Check for updates

Generalized Performance of LSTM in Time-Series Forecasting

Ryan Prater, Thomas Hanne, and Rolf Dornberger

Institute for Information Systems, University of Applied Sciences and Arts Northwestern Switzerland, Basel, Switzerland

ABSTRACT

Optimizing the time-series forecasting performance is a multi-objective problem which enables the comparison of general applicability of methods across multiple use cases such as finance and demographics. Libra, a time-series forecasting framework which shifts the problem of optimization from minimizing single to multiple evaluation measures and use cases, is used as a benchmark to evaluate the performance of the Long Short-Term Memory (LSTM) neural network. LSTMs with parameter tuning have been shown to perform well with time-series forecasting. This paper applies LSTMs (mostly with standard parameters and variations of some of them) to the Libra framework and concludes that due to data characteristic variance and without increased hardware and time constraints LSTMs do not outperform the median measures of Libra.

ARTICLE HISTORY
Received 16 May 2023
Revised 3 May 2024
Accepted 17 June 2024

Introduction

Until recently, there was no standardized benchmark for time-series forecasting (Bauer et al. 2020, 2021; Huang et al. 2019). This leads to poor quality evaluations of the generalized method accuracy and "fails to guide the choice of an appropriate method for a particular use case" (Bauer et al. 2021). A standardized benchmark would "accelerate the development of new algorithms and ML systems" (Huang et al. 2019) and allow comparison between methods and/or hyperparameters in order to increase the "quality of their prediction in general" across "many branches of applied sciences and business" (Peter and Matskevichus 2019).

Benchmarks exist for other branches of machine learning (ML) such as MLPerf (Huang et al. 2019) and the UCR dataset for time-series classification (Elsayed, Maida, and Bayoumi 2019). However, Bauer et al. (2020) show that 50% of their reviewed studies use "no more than three time series" for time-series forecasting evaluation.

CONTACT Thomas Hanne thomas.hanne@fhnw.ch Institute for Information Systems, University of Applied Sciences and Arts Northwestern Switzerland, Riggenbachstrasse 16, Olten, Basel 4600, Switzerland
Supplemental data for this article can be accessed online at https://doi.org/10.1080/08839514.2024.2377510

© 2024 The Author(s). Published with license by Taylor & Francis Group, LLC.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

Recently, Bauer et al. (2021) introduced Libra to fill this gap in benchmarking. The dataset is composed of four use cases – economics, finance, human access, and nature – each containing 100 heterogenous univariate time series from different domains, and three types of evaluation. This directly satisfies the benchmark suggestions of Huang et al. (2019): use cases, data properties, domain assortment, and output variability. Libra sets a standard with which new time-series forecasting methods may be generally evaluated.

The evaluation methods (EM) in Bauer et al. (2021) are ML but not deep learning methods, e.g., random forests. However, deep learning methods are commonly applied to time-series forecasting (Bauer et al. 2020; Elsayed, Maida, and Bayoumi 2019; Elsworth and Güttel 2020; Huang et al. 2019; Turkin 2017; Zaremba et al. 2014). Often, the Long Short-Term Memory (LSTM) method, introduced by Hochreiter and Schmidhuber (1997), demonstrates above-average performance (Elsworth and Güttel 2020; Huang et al. 2019). This paper applies the LSTM method to Libra to evaluate its generalized performance against the methods and results of Bauer et al. (2021). Thus, the contribution of this paper is twofold: We provide an experiment-based evaluation of LSTMs with standard parameters (with varying some of them, see Section 4) within the Libra as a standard framework for time-series forecasting (one-step-ahead forecast and *multi-step-ahead forecast)* considering data from different fields. Secondly, we provide the study as a paradigm for applying a rather wide range of performance measures related to time-series forecasting. Our article is structured as follows: In Section 2, related works are discussed with addressing contributions regarding time series, Libra, and LSTM models. Section 3 provides a description of the considered problem. The setup for our experiments is discussed in Section 4. Results are presented in Section 5. In Section 6 our conclusions are provided.

Related Work

Time Series

A time series is a sequence of data points distributed over time, usually in evenly spaced intervals. Time series can be *univariate* - one point of data per interval, or *multivariate* – multiple data observations per interval.

Time series contain several differentiating factors (Bauer et al. 2020, 2021; de Livera, Hyndman, and Snyder 2011), as with figurative *data* shown in Figure 1. *Trend* indicates the general direction of the series. *Seasonality* describes a regular pattern within series, e.g., *annual* Christmas shopping in sales data. *Frequency* is the length of one season. A *cycle* describes a pattern without a fixed frequency, e.g., stock market fluctuations. The *remainder* of the data outside of these factors is the *irregular component*.

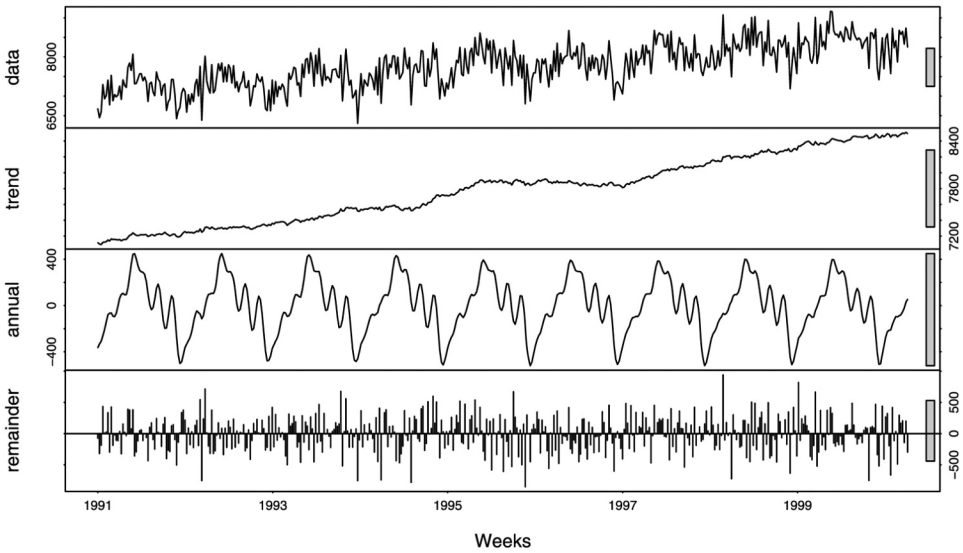

Figure 1. Decomposition of a time series into factors (de Livera, Hyndman, and Snyder 2011).

Libra

Libra provides four elements of standardization for time-series forecasting: three *datasets*, each of 100 time series of different domains, which forecast methods can evaluate on three *evaluation types* to compute 10 *performance measures* against benchmark *representative evaluation methods* (REM).

The source of time series varies, including publicly available datasets and the *M-Competitions* (Makridakis, Spiliotis, and Assimakopoulos 2018), which are commonly used for time-series forecasting, but display heterogeneity in their time-series features (Bauer et al. 2021). A further comparison of the datasets can be found in Bauer et al. (2021).

Bauer et al. (2021) shows that the frequency and length of the Libra datasets are more diverse and extensive than other time-series forecasting datasets. Across the Libra datasets, the length of the time series varies from 20 to 372,864 data points, and the frequency varies from 1 to 4,368. The median lengths of the Economics, Finance, Human, and Nature use cases are 170, 682, 1464, and 714 data points. This means that Libra has a large variation of horizon lengths *hz* (also known as the prediction interval) in time series across use cases, with the smallest, $hz = 1$ and the largest, $hz = 74{,}572$ (see Figure 2).

Libra provides three types of evaluation for methods to execute, which vary on the ratio of *training* and *test* data, where training data is the data used for method learning and test data is data to which the method is not exposed during training and which is used for error measurement (Bauer et al. 2021). The training data is sent to the EM as a time series or history to be evaluated (*hist*). The length of the

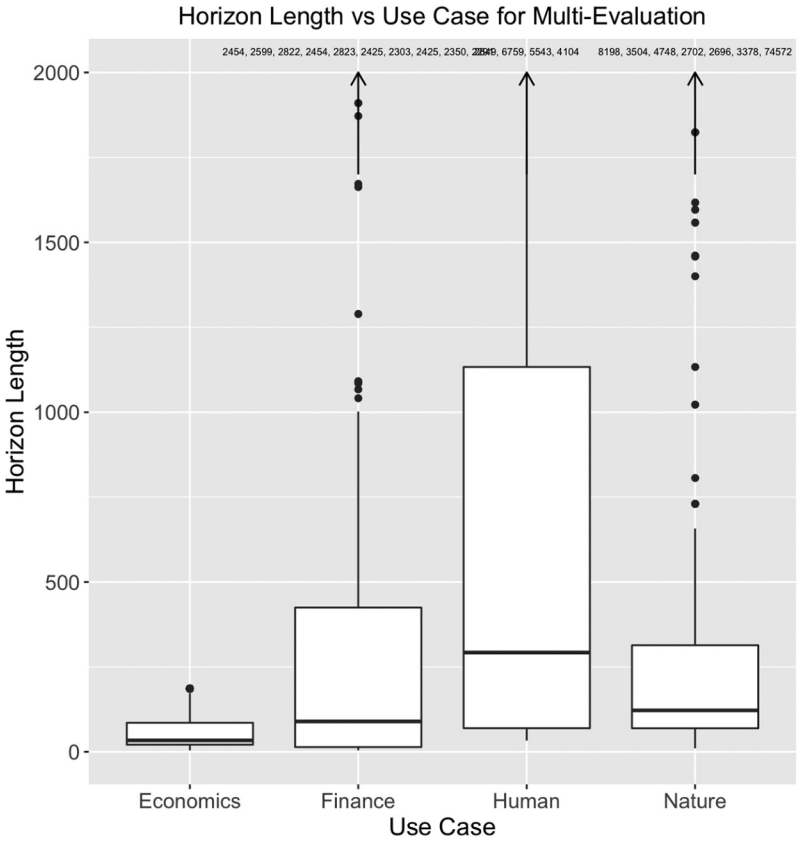

Figure 2. Multi-step-ahead forecast (MSAF) horizon (prediction) lengths by use case.

prediction (prediction interval) for the EM to make is the *horizon hz* (Bauer et al. 2021; de Livera, Hyndman, and Snyder 2011). The length of *hist* to be used to predict *hz* is known as the *lag* and varies according to EM implementation.

The first evaluation type Libra offers is the *one-step-ahead forecast* (OSAF) which provides *hz = 1* and all values of the time series excluding the last as *hist*. The *multi-step-ahead forecast* (MSAF) provides the first 80% of the time series as *hist* and the length of the last 20% as *hz*. The rolling-origin forecast (ROF) divides the time series into 100 equal indexes, is conscious of start/end indices, and performs a MSAF that starts with each of these indices.

Multiple performance measures exist for time-series forecasting and many papers use different measures to evaluate EM success. Bauer et al. (2021) optimize on error and time-to-result. Peter and Matskevichus (2019) assesses specific complexity, i.e., "the ratio of the number of model parameters . . . to its median error." Huang et al. (2019) suggest a broader difference between system performance (including error and time-to-result) and workload characterization.

Within error and time measurement, many measures exist to compare performance (de Livera, Hyndman, and Snyder 2011; Elsworth and Güttel 2020; Peter and Matskevichus 2019). To better measure the generalized performance and prevent overfitting, Libra measures EM performance across 10 measures to "counter the weakness of a specific error measure" (Bauer et al. 2021). Some error measures make use of the baseline function, sNaïve, which simply predicts the same value(s) from the previous observation period. Each measure is listed below where hz is the forecast horizon, x_t the actual value at time t, y_t the forecast value at time t, p the length of the period, q the length of the history, and $hist_i$ the historical values at time i.

- *Symmetrical mean absolute percentage error* (sMAPE) computes a percentage error between two numeric vectors, is independent of scale, and has an upper limit of 200%, but has a "heavier penalty when forecasts are high" and may produce the upper error limit when actual or prediction values are close to zero (Hyndman and Koehler 2006):

$$\bar{e}_{sM} := \frac{200\%}{hz} \sum_{t=1}^{hz} \frac{|x_t - y_t|}{|x_t + y_t|}. \tag{1}$$

- *Standard deviation of sMAPE*, $\sigma_{e_{sM}}$.
- *Mean absolute scaled error* (MASE) computes the forecast error "scaled by the in-sample mean absolute error obtained using the naïve forecasting method," is less sensitive to outliers, but produces a less immediately intuitive result (Hyndman and Koehler 2006):

$$\bar{e}_{MA} := \frac{\frac{1}{hz}\sum_{t=1}^{hz}|x_t - y_t|}{\frac{1}{q-p}\sum_{i=p+1}^{q}\left|hist_i - hist_{i-p}\right|}. \tag{2}$$

- *Standard deviation of MASE*, $\sigma_{e_{MA}}$.
- *Under mean wrong estimation* is the relative number of forecast values that underestimate the actual values:

$$\rho_U := \frac{1}{hz} \cdot \sum_{t=1}^{hz} \max(\sin(x_t - y_t), 0). \tag{3}$$

- *Over mean wrong estimation* is the relative number of forecast values that overestimate the actual values:

$$\rho_O := \frac{1}{hz} \cdot \sum_{t=1}^{hz} \max(\sin(y_t - x_t), 0). \tag{4}$$

- *Under mean wrong accuracy share* is the mean percentage error when underestimating the actual values:

$$\delta_U := \begin{cases} \frac{1}{hz \cdot \rho_U} \cdot \sum_{t=1}^{hz} \frac{\max(x_t - y_t, 0)}{|x_t|}, & \rho_U > 0, \\ 0 & otherwise, \end{cases} . \tag{5}$$

- *Over mean wrong accuracy share* is the mean percentage error when overestimating the actual values:

$$\delta_O := \begin{cases} \frac{1}{hz \cdot \rho_O} \cdot \sum_{t=1}^{hz} \frac{\max(y_t - x_t, 0)}{|x_t|}, & \rho_O > 0, \\ 0 & otherwise, \end{cases} . \tag{6}$$

- *Time-to-result*, t_{sN}, reflects the average time "normalized by sNäive" in order to be independent of the underlying hardware, and includes only the time "in which the forecasting method receives the time series, estimates the parameters, creates the model, and performs the forecast" (Bauer et al. 2021).
- *Standard deviation of time-to-result*, $\sigma_{t_{sN}}$.

Libra additionally provides pre-computed results for 10 REMs for EM comparison (Bauer et al. 2021):

- *ETS* (Error, Trend, Seasonal) decomposes time series into factors and determines factor relationships, is "good for time series with a strong trend," but "is rather bad in detecting long and complex seasonal patterns" (Bauer et al. 2020).
- *sARIMA* (Auto-Regressive Integrated Moving Average) is a seasonal variant of the ARIMA method. ARIMA is a traditionally accurate and popular EM, although its method selection can be time consuming (Bauer et al. 2020; Peter and Matskevichus 2019; Siami-Namini, Tavakoli and Siami Namin 2019).
- sNaïve
- *TBATS* is an extension of ETS which further improves seasonality and error modeling but requires positive values (Bauer et al. 2020).
- *Theta* de-seasonalizes and splits the data into short- and long-term components to form a combined weighted forecast but "struggles with long or multiple seasonal patterns" (Bauer et al. 2020).

This list includes the following ML REMs:

- *GPyTorch* applies Gaussian inference – a probability mapping over a set of functions.
- *NNetar* "is a feed-forward neural network with one hidden layer" (Bauer et al. 2021).

- *SVR* (Support Vector Regression Machine) attempts to separate data using mathematical methods but is sensitive to parameterization and training can be computationally expensive (Bauer et al. 2020).
- *RF* (Random Forest) is a set of decision trees, each trained on a random sample of the features.
- XGBoost is also a set of decision trees, except that each is grown "sequentially with knowledge from the preceding tree" (Bauer et al. 2021). It "requires many hyper-parameter settings and is sensitive to overfitting if the data are noisy" (Bauer et al. 2020).

For each evaluation of an EM on a given use case for a given evaluation type (*use case evaluation*), Libra outputs a comma-separated-values file of a (pre-computed) matrix of performance measure – REM result values. In Bauer et al. (2021) full results are included only for MSAF, and the combined results of MSAF across all use cases are provided below for basic reference.

The Long Short-Term Memory Method

A neural network is a processing network which contains at least one input, hidden, and output layer where the input and output layers are connected to the hidden layer. The hidden layer is composed of nodes (also known as units) which learn and hold processing weights as information passes through. In its basic form, a neural network is *feed-forward*, meaning information passes from input to output. For continuous learning networks, backpropagation can be programmed, enabling the error of the output (the *loss function*) to be fed back into the hidden layers, resulting in a learning phenomenon where the network becomes more accurate over time (Elsworth and Güttel 2020; Hochreiter and Schmidhuber 1997; Siami-Namini, Tavakoli, and Siami Namin 2019).

Recurrent Neural Networks (RNNs) are back propagated neural networks which are designed for sequences. Their input consists of the next weighted element of the sequence, combined with the output from the previous sequence RNN output. The benefit is that the method is "independent of sequence length" (Elsworth and Güttel 2020) and learning is influenced by previous sequence elements, although this memory via weights is short term and learned patterns are easily forgotten by new input (Connor and Atlas 1991; Rumelhart et al. 1986; Siami-Namini, Tavakoli, and Siami Namin 2019). This is known as the vanishing gradient problem (Elsworth and Güttel 2020).

To counter the vanishing gradient, Hochreiter proposed the LSTM, composed of an input gate, memory cell, and output gate, a similar structure as for the RNN (Hochreiter and Schmidhuber 1997; also cf.; Elsworth and Güttel 2020). However, the LSTM retains information for a longer time by leveraging three internal gates which use the sigmoid function to make decisions on

values which approach 0 or 1. The forget gate (*f_g*) determines whether to keep or forget the cell state, the input gate (*i_g*) determines whether to accept new data into the cell state, and the output gate (*o_g*) determines which cell states should leave the cell and feed into the next hidden state (Elsworth and Güttel 2020; Siami-Namini, Tavakoli, and Siami Namin 2019). The input gate also contains a cell update layer (*c_u*), also known as the candidate gate which transforms the input into a "vector of new candidate values" (Siami-Namini, Tavakoli, and Siami Namin 2019).

The input-output of the LSTM can be described as:

$$\left(h^{(t)}, c^{(t)}\right) = \mathcal{L}\left(h^{(t-1)}, c^{(t-1)}, x^{(t)}\right), \tag{7}$$

where $h^{(t)}$is the hidden state and $c^{(t)}$ the cell state at time t with $h^{(t)}, c^{(t)} \in [-1, 1]$, $x^{(t)}$ the element of the sequence $\mathbb{R}^d$ (Elsworth and Güttel 2020).

LSTMs have been shown to perform well with long lag times and generalize well (Hochreiter and Schmidhuber 1997), although they require "very large amounts of training data and rather long training time" (Bauer et al. 2020). To provide greater lag in cases of a small horizon (such as the OSAF), a difference in input/output size may be required. The Encoder-Decoder architecture of the RNN (see Figure 3) can be used to map "a variable-length source sequence to a fixed-length vector" and further into "a variable-length target sequence" (Cho et al. 2014). Such an architecture (see Figure 3) is often used for translations (len_e words of English to len_f words of French), or in the case of time series, len_{la} length of lag to len_{hz} of horizon).

During recent years, various studies investigated LSTM models for time-series prediction in different scenarios. Karasu and Altan (2022)

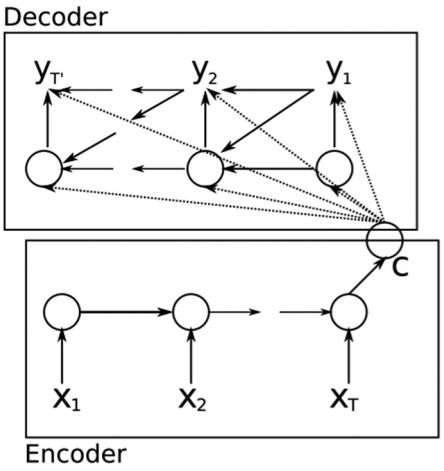

Figure 3. An encoder-decoder architecture which maps the output of a method (the encoder LSTM) to a vector which is used for the input of the decoder LSTM.

suggest an LSTM model with hyperparameter optimization by two metaheuristic methods for predicting volatile oil prices. Abbasimehr and Paki (2022) apply LSTM and a multi-head attention model to time-series prediction, which are compared with some standard time-series forecasting techniques and hybrid approaches. A comparison of LSTM and CNN models for predicting financial time series is provided by Mehtab and Sen (2022). Studies on multivariate time-series prediction using LSTMs are provided by Guo et al. (2020) and Ghanbari and Borna (2021). A recent survey of applications of LSTM to time-series prediction is given by Lindemann et al. (2021). Although comparative results are provided in various of these studies, they do not make use of a rather comprehensive library of test data, such as Libra which allows to evaluate and benchmark the results under various performance measures as discussed above.

Problem Description

The problem considered in Bauer et al. (2021), Bauer et al. (2020), Peter and Matskevichus (2019) and Elsayed, Maida, and Bayoumi (2019) is the multi-objective "selection of the most appropriate forecasting method for a given use case" (Bauer et al. 2021) with a focus on minimization of method error across a dataset constrained to univariate time series. Bauer et al. (2021) conclude that "no method performs best for all use cases." Peter and Matskevichus (2019) show that the application of certain method architectures is more common among certain domains, such as finance. Elsayed, Maida, and Bayoumi (2019) demonstrate how the variability of deep learning parameters affects "the optimal neural network architecture for forecasting time series."

Time-Series Forecasting

In time-series forecasting, an EM is given a time series (ts) of length n and horizon to predict the values $ts_{n+1\ldots n+hz}$ denoted $p_{0\ldots hz-1}$. These values are used as inputs for a performance measure, which typically measures the difference to a baseline method or the actual future values, denoted $a_{0\ldots hz-1}$, producing the error, $e_{0\ldots hz-1}$.

In the context of this study, the horizon depends on the evaluation type (OSAF, MSAF, ROF). Given the time and complexity requirements, this study *only focuses on the OSAF and MSAF*. This means, values of hz will be one- or two-dimensional and avoid the third dimension of the ROF.

The time series and horizon are both programmatically provided by Libra to the EM, as well as the 10 performance measures. For each use case evaluation, the EM generates $p_{0\ldots hz-1}$ which Libra uses as input to output a value for each performance measure.

The difficulty in optimization with many performance measures is choosing which measure to optimize on. Because the function of optimization depends on the performance measure (e.g., minimizing or maximizing a value), for simplicity an optimized outcome will now be referred to as *well performing*. An EM can perform well in any number of performance measures (Elsworth and Güttel 2020).

To draw meaningful conclusions about the generalized performance of an EM and prevent numerous comparisons across the matrix of measures, certain heuristics can be evaluated. In this study, the focus is on *maximizing the sum of best performance measures* (*wins*), as well as *comparing the performance measure of the EM to the mean and median across all REMs* (*above or below average performance*).

For example, in the LIBRA results for MSAF on all use cases reported in Bauer et al. (2021), XGBoost exhibits five *wins*, and performs well with a *below average* $\bar{e}_{MA}$ (1.00) compared to the *mean* (1.41), and a *below average* $\bar{e}_{MA}$ (1.00) compared to the *median* (1.12). This aligns with the conclusion of Bauer et al. (2021) that XGBoost is "the most accurate" and "for two use cases, the best" machine learning method.

LSTM Parameterization

The complexity of the LSTM method provides many parameters, such as weight bias, activation functions, the number of hidden units, and stacked layers (ensembling; see below). It is common to use ML frameworks which provide *out of the box* solutions with reasonably preset parameter values (Bauer et al. 2021). This prevents over-parameterization, and as demonstrated by Peter and Matskevichus (2019), "an increase in the number of parameters does not greatly affect the accuracy of the models." In the next section, these parameters are described along with their values for the LSTM evaluation.

Computation Framework and Setup for Experiments

The Libra framework is written in the R programming language and consists of three core files: *benchmark.R*, used as a test harness or *runner* to be called from the implemented EM, *forecasting.R* which defines the evaluation types and iterates through the use case datasets, and *measures.R* which defines the performance measures for standardized implementation. The EM is to be written in a separate file, such as *lstm.R*, and call the runner. The implementation decisions of the EM and the frameworks used are left to the researcher.

In this study, the LSTM method was implemented using the *Keras* framework (Keras 2021). Keras is an API layer on top of the popular TensorFlow core, a provider of ML methods and accessory functions. Keras includes an LSTM layer which was used. However, a few

parameters were modified in this study to 1) build a method which generalizes over the datasets, 2) provide a means of optimization testing, and 3) limit the constraints on hardware and time. Apart from the parameters for which the Libra is responsible (such as variation on the horizon), parameters inside of the EM corresponding to this study are listed below.

• *Data normalization*: Because of the trigonometric functionality of LSTMs, data can be normalized in different manners to standardize the scale of weights. Z-Score (standard score) was used to represent input values as the number of standard deviations away from the mean. The data were scaled with

$$Z := \frac{x - \mu}{\sigma}, \tag{8}$$

and descaled with

$$Z' := x \cdot \sigma + \mu, \tag{9}$$

where x is the raw value, μ is the data mean, and σ is the data standard deviation.

- *Lag*: computed as 10% of the time-series length to adapt to assorted lengths. Setting lag equal to the horizon was considered but did not perform well in preliminary tests when $hz = 1$.
- *Architecture*: LSTMs can be ensembled (combined) for increased accuracy (Bauer et al. 2020; Elsayed, Maida, and Bayoumi 2019). Using the Encoder-Decoder architecture, a preliminary test led the design such that the Encoder consists of two stacked LSTMs and the Decoder a single LSTM followed by a single node to combine the output into one value.
- *Number of samples*: The number of slices of lag from each time series which was used for learning. Generally, LSTMs perform better with more training data (Bauer et al. 2020). As mentioned in the results section, tests were conducted with 100 to 1024 samples.
- *Number of hidden layer units*: The input is univariate, and therefore a direct relation between number of internal units and features cannot be easily determined. Feature extraction prior to training was not used. As discussed in the results section, experiments were conducted with 16 to 1024 Encoder hidden units and 8 to 512 Decoder hidden units.
- *Batch size*: The number of samples processed in a single iteration of back propagation before the weights are updated. A higher number decreases training time but can lead to less performant methods. This value was set to 32 as a balance between performance and hardware constraints.
- *Epochs*: The number of passes over the training dataset. As mentioned in the results section, tests were conducted with 16 to 200 epochs.

- *Kernel regularizer*: Applies a regression analysis on the weights of the LSTM. This was set to 10% lasso regression.
- *Bias regularizer*: Applies a regression analysis on the biases of the LSTM. This was set to 10% lasso regression.
- *Kernel initializer*: The initial weights for the LSTM. This was set to *orthogonal* following the study of Hu, Xiao, and Pennington (2020).
- *Loss function*: This was set as *mean squared error* to add penalty to estimations further from the mean which is formalized as MSE:

$$\text{MSE} := \frac{1}{hz}\sum_{i=0}^{hz}(a_i - x_i)^2, \tag{10}$$

- *Optimizer learning rate*: The gradient descent algorithm and rate to update network weights. Adam – the default optimizer – was used with a learning rate of 0.001.
- *Loss reduction*: The Keras training loop provides a callback which can reduce the learning rate if a plateau is detected. This is set to a reduction of 0.01 if the learning rate has not changed in two epochs.
- *Validation split*: The percent of data to withhold from the training data to provide an out-of-sample dataset for monitoring the unbiased accuracy of the method. The size of this test data set was set to 10%. For training purposes, 90% of the data was used.
- *Sample shuffle*: Shuffles the samples to promote diversified sample learning. This was set to True.
- *Early stopping*: A Keras callback which stops training earlier than the number of epochs if no change in loss is detected. This was set to 10 epochs.

Although this is not essential for the analysis of the computational effort (as we use the hardware-independent normalized time-to-result measure suggested by Bauer et al. (2021) as discussed in Section 2.2) we would like to mention, that all experiments have been conducted on a computer with a 2.3 GHz Quad Core Intel Core i7 CPU (with Turbo Boost up to 4.1 GHz), an Onboard GPU (Intel Iris Plus Graphics) (not used for the experiments), 32 GB 3733 MHz LPDDR4X RAM, and a 512 GB SSD.

Results

OSAF was first tested by training a method of 1000 samples, 64 Encoder hidden units, 32 Decoder hidden units, and 100 epochs for each time series. Each of the four use cases were tested and their results are given in Tables A1–A5 in Appendix A. A summary overview across use cases via mean average was made; the results are listed in Table 1. A cell entry in italics indicates above average (worse) performance, and a bold value, below average (better).

When predicting one step ahead, the LSTM had zero *wins*, i.e., did not perform well (Min Measure column) for any of the measures. When comparing error measures ($\bar{e}_{sM}$ and $\bar{e}_{MA}$), the LSTM was above the median value for most measures but below the mean value for $\bar{e}_{sM}$, its standard deviation, and $\sigma_{e_{MA}}$. For over- and underestimation, the LSTM tended to underpredict, although accuracy was comparable to the accuracy of its overpredictions. The time-to-result (t_{sN}) of the LSTM was below the mean, but over the median.

The MSAF was tested with 100 samples, 16 Encoder hidden units, 8 Decoder hidden units, and 16 epochs. The reduced values compared to the OSAF case are chosen to have a similar computational effort. Again, each of the four use cases with detailed statistics are given in Tables A6–A10 in Appendix B. These parameters were intentionally set low due to hardware constraints and some of the maximum time-series lengths (the size of the Decoder LSTM is equal to the horizon). The mean average results for MSAF across each use case are shown in Table 2.

With MSAF, the LSTM achieved a minimum share of overpredictions, but this is complemented by the maximum share of underpredictions, ρ_U. Again, the LSTM was under-mean and over-median for $\bar{e}_{sM}$, $\sigma_{e_{sM}}$, t_{sN}, and $\sigma_{t_{sN}}$, similar to the OSAF. These statements together imply that the LSTM on the MSAF tends to consistently predict the mean trend, with a bias toward underprediction.

Table 1. OSAF abbreviated results of combined use cases.

Measures	LSTM	Min Measure	Max Measure	Mean	Median
$\bar{e}_{sM}$	5.03E + 03	Theta	sARIMA	**4.24E + 04**	*2.48E + 02*
$\sigma_{e_{sM}}$	3.86E + 03	sNaive	sARIMA	**3.00E + 05**	*4.13E + 02*
$\bar{e}_{MA}$	9.72E + 00	NNetar	GPyTorch	*8.18E + 00*	*8.06E + 00*
$\sigma_{e_{MA}}$	1.28E + 01	NNetar	Theta	**1.44E + 01**	**1.29E + 01**
ρ_U	4.09E − 01	TBATS	sNaive	*3.64E − 01*	*3.83E − 01*
ρ_O	4.33E − 01	GPyTorch	sARIMA	**4.88E − 01**	**4.73E − 01**
δ_U	2.23E − 01	Theta	XGBoost	*2.04E − 01*	*2.12E − 01*
δ_O	2.05E + 00	sNaive	LSTM	*5.11E − 01*	*2.99E − 01*
t_{sN}	1.00E + 04	Theta	sARIMA	**4.77E + 05**	*4.45E + 02*
$\sigma_{t_{sN}}$	8.55E + 03	Theta	sARIMA	**2.49E + 06**	*2.07E + 03*

Table 2. MSAF abbreviated results of combined use cases.

Measures	LSTM	Min Measure	Max Measure	Mean	Median
$\bar{e}_{sM}$	4.13E + 01	sARIMA	NNetar	**5.95E + 01**	*2.57E + 01*
$\sigma_{e_{sM}}$	7.47E + 01	sNaive	NNetar	**3.69E + 02**	*5.14E + 01*
$\bar{e}_{MA}$	2.26E + 00	sARIMA	LSTM	*1.28E + 00*	*1.12E + 00*
$\sigma_{e_{MA}}$	6.06E + 00	sARIMA	SVR	*3.35E + 00*	*2.43E + 00*
ρ_U	6.32E − 01	sARIMA	LSTM	*5.79E − 01*	*5.91E − 01*
ρ_O	3.68E − 01	LSTM	sARIMA	**4.20E − 01**	**4.09E − 01**
δ_U	2.39E − 01	Theta	ETS	*2.37E − 01*	*1.71E − 01*
δ_O	1.03E + 01	TBATS	ETS	*9.23E + 00*	*7.48E + 00*
t_{sN}	1.13E + 04	sNaive	sARIMA	**7.99E + 04**	*6.19E + 02*
$\sigma_{t_{sN}}$	1.11E + 04	sNaive	sARIMA	**5.10E + 05**	*1.76E + 03*

To understand how different parameters affected the performance measures, further tests were made applying the LSTM to MSAF on the Economics dataset, due to its lowest mean length, with the changes in parameters listed below.

- *Test 1: Testing more robust parameters*. Entire economic dataset (median length = 170), 1024 samples, 256 Encoder hidden units, 128 Decoder hidden units, 200 epochs.
- *Test 2: Testing a smaller mean length*. Series 1-10 of the Economic dataset (median length = 20), 1024 samples, 256 Encoder hidden units, 128 Decoder hidden units, 200 epochs. This test was further developed due to its faster execution time.
- *Test 3: Testing equal Encoder/Decoder units*. Series 1-10 of the Economic dataset, 1024 samples, 256 Encoder hidden units, 256 Decoder hidden units, 200 epochs.
- *Test 4: Testing more Encoder/Decoder units*. Series 1-10 of the Economic dataset, 1024 samples, 1024 Encoder hidden units, 512 Decoder hidden units, 200 epochs.
- *Test 5: Testing fewer Encoder/Decoder units*. Series 1-10 of the Economic dataset, 1024 samples, 64 Encoder hidden units, 32 Decoder hidden units, 200 epochs.
- *Test 6: Testing a longer lag*. Series 1-10 of the Economic dataset, 1024 samples, 256 Encoder hidden units, 128 Decoder hidden units, 200 epochs, and a lag of 60% instead of 10% of *hist* length.

The results of the six tests (see Table 3) indicates that adding more hidden units (Test 4) has a negative effect across most measures, which could be due to the small median lengths of the time series (too many units are "underweighted"). In comparison to Test 1, Tests 2, 3, 5, and 6 improve on $\bar{e}_{MA}$ and t_{sN} but perform worse on $\bar{e}_{sM}$. This inconsistency could be due to the fact that these tests use only 10% of the dataset of Test 1, which is consistent with the warning of Bauer et al. (2020) that LSTMs require a lot of training data. Their lack of minimization across other measures could indicate that the test series

Table 3. LSTM MSAF parameter testing.

Measures	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
$\bar{e}_{sM}$	2.55E + 01	4.93E + 01	4.43E + 01	5.99E + 01	5.05E + 01	3.84E + 01
$\sigma_{e_{sM}}$	2.39E + 01	4.07E + 01	3.57E + 01	8.01E + 01	3.90E + 01	2.20E + 01
$\bar{e}_{MA}$	1.39E + 00	9.97E − 01	1.01E + 00	1.81E + 00	1.11E + 00	8.37E − 01
$\sigma_{e_{MA}}$	3.33E + 00	9.52E − 01	9.88E − 01	3.13E + 00	9.30E − 01	5.50E − 01
ρ_U	6.76E − 01	6.13E − 01	4.75E − 01	4.50E − 01	5.63E − 01	6.50E − 01
ρ_O	3.24E − 01	3.88E − 01	5.25E − 01	5.50E − 01	4.38E − 01	3.50E − 01
δ_U	1.56E − 01	2.68E − 01	2.65E − 01	3.95E − 01	2.48E − 01	2.22E − 01
δ_O	4.39E + 00	2.99E − 01	5.25E − 01	8.35E − 01	3.86E − 01	2.89E − 01
t_{sN}	1.13E + 05	1.98E + 04	2.47E + 04	1.23E + 05	1.14E + 04	4.64E + 04
$\sigma_{t_{sN}}$	9.65E + 04	7.23E + 03	9.56E + 03	4.79E + 04	4.63E + 03	5.84E + 03

had too few data. Test 6 shows an error measure improvement at the sacrifice of t_{sN} and ρ_U. Given the correct hardware, future research could test the longer lag of Test 6 with the full use case dataset of Test 1.

Conclusions and Outlook

In relation to the REM values provided by Libra, the LSTM performed under-mean and over-median for $\bar{e}_{sM}$ and over-mean and over-median for $\bar{e}_{MA}$, indicating that the LSTM learned to predict the mean value but not the distinguished patterns (frequency or seasonality) of the training data. Indeed, Elsworth and Güttel (2020) state that high frequency, low data time series can "appear as noise" to LSTMs, and finding a suitable lag size is "tricky" because as lag increases, "the size of the training set decreases."

A difficulty of this study was the large variance of time series found in the Libra dataset. Although this is a benefit for diversified testing, many studies mention "tuning" LSTM parameters for the given training data (Bauer et al. 2020; Elsworth and Güttel 2020; Hochreiter and Schmidhuber 1997; Siami-Namini, Tavakoli and Siami Namin 2018), which is difficult for time series of such varied nature. Additionally, some time series required a large horizon (predicting 74,572 values at once) which increased the size of the method structures, and therefore significantly impacts the training time. Parameter tuning of an LSTM to achieve optimal generalized performance across the full Libra dataset would require a significant amount of hardware potential and time. This challenges Libra's purpose as a "level playing field" (Bauer et al. 2021) as the complexity could be prohibitive for new experimental methods.

This study can serve as a basis for LSTM generalization and the development of the Libra framework. To begin with, the method parameters provided were based on previous studies, but so far, the studies did not explicitly indicate optimal values. Further experimentation of the parameter values such as batch sizes discussed in Section 4 could indicate how certain parameters influence the performance measures of Libra and what most preferable settings might be. It should be noted, however, that such a tuning of parameters for a domain overarching set of time series (as provided in Libra) has its limitations regarding prediction quality as already indicated by our current findings. Therefore, the way better results by parameter fine tuning should be focusing on more homogenous time-series data.

Another interesting topic for future research could be the measurement of the effect of graphics processing units on t_{sN}. Also, gated-recurrent neural network which have a similar structure to the LSTM, but lower memory requirements (Elsayed, Maida, and Bayoumi 2019) could be evaluated for performance differences. The trends and seasonality of the time series was not removed before training, but is

a recommendation by Elsworth and Güttel (2020). A key design choice in this study was to train a new LSTM based on *each* incoming time series. A future study could investigate first training the LSTM on *all* time series before making the first prediction, although this would require an architectural change of Libra. Likewise, instead of predicting all values at once, the LSTM could be changed for MSAF to always predict one value, and repeat *hz* times, which might decrease complexity. Finally, offering Libra in other programming languages, such as Python, could indicate how frameworks such as Keras or methods such as the LSTM differ in performance, as well as broaden the exposure of the benchmarking framework.

Libra fills a gap in time-series forecast benchmarking by providing a diverse dataset of time series of wide characteristic variance. Studies of LSTMs have shown success in forecasting, albeit with predictions on single or few time series. Applying the LSTM method with common parameters to the Libra dataset results in a prediction of the mean but results which do not perform better than existing representative methods, such as sARIMA or XGBoost. For improved performance of the LSTM on Libra, the complexity of hardware and/or time would need to increase, or a subset of the data should be evaluated.

Increasing hardware power and the availability of high-performance computing clusters may allow for LSTM neural networks (or other architectures) to become available with increased complexity and aspects of computation time may become less relevant. As a consequence, our suggested set of measures might be modified, e.g., by omitting time-related measures such as time-to-result. It could also be useful to consider weights for specifying user-specific preferences. For instance, prediction accuracy could be weighted higher than other measures if this aspect is the main concern. This could be considered in an approach from the field of multiple criteria decision-making (Hanne 2012).

Another suggestion, for future research, would be to analyze in detail the factors that prevent LSTMs from doing better. This could be done by investigating the considered dataset regarding properties that either let LSTMs struggle or perform well.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

ORCID

Ryan Prater http://orcid.org/0000-0003-3698-0111
Thomas Hanne http://orcid.org/0000-0002-5636-1660

Statements and Declarations

The authors have no relevant financial or non-financial interests to disclose. No funding was received to assist with the preparation of this manuscript. All data used in this study is included in the Libra database available at Zenodo: http://doi.org/10.5281/zenodo.4399959.

References

Abbasimehr, H., and R. Paki. 2022. Improving time series forecasting using LSTM and attention models. *Journal of Ambient Intelligence and Humanized Computing* 13 (1):673–91. doi: 10.1007/s12652-020-02761-x.

Bauer, A., M. Züfle, S. Eismann, J. Grohmann, N. Herbst, and S. Kounev. 2021. Libra: A benchmark for time series forecasting methods. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, 189–200, ACM. doi: 10.1145/3427921.3450241.

Bauer, A., M. Zufle, N. Herbst, A. Zehe, A. Hotho, and S. Kounev. 2020. Time series forecasting for self-aware systems. *Proceedings of the IEEE* 108 (7):1068–93. doi: 10.1109/JPROC.2020.2983857.

Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 1724–34. doi: 10.3115/v1/d14-1179.

Connor, J., and L. Atlas. 1991. Recurrent neural networks and time series prediction. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. 1, 301–06, IEEE. doi: 10.1109/IJCNN.1991.155194.

de Livera, A. M., R. J. Hyndman, and R. D. Snyder. 2011. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association* 106 (496):1513–27. doi: 10.1198/jasa.2011.tm09771.

Elsayed, N., A. S. Maida, and M. Bayoumi. 2019. Gated recurrent neural networks empirical utilization for time series classification. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 1207–10, IEEE. doi: 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00202.

Elsworth, S., and S. Güttel. 2020. Time series forecasting using LSTM networks: A symbolic approach. *arXiv preprint arXiv:2003.05672*. 1–12. doi: 10.48550/arXiv.2003.05672.

Ghanbari, R., and K. Borna. 2021. Multivariate time-series prediction using LSTM neural networks. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*, Tehran, Iran, March 3–4, 2021, 1–5. IEEE.

Guo, Z., P. Liu, J. Yang, and Y. Hu. 2020. Multivariate time series classification based on mcnn-lstms network. In *Proceedings of the 2020 12th International Conference on Machine Learning and Computing*, Shenzhen China, February, 510–17.

Hanne, T. 2012. *Intelligent strategies for meta multiple criteria decision making*, Reprint ed. New York: Springer.

Hochreiter, S., and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9 (8):1735–80. doi: 10.1162/neco.1997.9.8.1735.

Hu, W., L. Xiao, and J. Pennington. 2020. Provable benefit of orthogonal initialization in optimizing deep linear networks. *arXiv preprint* arXiv:2001.05992 1–16. doi: 10.48550/arXiv.2001.05992.

Huang, X., G. C. Fox, S. Serebryakov, A. Mohan, P. Morkisz, and D. Dutta. 2019. Benchmarking deep learning for time series: Challenges and directions. *2019 IEEE International Conference on Big Data (Big Data)*, 5679–82, IEEE. doi: 10.1109/BigData47090.2019.9005496.

Hyndman, R. J., and A. B. Koehler. 2006. Another look at measures of forecast accuracy. *International Journal of Forecasting* 22 (4):679–88. doi: 10.1016/j.ijforecast.2006.03.001.

Karasu, S., and A. Altan. 2022. Crude oil time series prediction model based on LSTM network with chaotic Henry gas solubility optimization. *Energy* 242:122964. doi: 10.1016/j.energy.2021.122964.

Keras. 2021. Keras - LSTM layer. https://keras.io/api/layers/recurrent_layers/lstm/.

Lindemann, B., T. Müller, H. Vietz, N. Jazdi, and M. Weyrich. 2021. A survey on long short-term memory networks for time series prediction. *Procedia CIRP* 99:650–55. doi: 10.1016/j.procir.2021.03.088.

Makridakis, S., E. Spiliotis, and V. Assimakopoulos. 2018. The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting* 34 (4):802–08. doi: 10.1016/j.ijforecast.2018.06.001.

Mehtab, S., and J. Sen. 2022. Analysis and forecasting of financial time series using CNN and LSTM-based deep learning models. In *Advances in Distributed Computing and Machine Learning: Proceedings of ICADCML 2021*, Bhubaneswar, India, 405–23. Springer Singapore.

Peter, G., and M. Matskevichus. 2019. Hyperparameters tuning for machine learning models for time series forecasting. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 328–32, IEEE. doi: 10.1109/SNAMS.2019.8931860.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition*, ed. D. E. Rumelhart and J. L. McClelland, vol. 1, 318–62. Cambridge, MA: MIT Press [Online]. https://apps.dtic.mil/docs/citations/ADA164453.

Siami-Namini, S., N. Tavakoli, and A. Siami Namin. 2019. A comparison of ARIMA and LSTM in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1394–401, IEEE. doi: 10.1109/ICMLA.2018.00227.

Turkin, A. 2017. Tikhonov regularization for long short-term memory networks. *arXiv preprint*. arXiv:1708.02979. doi: 10.48550/arXiv.1708.02979.

Zaremba, W., I. Sutskever, O. Vinyals, and N. M. Rouphail. 2014. Recurrent neural network regularization. *Transportation Research Record*. 2013 (2389):1–11. [Online] http://arxiv.org/abs/1409.2329.

Appendices

Appendix A One Step Ahead Forecast Results

Table A1. Results of the OSAF combined use cases via mean averaging.

2.98E–01	LSTM	ETS	sARIMA	sNaive	TBATS	Theta	GPyTorch	NNetar	Random Forest	SVR	XGBoost	Min Measure	Min Value	Max Measure	Max Value	Mean	Median
Avg. Symmetrical Mean Absolute Percentage Error	5.03E+03	2.22E+02	4.54E+05	1.23E+01	3.39E+03	1.14E+01	1.65E+03	2.04E+02	7.28E+02	2.48E+02	2.85E+01	Theta	1.14E+01	sARIMA	4.54E+05	4.24E+04	2.48E+02
SD. Symmetrical Mean Absolute Percentage Error	3.86E+03	3.77E+02	3.29E+06	1.87E+01	3.17E+03	2.50E+01	1.23E+03	2.11E+02	1.12E+03	4.13E+02	1.09E+02	sNaive	1.87E+01	sARIMA	3.29E+06	3.00E+05	4.13E+02
Avg. Mean Absolute Scaled Error	9.72E+00	7.01E+00	6.79E+00	8.06E+00	7.24E+00	8.57E+00	1.03E+01	6.65E+00	8.54E+00	9.99E+00	7.12E+00	NNetar	6.65E+00	GPyTorch	1.03E+01	8.18E+00	8.06E+00
SD. Mean Absolute Scaled Error	1.28E+01	1.31E+01	1.29E+01	1.27E+01	1.25E+01	2.35E+01	1.81E+01	1.24E+01	1.32E+01	1.50E+01	1.24E+01	NNetar	1.24E+01	Theta	2.35E+01	1.44E+01	1.29E+01
Avg. Mean Under-Estimation Share	4.09E–01	3.06E–01	2.91E–01	4.41E–01	2.84E–01	3.21E–01	4.26E–01	3.11E–01	4.04E–01	4.27E–01	3.83E–01	TBATS	2.84E–01	sNaive	4.41E–01	3.64E–01	3.83E–01
SD. Mean Under-Estimation Share	3.74E–01	3.64E–01	3.68E–01	3.67E–01	3.67E–01	3.74E–01	3.65E–01	3.73E–01	3.79E–01	3.88E–01	3.73E–01	ETS	3.64E–01	SVR	3.88E–01	3.72E–01	3.73E–01
Avg. Mean Over-Estimation Share	4.33E–01	5.40E–01	5.75E–01	4.28E–01	5.73E–01	5.45E–01	4.03E–01	5.50E–01	4.23E–01	4.33E–01	4.73E–01	GPyTorch	4.03E–01	sARIMA	5.75E–01	4.88E–01	4.73E–01
SD. Mean Over-Estimation Share	4.90E–01	4.85E–01	4.91E–01	4.90E–01	4.88E–01	4.96E–01	4.80E–01	4.95E–01	4.77E–01	4.91E–01	4.95E–01	Random Forest	4.77E–01	Theta	4.96E–01	4.89E–01	4.90E–01
Avg. Mean Under-Accuracy Share	2.23E–01	1.75E–01	1.60E–01	1.81E–01	2.12E–01	1.52E–01	2.28E–01	1.58E–01	2.25E–01	2.15E–01	3.15E–01	Theta	1.52E–01	XGBoost	3.15E–01	2.04E–01	2.12E–01
SD. Mean Under-Accuracy Share	2.28E–01	1.70E–01	2.17E–01	2.27E–01	4.32E–01	1.71E–01	2.33E–01	1.71E–01	1.99E–01	2.68E–01	9.56E–01	ETS	1.70E–01	XGBoost	9.56E–01	2.97E–01	2.27E–01
Avg. Mean Over-Accuracy Share	2.05E+00	2.46E–01	9.05E–01	7.73E–02	2.44E–01	2.99E–01	2.98E–01	5.06E–01	3.11E–01	3.99E–01	2.83E–01	sNaive	7.73E–02	LSTM	2.05E+00	5.11E–01	2.99E–01
SD. Mean Over-Accuracy Share	6.48E+00	1.91E+00	8.24E+00	2.15E–01	1.53E+00	2.17E+00	2.04E+00	1.88E+00	1.84E+00	2.06E+00	1.83E+00	sNaive	2.15E–01	sARIMA	8.24E+00	2.75E+00	1.91E+00
Avg. Normalized Time	1.00E+04	3.91E+02	5.23E+06	1.16E+02	4.19E+03	8.28E+01	3.53E+03	3.87E+02	1.07E+03	4.45E+02	1.14E+02	Theta	8.28E+01	sARIMA	5.23E+06	4.77E+05	4.45E+02
SD. Normalized Time	8.55E+03	8.83E+02	2.74E+07	8.38E+02	4.41E+03	5.15E+02	4.70E+03	1.00E+03	2.68E+03	2.07E+03	8.05E+02	Theta	5.15E+02	sARIMA	2.74E+07	2.49E+06	2.07E+03

Table A2. Results of the OSAF economics use case.

Measures	LSTM	ETS	sARIMA	sNaive	TBATS	Theta	GPyTorch	NNetar	Random Forest	SVR	XGBoost	Min Measure	Min Value	Max Measure	Max Value	Mean	Median
Avg. Symmetrical Mean Absolute Percentage Error	4.06E+01	9.37E+00	2.27E+02	6.63E+00	2.32E+01	1.14E+01	1.30E+01	2.55E+01	2.16E+01	3.20E+01	7.05E+01	sNaive	6.63E+00	sARIMA	2.27E+02	4.37E+01	2.32E+01
SD. Symmetrical Mean Absolute Percentage Error	6.15E+01	2.26E+01	2.13E+03	1.22E+01	5.39E+01	2.93E+01	2.59E+01	4.97E+01	3.05E+01	4.77E+01	3.27E+02	sNaive	1.22E+01	sARIMA	2.13E+03	2.54E+02	4.77E+01
Avg. Mean Absolute Scaled Error	4.00E–02	2.00E–02	2.00E–02	2.00E–02	2.00E–02	2.00E–02	2.00E–02	2.00E–02	5.00E–02	8.00E–02	2.00E–02	ETS	2.00E–02	SVR	8.00E–02	3.00E–02	2.00E–02
SD. Mean Absolute Scaled Error	9.00E–02	6.00E–02	6.00E–02	6.00E–02	7.00E–02	6.00E–02	3.00E–02	7.00E–02	1.30E–01	2.20E–01	6.00E–02	GPyTorch	3.00E–02	SVR	2.20E–01	8.27E–02	6.00E–02
Avg. Mean Under-Estimation Share	5.40E–01	5.10E–01	3.90E–01	6.00E–01	4.00E–01	5.00E–01	5.60E–01	4.00E–01	5.10E–01	5.60E–01	5.50E–01	sARIMA	3.90E–01	sNaive	6.00E–01	5.02E–01	5.10E–01
SD. Mean Under-Estimation Share	5.00E–01	5.00E–01	4.90E–01	4.90E–01	4.90E–01	5.00E–01	5.00E–01	4.90E–01	5.00E–01	5.00E–01	5.00E–01	sARIMA	4.90E–01	LSTM	5.00E–01	4.96E–01	5.00E–01
Avg. Mean Over-Estimation Share	4.60E–01	4.90E–01	6.10E–01	4.00E–01	6.00E–01	5.00E–01	4.20E–01	6.00E–01	4.90E–01	4.40E–01	4.50E–01	sNaive	4.00E–01	sARIMA	6.10E–01	4.96E–01	4.90E–01
SD. Mean Over-Estimation Share	5.00E–01	5.00E–01	4.90E–01	4.90E–01	4.90E–01	5.00E–01	5.00E–01	4.90E–01	5.00E–01	5.00E–01	5.00E–01	sARIMA	4.90E–01	LSTM	5.00E–01	4.96E–01	5.00E–01
Avg. Mean Under-Accuracy Share	8.00E–02	3.00E–02	5.00E–02	3.00E–02	2.20E–01	2.00E–02	5.00E–02	2.00E–02	6.00E–02	8.00E–02	5.50E–01	Theta	2.00E–02	XGBoost	5.50E–01	1.08E–01	5.00E–02
SD. Mean Under-Accuracy Share	1.30E–01	5.00E–02	2.30E–01	6.00E–02	1.11E+00	5.00E–02	8.00E–02	5.00E–02	1.00E–01	1.70E–01	2.97E+00	ETS	5.00E–02	XGBoost	2.97E+00	4.55E–01	1.00E–01
Avg. Mean Over-Accuracy Share	7.71E+00	6.70E–01	6.80E–01	4.00E–02	7.10E–01	8.60E–01	7.00E–01	1.70E+00	8.00E–01	1.26E+00	8.30E–01	sNaive	4.00E–02	LSTM	7.71E+00	1.45E+00	8.00E–01
SD. Mean Over-Accuracy Share	2.46E+01	6.17E+00	5.46E+00	1.70E–01	5.12E+00	7.07E+00	6.05E+00	6.41E+00	5.94E+00	7.27E+00	6.50E+00	sNaive	1.70E–01	LSTM	2.46E+01	7.34E+00	6.17E+00
Avg. Normalized Time	7.22E+03	4.13E+02	5.70E+03	1.00E+00	4.18E+03	4.58E+00	1.86E+03	1.04E+02	8.31E+01	1.29E+01	3.17E+00	sNaive	1.00E+00	LSTM	7.22E+03	1.78E+03	1.04E+02
SD. Normalized Time	3.64E+03	3.18E+02	1.60E+04	0.00E+00	3.95E+03	3.17E+00	1.63E+03	1.01E+02	1.05E+02	2.06E+01	1.11E+00	sNaive	0.00E+00	sARIMA	1.60E+04	2.35E+03	1.05E+02

Table A3. Results of the OSAF finance use case.

Measures	LSTM	ETS	sARIMA	sNaive	TBATS	Theta	GPyTorch	NNetar	Random Forest	SVR	XGBoost	Min Measure	Min Value	Max Measure	Max Value	Mean	Median
Avg. Symmetrical Mean Absolute Percentage Error	1.76E+01	4.69E+00	3.05E+00	1.83E+01	4.05E+00	4.91E+00	2.65E+01	3.99E+00	1.44E+01	1.86E+01	4.51E+00	sARIMA	3.05E+00	GPyTorch	2.65E+01	1.10E+01	4.91E+00
SD. Symmetrical Mean Absolute Percentage Error	2.34E+01	1.72E+01	6.96E+00	2.58E+01	1.30E+01	1.77E+01	3.61E+01	1.40E+01	2.10E+01	3.15E+01	8.96E+00	sARIMA	6.96E+00	GPyTorch	3.61E+01	1.96E+01	1.77E+01
Avg. Mean Absolute Scaled Error	4.54E−02	1.05E−02	9.56E−03	1.47E−02	1.09E−02	1.31E−02	8.95E−03	9.03E−03	5.14E−02	7.77E−02	1.28E−02	GPyTorch	8.95E−03	SVR	7.77E−02	2.40E−02	1.28E−02
SD. Mean Absolute Scaled Error	1.07E−01	2.54E−02	2.44E−02	3.91E−02	2.67E−02	3.76E−02	1.91E−02	2.28E−02	1.72E−01	2.77E−01	3.91E−02	GPyTorch	1.91E−02	SVR	2.77E−01	7.18E−02	3.76E−02
Avg. Mean Under-Estimation Share	6.60E−01	4.00E−01	4.10E−01	6.80E−01	3.80E−01	4.10E−01	7.20E−01	4.70E−01	7.00E−01	6.80E−01	5.90E−01	TBATS	3.80E−01	GPyTorch	7.20E−01	5.55E−01	5.90E−01
SD. Mean Under-Estimation Share	4.76E−01	4.92E−01	4.94E−01	4.69E−01	4.88E−01	4.94E−01	4.51E−01	5.02E−01	4.61E−01	4.69E−01	4.94E−01	GPyTorch	4.51E−01	NNetar	5.02E−01	4.81E−01	4.88E−01
Avg. Mean Over-Estimation Share	3.40E−01	6.00E−01	5.90E−01	3.20E−01	6.20E−01	5.90E−01	2.60E−01	5.30E−01	3.00E−01	3.20E−01	4.10E−01	GPyTorch	2.60E−01	TBATS	6.20E−01	4.44E−01	4.10E−01
SD. Mean Over-Estimation Share	4.76E−01	4.92E−01	4.94E−01	4.69E−01	4.88E−01	4.94E−01	4.41E−01	5.02E−01	4.61E−01	4.69E−01	4.94E−01	GPyTorch	4.41E−01	NNetar	5.02E−01	4.80E−01	4.88E−01
Avg. Mean Under-Accuracy Share	1.00E−01	8.56E−03	9.47E−03	1.14E−01	8.20E−03	8.78E−03	1.64E−01	1.01E−02	8.13E−02	1.01E−01	1.95E−02	TBATS	8.20E−03	GPyTorch	1.64E−01	5.68E−02	1.95E−02
SD. Mean Under-Accuracy Share	1.48E−01	2.24E−02	2.65E−02	1.75E−01	2.06E−02	2.26E−02	2.28E−01	2.50E−02	1.10E−01	1.85E−01	4.94E−02	TBATS	2.06E−02	GPyTorch	2.28E−01	9.21E−02	4.94E−02
Avg. Mean Over-Accuracy Share	8.30E−02	1.00E−01	2.38E−02	5.62E−02	5.14E−02	1.14E−01	1.33E−01	5.80E−02	1.08E−01	8.33E−02	2.78E−02	sARIMA	2.38E−02	GPyTorch	1.33E−01	7.63E−02	8.30E−02
SD. Mean Over-Accuracy Share	3.34E−01	7.42E−01	9.15E−02	2.43E−01	2.95E−01	8.66E−01	9.61E−01	4.09E−01	6.44E−01	3.95E−01	9.74E−02	sARIMA	9.15E−02	GPyTorch	9.61E−01	4.62E−01	3.95E−01
Avg. Normalized Time	1.80E+04	1.99E+02	2.08E+07	1.00E+00	4.74E+03	8.13E+00	6.56E+03	5.68E+02	2.18E+03	3.45E+02	4.93E+00	sNaive	1.00E+00	sARIMA	2.08E+07	1.89E+06	5.68E+02
SD. Normalized Time	1.56E+04	2.73E+02	1.09E+08	0.00E+00	4.27E+03	8.67E+00	8.84E+03	6.72E+02	3.89E+03	5.83E+02	2.52E+00	sNaive	0.00E+00	sARIMA	1.09E+08	9.91E+06	6.72E+02

Table A4. Results of the OSAF human use case.

Measures	LSTM	ETS	sARIMA	sNaive	TBATS	Theta	GPyTorch	NNetar	Random Forest	SVR	XGBoost	Min Measure	Min Value	Max Measure	Max Value	Mean	Median
Avg. Symmetrical Mean Absolute Percentage Error	2.00E+04	8.57E+02	1.82E+06	1.00E+00	1.35E+04	1.38E+01	6.53E+03	7.63E+02	2.85E+03	9.16E+02	4.44E+00	sNaive	1.00E+00	sARIMA	1.82E+06	1.69E+05	9.16E+02
SD. Symmetrical Mean Absolute Percentage Error	1.53E+04	1.44E+03	1.31E+07	0.00E+00	1.26E+04	2.68E+01	4.82E+03	7.27E+02	4.40E+03	1.53E+03	2.00E+00	sNaive	0.00E+00	sARIMA	1.31E+07	1.20E+06	1.53E+03
Avg. Mean Absolute Scaled Error	3.88E+01	2.80E+01	2.71E+01	3.22E+01	2.89E+01	3.42E+01	4.11E+01	2.66E+01	3.41E+01	3.98E+01	2.85E+01	NNetar	2.66E+01	GPyTorch	4.11E+01	3.27E+01	3.22E+01
SD. Mean Absolute Scaled Error	5.09E+01	5.23E+01	5.17E+01	5.08E+01	4.98E+01	9.40E+01	7.25E+01	4.95E+01	5.22E+01	5.95E+01	4.95E+01	NNetar	4.95E+01	Theta	9.40E+01	5.75E+01	5.17E+01
Avg. Mean Under-Estimation Share	7.53E−03	2.31E−03	3.11E−03	3.48E−03	4.10E−03	4.02E−03	3.70E−03	4.00E−03	1.57E−02	1.66E−02	3.50E−03	ETS	2.31E−03	SVR	1.66E−02	6.19E−03	4.00E−03
SD. Mean Under-Estimation Share	2.06E−02	3.60E−03	7.36E−03	9.91E−03	1.09E−02	1.16E−02	8.43E−03	1.19E−02	6.42E−02	8.26E−02	9.46E−03	ETS	3.60E−03	SVR	8.26E−02	2.19E−02	1.09E−02
Avg. Mean Over-Estimation Share	3.60E−01	3.80E−01	4.60E−01	4.70E−01	4.20E−01	4.60E−01	3.50E−01	4.40E−01	2.90E−01	4.20E−01	4.20E−01	Random Forest	2.90E−01	sNaive	4.70E−01	4.06E−01	4.20E−01
SD. Mean Over-Estimation Share	4.82E−01	4.88E−01	5.01E−01	5.02E−01	4.96E−01	5.01E−01	4.79E−01	4.99E−01	4.56E−01	4.96E−01	4.96E−01	Random Forest	4.56E−01	sNaive	5.02E−01	4.91E−01	4.96E−01
Avg. Mean Under-Accuracy Share	6.40E−01	6.20E−01	5.40E−01	5.00E−01	5.80E−01	5.40E−01	6.40E−01	5.60E−01	7.10E−01	5.80E−01	5.80E−01	sNaive	5.00E−01	Random Forest	7.10E−01	5.90E−01	5.80E−01
SD. Mean Under-Accuracy Share	4.82E−01	4.88E−01	5.01E−01	5.03E−01	4.96E−01	5.01E−01	4.82E−01	4.99E−01	4.56E−01	4.96E−01	4.96E−01	Random Forest	4.56E−01	sNaive	5.03E−01	4.91E−01	4.96E−01
Avg. Mean Over-Accuracy Share	7.39E−02	3.57E−02	2.74E+00	6.29E−02	5.46E−02	5.33E−02	7.08E−02	3.79E−02	4.66E−02	9.38E−02	5.50E−02	ETS	3.57E−02	sARIMA	2.74E+00	3.02E−01	5.50E−02
SD. Mean Over-Accuracy Share	1.53E−01	1.12E−01	2.70E+01	1.18E−01	1.29E−01	1.73E−01	1.86E−01	7.45E−02	1.11E−01	2.03E−01	1.28E−01	NNetar	7.45E−02	sARIMA	2.70E+01	2.58E+00	1.29E−01
Avg. Normalized Time	3.16E+02	3.42E+02	4.55E+02	4.62E+02	3.38E+02	3.11E+02	4.53E+02	3.92E+02	3.97E+02	4.32E+02	4.46E+02	Theta	3.11E+02	sNaive	4.62E+02	3.95E+02	3.97E+02
SD. Normalized Time	1.96E+03	2.28E+03	3.29E+03	3.35E+03	2.30E+03	2.04E+03	3.22E+03	2.64E+03	2.78E+03	3.24E+03	3.22E+03	LSTM	1.96E+03	sNaive	3.35E+03	2.76E+03	2.78E+03

Table A5. Results of the OSAF nature use case.

Measures	LSTM	ETS	sARIMA	sNaive	TBATS	Theta	GPyTorch	NNetar	Random Forest	SVR	XGBoost	Min Measure	Min Value	Max Measure	Max Value	Mean	Median
Avg. Symmetrical Mean Absolute Percentage Error	2.75E+01	1.67E+01	1.77E+01	2.31E+01	1.52E+01	1.55E+01	2.32E+01	2.33E+01	2.72E+01	2.70E+01	3.47E+01	TBATS	1.52E+01	XGBoost	3.47E+01	2.28E+01	2.32E+01
SD. Symmetrical Mean Absolute Percentage Error	3.51E+01	2.88E+01	2.56E+01	3.68E+01	2.90E+01	2.62E+01	3.73E+01	5.26E+01	4.73E+01	4.77E+01	9.94E+01	sARIMA	2.56E+01	XGBoost	9.94E+01	4.23E+01	3.68E+01
Avg. Mean Absolute Scaled Error	1.00E−02	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E−02	2.00E−02	0.00E+00	ETS	0.00E+00	SVR	2.00E−02	3.64E−03	0.00E+00
SD. Mean Absolute Scaled Error	2.00E−02	1.00E−02	1.00E−02	1.00E−02	1.00E−02	1.00E−02	1.00E−02	1.00E−02	1.00E−01	1.50E−01	1.00E−02	ETS	1.00E−02	SVR	1.50E−01	3.18E−02	1.00E−02
Avg. Mean Under-Estimation Share	4.30E−01	3.10E−01	3.60E−01	4.80E−01	3.50E−01	3.70E−01	4.20E−01	3.70E−01	3.90E−01	4.50E−01	3.90E−01	ETS	3.10E−01	sNaive	4.80E−01	3.93E−01	3.90E−01
SD. Mean Under-Estimation Share	5.00E−01	4.60E−01	4.80E−01	5.00E−01	4.80E−01	4.90E−01	5.00E−01	4.90E−01	4.90E−01	5.00E−01	4.90E−01	ETS	4.60E−01	LSTM	5.00E−01	4.89E−01	4.90E−01
Avg. Mean Over-Estimation Share	5.70E−01	6.90E−01	6.40E−01	5.20E−01	6.50E−01	6.30E−01	5.80E−01	6.30E−01	6.10E−01	5.50E−01	6.10E−01	sNaive	5.20E−01	ETS	6.90E−01	6.07E−01	6.10E−01
SD. Mean Over-Estimation Share	5.00E−01	4.60E−01	4.80E−01	5.00E−01	4.80E−01	4.90E−01	5.00E−01	4.90E−01	4.90E−01	5.00E−01	4.90E−01	ETS	4.60E−01	LSTM	5.00E−01	4.89E−01	4.90E−01
Avg. Mean Under-Accuracy Share	7.00E−02	4.00E−02	4.00E−02	8.00E−02	4.00E−02	4.00E−02	6.00E−02	4.00E−02	5.00E−02	1.00E−01	1.10E−01	ETS	4.00E−02	XGBoost	1.10E−01	6.09E−02	5.00E−02
SD. Mean Under-Accuracy Share	1.50E−01	1.20E−01	1.10E−01	1.70E−01	1.00E−01	1.10E−01	1.40E−01	1.10E−01	1.30E−01	2.20E−01	3.10E−01	TBATS	1.00E−01	XGBoost	3.10E−01	1.52E−01	1.30E−01
Avg. Mean Over-Accuracy Share	3.40E−01	1.80E−01	1.80E−01	1.50E−01	1.60E−01	1.70E−01	2.90E−01	2.30E−01	2.90E−01	1.60E−01	2.20E−01	sNaive	1.50E−01	LSTM	3.40E−01	2.15E−01	1.80E−01
SD. Mean Over-Accuracy Share	8.60E−01	6.10E−01	4.60E−01	3.30E−01	5.70E−01	5.90E−01	9.50E−01	6.40E−01	6.80E−01	3.90E−01	5.90E−01	sNaive	3.30E−01	GPyTorch	9.50E−01	6.06E−01	5.90E−01
Avg. Normalized Time	1.45E+04	6.11E+02	1.47E+05	1.00E+00	7.50E+03	7.65E+00	5.25E+03	4.84E+02	1.61E+03	9.88E+02	4.02E+00	sNaive	1.00E+00	sARIMA	1.47E+05	1.62E+04	9.88E+02
SD. Normalized Time	1.30E+04	6.64E+02	6.84E+05	0.00E+00	7.12E+03	5.69E+00	5.09E+03	5.90E+02	3.93E+03	4.42E+03	1.71E+00	sNaive	0.00E+00	sARIMA	6.84E+05	6.53E+04	3.93E+03

Table A6. Results of the MSAF combined use cases via mean averaging.

Measures	LSTM	ETS	sARIMA	sNaive	TBATS	Theta	GPyTorch	NNetar	Random Forest	SVR	XGBoost	Min Measure	Min	Max Measure	Max	Mean	Median
Avg. Symmetrical Mean Absolute Percentage Error	4.13E+01	2.89E+01	2.06E+01	2.18E+01	2.36E+01	2.11E+01	2.57E+01	3.52E+02	3.67E+01	5.86E+01	2.38E+01	sARIMA	2.06E+01	NNetar	3.52E+02	5.95E+01	2.57E+01
SD. Symmetrical Mean Absolute Percentage Error	7.47E+01	5.14E+01	3.30E+01	2.78E+01	5.20E+01	3.86E+01	3.20E+01	3.30E+03	1.19E+02	3.03E+02	3.21E+01	sNaive	2.78E+01	NNetar	3.30E+03	3.69E+02	5.14E+01
Avg. Mean Absolute Scaled Error	2.26E+00	1.20E+00	7.30E−01	1.02E+00	8.80E−01	9.43E−01	1.13E+00	1.12E+00	1.68E+00	2.14E+00	1.00E+00	sARIMA	7.30E−01	LSTM	2.26E+00	1.28E+00	1.12E+00
SD. Mean Absolute Scaled Error	6.06E+00	3.10E+00	1.47E+00	2.41E+00	1.97E+00	2.32E+00	2.43E+00	2.65E+00	4.96E+00	6.99E+00	2.43E+00	sARIMA	1.47E+00	SVR	6.99E+00	3.35E+00	2.43E+00
Avg. Mean Under-Estimation Share	6.32E−01	5.32E−01	5.23E−01	6.12E−01	5.45E−01	5.77E−01	6.17E−01	5.35E−01	5.91E−01	6.02E−01	6.07E−01	sARIMA	5.23E−01	LSTM	6.32E−01	5.79E−01	5.91E−01
SD. Mean Under-Estimation Share	2.84E−01	3.14E−01	2.78E−01	2.69E−01	2.62E−01	2.86E−01	2.80E−01	2.68E−01	2.93E−01	2.69E−01	2.82E−01	TBATS	2.62E−01	ETS	3.14E−01	2.80E−01	2.80E−01
Avg. Mean Over-Estimation Share	3.68E−01	4.68E−01	4.77E−01	3.85E−01	4.55E−01	4.23E−01	3.81E−01	4.65E−01	4.09E−01	3.98E−01	3.93E−01	LSTM	3.68E−01	sARIMA	4.77E−01	4.20E−01	4.09E−01
SD. Mean Over-Estimation Share	2.84E−01	3.14E−01	2.78E−01	2.68E−01	2.62E−01	2.86E−01	2.80E−01	2.68E−01	2.93E−01	2.69E−01	2.82E−01	TBATS	2.62E−01	ETS	3.14E−01	2.80E−01	2.80E−01
Avg. Mean Under-Accuracy Share	2.39E−01	7.04E−01	2.17E−01	1.54E−01	1.52E−01	1.45E−01	1.71E−01	1.57E−01	1.61E−01	1.93E−01	3.17E−01	Theta	1.45E−01	ETS	7.04E−01	2.37E−01	1.71E−01
SD. Mean Under-Accuracy Share	1.88E−01	5.72E+00	8.66E−01	1.53E−01	2.41E−01	2.35E−01	1.64E−01	2.62E−01	1.54E−01	1.96E−01	1.40E+00	sNaive	1.53E−01	ETS	5.72E+00	8.71E−01	2.35E−01
Avg. Mean Over-Accuracy Share	1.03E+01	2.02E+01	6.07E+00	5.46E+00	3.48E+00	1.65E+01	5.34E+00	7.48E+00	1.02E+01	9.47E+00	7.14E+00	TBATS	3.48E+00	ETS	2.02E+01	9.23E+00	7.48E+00
SD. Mean Over-Accuracy Share	3.68E+01	1.81E+02	3.99E+01	4.08E+01	1.61E+01	1.13E+02	2.11E+01	3.43E+01	6.52E+01	4.58E+01	4.92E+01	TBATS	1.61E+01	ETS	1.81E+02	5.84E+01	4.08E+01
Avg. Normalized Time	1.13E+04	4.25E+02	8.48E+05	1.00E+00	7.59E+03	1.01E+01	9.39E+03	5.92E+02	1.42E+03	6.19E+02	6.73E+00	sNaive	1.00E+00	sARIMA	8.48E+05	7.99E+04	6.19E+02
SD. Normalized Time	1.11E+04	5.16E+02	5.57E+06	0.00E+00	8.23E+03	1.11E+01	8.52E+03	6.74E+02	3.06E+03	1.76E+03	4.77E+00	sNaive	0.00E+00	sARIMA	5.57E+06	5.10E+05	1.76E+03

Table A7. Results of the MSAF economics use case.

Measures	LSTM	ETS	sARIMA	sNaive	TBATS	Theta	GPyTorch	NNetar	Random Forest	SVR	XGBoost	Min Measure	Min	Max Measure	Max	Mean	Median
Avg. Symmetrical Mean Absolute Percentage Error	3.14E+01	1.30E+01	1.49E+01	1.33E+01	1.42E+01	1.34E+01	1.73E+01	1.65E+01	1.98E+01	2.51E+01	1.76E+01	ETS	1.30E+01	LSTM	3.14E+01	1.79E+01	1.65E+01
SD. Symmetrical Mean Absolute Percentage Error	2.74E+01	1.63E+01	2.59E+01	1.22E+01	2.12E+01	2.15E+01	1.34E+01	1.69E+01	1.83E+01	2.94E+01	2.00E+01	sNaive	1.22E+01	SVR	2.94E+01	2.02E+01	2.00E+01
Avg. Mean Absolute Scaled Error	2.08E+00	6.58E−01	5.33E−01	8.97E−01	5.87E−01	7.68E−01	1.04E+00	9.08E−01	1.69E+00	2.20E+00	8.65E−01	sARIMA	5.33E−01	SVR	2.20E+00	1.11E+00	8.97E−01
SD. Mean Absolute Scaled Error	5.25E+00	1.42E+00	9.76E−01	2.21E+00	9.92E−01	1.85E+00	2.07E+00	2.02E+00	5.73E+00	8.18E+00	2.17E+00	sARIMA	9.76E−01	SVR	8.18E+00	2.99E+00	2.07E+00
Avg. Mean Under-Estimation Share	7.40E−01	5.55E−01	4.93E−01	6.79E−01	4.97E−01	6.01E−01	7.18E−01	5.29E−01	7.37E−01	7.26E−01	7.09E−01	sARIMA	4.93E−01	LSTM	7.40E−01	6.35E−01	6.79E−01
SD. Mean Under-Estimation Share	3.10E−01	3.23E−01	3.04E−01	3.00E−01	3.05E−01	3.08E−01	3.08E−01	2.95E−01	2.71E−01	2.71E−01	2.89E−01	Random Forest	2.71E−01	ETS	3.23E−01	2.99E−01	3.04E−01
Avg. Mean Over-Estimation Share	2.60E−01	4.45E−01	5.07E−01	3.21E−01	5.03E−01	3.99E−01	2.82E−01	4.71E−01	2.63E−01	2.74E−01	2.91E−01	LSTM	2.60E−01	sARIMA	5.07E−01	3.65E−01	3.21E−01
SD. Mean Over-Estimation Share	3.10E−01	3.23E−01	3.04E−01	3.00E−01	3.05E−01	3.08E−01	3.08E−01	2.95E−01	2.71E−01	2.71E−01	2.89E−01	Random Forest	2.71E−01	ETS	3.23E−01	2.99E−01	3.04E−01
Avg. Mean Under-Accuracy Share	2.03E−01	7.44E−02	7.45E−02	1.02E−01	9.51E−02	9.05E−02	1.21E−01	7.85E−02	1.35E−01	1.67E−01	2.60E−01	ETS	7.44E−02	XGBoost	2.60E−01	1.27E−01	1.02E−01
SD. Mean Under-Accuracy Share	1.78E−01	9.49E−02	1.01E−01	7.57E−02	2.43E−01	1.23E−01	9.49E−02	7.09E−02	1.23E−01	1.73E−01	7.19E−01	NNetar	7.09E−02	XGBoost	7.19E−01	1.82E−01	1.23E−01
Avg. Mean Over-Accuracy Share	4.37E+00	1.58E+00	9.44E−01	9.04E−01	1.06E+00	2.34E+00	1.05E+00	2.84E+00	1.51E+00	1.38E+00	2.01E+00	sNaive	9.04E−01	LSTM	4.37E+00	1.82E+00	1.51E+00
SD. Mean Over-Accuracy Share	1.40E+01	4.84E+00	2.71E+00	2.69E+00	3.46E+00	1.52E+01	3.63E+00	9.09E+00	5.33E+00	4.09E+00	6.73E+00	sNaive	2.69E+00	Theta	1.52E+01	6.53E+00	4.84E+00
Avg. Normalized Time	7.46E+03	3.40E+02	7.29E+03	1.00E+00	2.63E+03	6.04E+00	3.39E+03	1.18E+02	6.23E+01	1.31E+01	6.41E+00	sNaive	1.00E+00	LSTM	7.46E+03	1.94E+03	1.18E+02
SD. Normalized Time	2.33E+03	2.52E+02	2.36E+04	0.00E+00	1.85E+03	8.16E+00	2.93E+03	1.13E+02	7.45E+01	2.12E+01	5.13E+00	sNaive	0.00E+00	sARIMA	2.36E+04	2.84E+03	1.13E+02

Table A8. Results of the MSAF finance use case.

Measures	LSTM	ETS	sARIMA	sNaive	TBATS	Theta	GPyTorch	NNetar	Random Forest	SVR	XGBoost	Min Measure	Min	Max Measure	Max	Mean	Median
Avg. Symmetrical Mean Absolute Percentage Error	4.04E+01	1.76E+01	1.78E+01	2.44E+01	1.80E+01	1.80E+01	3.16E+01	1.33E+03	2.49E+01	3.04E+01	1.92E+01	ETS	1.76E+01	NNetar	1.33E+03	1.43E+02	2.44E+01
SD. Symmetrical Mean Absolute Percentage Error	3.54E+01	1.80E+01	2.21E+01	2.44E+01	1.75E+01	1.82E+01	3.28E+01	1.31E+04	2.35E+01	3.08E+01	1.96E+01	TBATS	1.75E+01	NNetar	1.31E+04	1.21E+03	2.35E+01
Avg. Mean Absolute Scaled Error	4.91E+00	1.88E+00	1.58E+00	2.25E+00	1.94E+00	1.96E+00	2.36E+00	2.39E+00	3.62E+00	4.80E+00	2.13E+00	sARIMA	1.58E+00	LSTM	4.91E+00	2.71E+00	2.25E+00
SD. Mean Absolute Scaled Error	1.54E+01	5.34E+00	4.09E+00	5.92E+00	5.32E+00	5.27E+00	5.82E+00	6.92E+00	1.17E+01	1.67E+01	5.95E+00	sARIMA	4.09E+00	SVR	1.67E+01	8.04E+00	5.92E+00
Avg. Mean Under-Estimation Share	7.45E–01	6.68E–01	6.48E–01	6.99E–01	6.80E–01	7.03E–01	7.05E–01	6.98E–01	7.08E–01	7.00E–01	7.07E–01	sARIMA	6.48E–01	LSTM	7.45E–01	6.96E–01	7.00E–01
SD. Mean Under-Estimation Share	3.81E–01	3.26E–01	3.31E–01	3.42E–01	3.23E–01	3.02E–01	3.68E–01	3.29E–01	3.69E–01	3.55E–01	3.09E–01	Theta	3.02E–01	LSTM	3.81E–01	3.40E–01	3.31E–01
Avg. Mean Over-Estimation Share	2.55E–01	3.32E–01	3.52E–01	3.01E–01	3.20E–01	2.97E–01	2.90E–01	3.02E–01	2.92E–01	3.00E–01	2.93E–01	LSTM	2.55E–01	sARIMA	3.52E–01	3.03E–01	3.00E–01
SD. Mean Over-Estimation Share	3.81E–01	3.26E–01	3.31E–01	3.42E–01	3.23E–01	3.02E–01	3.69E–01	3.29E–01	3.69E–01	3.55E–01	3.09E–01	Theta	3.02E–01	LSTM	3.81E–01	3.40E–01	3.31E–01
Avg. Mean Under-Accuracy Share	2.67E–01	1.24E–01	1.22E–01	1.72E–01	1.27E–01	1.31E–01	2.04E–01	1.50E–01	1.73E–01	2.04E–01	1.40E–01	sARIMA	1.22E–01	LSTM	2.67E–01	1.65E–01	1.50E–01
SD. Mean Under-Accuracy Share	2.43E–01	1.23E–01	1.33E–01	1.66E–01	1.24E–01	1.23E–01	2.14E–01	1.42E–01	1.65E–01	2.03E–01	1.26E–01	ETS	1.23E–01	LSTM	2.43E–01	1.60E–01	1.42E–01
Avg. Mean Over-Accuracy Share	1.20E–01	1.86E–01	2.53E–01	1.98E–01	1.55E–01	2.01E–01	1.79E–01	2.92E–01	1.72E–01	1.55E–01	2.21E–01	LSTM	1.20E–01	NNetar	2.92E–01	1.94E–01	1.86E–01
SD. Mean Over-Accuracy Share	3.94E–01	6.15E–01	1.23E+00	7.55E–01	4.50E–01	7.38E–01	7.06E–01	1.66E+00	7.17E–01	4.23E–01	9.84E–01	LSTM	3.94E–01	NNetar	1.66E+00	7.88E–01	7.17E–01
Avg. Normalized Time	1.18E+04	1.69E+02	2.35E+06	1.00E+00	5.35E+03	8.58E+00	9.95E+03	6.78E+02	1.64E+03	3.60E+02	7.27E+00	sNaive	1.00E+00	sARIMA	2.35E+06	2.17E+05	6.78E+02
SD. Normalized Time	4.80E+03	2.07E+02	1.73E+07	0.00E+00	6.44E+03	8.40E+00	1.12E+04	8.21E+02	3.09E+03	5.97E+02	5.20E+00	sNaive	0.00E+00	sARIMA	1.73E+07	1.57E+06	8.21E+02

Appendix B Multi Step Ahead Forecast Results

Table A9. Results of the MSAF human use case.

Measures	LSTM	ETS	sARIMA	sNaive	TBATS	Theta	GPyTorch	NNetar	Random Forest	SVR	XGBoost	Min Measure	Min	Max Measure	Max	Mean	Median
Avg. Symmetrical Mean Absolute Percentage Error	5.82E+01	4.66E+01	2.79E+01	2.36E+01	4.25E+01	3.12E+01	2.99E+01	3.31E+01	7.56E+01	1.47E+02	2.84E+01	sNaive	2.36E+01	SVR	1.47E+02	4.95E+01	3.31E+01
SD. Symmetrical Mean Absolute Percentage Error	1.97E+02	6.56E+01	5.52E+01	3.51E+01	1.47E+02	9.01E+01	5.04E+01	6.07E+01	4.07E+02	1.08E+03	4.29E+01	sNaive	3.51E+01	SVR	1.08E+03	2.03E+02	6.56E+01
Avg. Mean Absolute Scaled Error	1.16E+00	1.42E+00	5.03E−01	5.12E−01	6.34E−01	6.49E−01	6.48E−01	6.62E−01	8.24E−01	1.01E+00	5.45E−01	sARIMA	5.03E−01	ETS	1.42E+00	7.78E−01	6.49E−01
SD. Mean Absolute Scaled Error	1.99E+00	2.88E+00	5.39E−01	8.91E−01	1.03E+00	1.59E+00	1.03E+00	9.09E−01	1.44E+00	1.84E+00	8.69E−01	sARIMA	5.39E−01	ETS	2.88E+00	1.36E+00	1.03E+00
Avg. Mean Under-Estimation Share	5.39E−01	4.57E−01	4.87E−01	5.36E−01	4.94E−01	5.00E−01	5.27E−01	4.86E−01	4.38E−01	4.74E−01	5.24E−01	Random Forest	4.38E−01	LSTM	5.39E−01	4.96E−01	4.94E−01
SD. Mean Under-Estimation Share	2.37E−01	3.23E−01	2.74E−01	2.19E−01	2.15E−01	2.74E−01	2.22E−01	2.37E−01	2.59E−01	2.30E−01	2.53E−01	TBATS	2.15E−01	ETS	3.23E−01	2.49E−01	2.37E−01
Avg. Mean Over-Estimation Share	4.61E−01	5.43E−01	5.13E−01	4.56E−01	5.06E−01	5.00E−01	4.71E−01	5.14E−01	5.62E−01	5.26E−01	4.76E−01	sNaive	4.56E−01	Random Forest	5.62E−01	5.03E−01	5.06E−01
SD. Mean Over-Estimation Share	2.37E−01	3.23E−01	2.74E−01	2.17E−01	2.15E−01	2.74E−01	2.21E−01	2.37E−01	2.59E−01	2.30E−01	2.53E−01	TBATS	2.15E−01	ETS	3.23E−01	2.49E−01	2.37E−01
Avg. Mean Under-Accuracy Share	2.61E−01	2.47E+00	5.28E−01	1.76E−01	2.36E−01	2.10E−01	1.99E−01	2.48E−01	1.83E−01	2.30E−01	6.89E−01	sNaive	1.76E−01	ETS	2.47E+00	4.93E−01	2.36E−01
SD. Mean Under-Accuracy Share	1.61E−01	2.25E+01	3.08E+00	2.18E−01	4.35E−01	5.32E−01	1.91E−01	6.89E−01	1.95E−01	2.43E−01	4.53E+00	LSTM	1.61E−01	ETS	2.25E+01	2.98E+00	4.35E−01
Avg. Mean Over-Accuracy Share	3.55E+01	7.74E+01	2.26E+01	2.02E+01	1.24E+01	6.31E+01	1.95E+01	2.59E+01	3.82E+01	3.59E+01	2.58E+01	TBATS	1.24E+01	ETS	7.74E+01	3.42E+01	2.59E+01
SD. Mean Over-Accuracy Share	1.30E+02	7.10E+02	1.55E+02	1.58E+02	5.96E+01	4.35E+02	7.82E+01	1.25E+02	2.53E+02	1.77E+02	1.88E+02	TBATS	5.96E+01	ETS	7.10E+02	2.24E+02	1.58E+02
Avg. Normalized Time	1.25E+04	6.90E+02	9.05E+05	1.00E+00	1.48E+04	1.53E+01	1.41E+04	9.60E+02	2.40E+03	1.06E+03	6.72E+00	sNaive	1.00E+00	sARIMA	9.05E+05	8.65E+04	1.06E+03
SD. Normalized Time	5.35E+03	1.09E+03	4.50E+06	0.00E+00	1.48E+04	1.68E+01	1.04E+04	9.68E+02	4.77E+03	2.13E+03	4.23E+00	sNaive	0.00E+00	sARIMA	4.50E+06	4.13E+05	2.13E+03

Table A10. Results of the MSAF nature use case.

Measures	LSTM	ETS	sARIMA	sNaive	TBATS	Theta	GPyTorch	NNetar	Random Forest	SVR	XGBoost	Min Measure	Min	Max Measure	Max	Mean	Median
Avg. Symmetrical Mean Absolute Percentage Error	3.53E+01	3.85E+01	2.19E+01	2.60E+01	1.98E+01	2.18E+01	2.40E+01	2.88E+01	2.65E+01	3.18E+01	3.01E+01	TBATS	1.98E+01	ETS	3.85E+01	2.77E+01	2.65E+01
SD. Symmetrical Mean Absolute Percentage Error	3.85E+01	1.06E+02	2.86E+01	3.97E+01	2.26E+01	2.45E+01	3.14E+01	4.08E+01	2.71E+01	6.99E+01	4.59E+01	TBATS	2.26E+01	ETS	1.06E+02	4.32E+01	3.85E+01
Avg. Mean Absolute Scaled Error	9.07E−01	8.32E−01	3.07E−01	4.25E−01	3.58E−01	3.95E−01	4.81E−01	5.22E−01	5.70E−01	5.44E−01	4.73E−01	sARIMA	3.07E−01	LSTM	9.07E−01	5.29E−01	4.81E−01
SD. Mean Absolute Scaled Error	1.60E+00	2.76E+00	2.81E−01	6.40E−01	5.34E−01	5.82E−01	8.10E−01	7.49E−01	9.87E−01	1.28E+00	7.46E−01	sARIMA	2.81E−01	ETS	2.76E+00	9.97E−01	7.49E−01
Avg. Mean Under-Estimation Share	5.03E−01	4.49E−01	4.62E−01	5.34E−01	5.10E−01	5.04E−01	5.18E−01	4.28E−01	4.80E−01	5.08E−01	4.88E−01	NNetar	4.28E−01	sNaive	5.34E−01	4.89E−01	5.03E−01
SD. Mean Under-Estimation Share	2.10E−01	2.84E−01	2.01E−01	2.14E−01	2.04E−01	2.61E−01	2.23E−01	2.12E−01	2.71E−01	2.17E−01	2.79E−01	sARIMA	2.01E−01	ETS	2.84E−01	2.34E−01	2.17E−01
Avg. Mean Over-Estimation Share	4.97E−01	5.51E−01	5.38E−01	4.64E−01	4.90E−01	4.96E−01	4.82E−01	5.72E−01	5.20E−01	4.92E−01	5.12E−01	sNaive	4.64E−01	NNetar	5.72E−01	5.10E−01	4.97E−01
SD. Mean Over-Estimation Share	2.10E−01	2.84E−01	2.01E−01	2.12E−01	2.04E−01	2.61E−01	2.23E−01	2.12E−01	2.71E−01	2.17E−01	2.79E−01	sARIMA	2.01E−01	ETS	2.84E−01	2.34E−01	2.17E−01
Avg. Mean Under-Accuracy Share	2.25E−01	1.51E−01	1.43E−01	1.65E−01	1.52E−01	1.48E−01	1.60E−01	1.52E−01	1.50E−01	1.69E−01	1.78E−01	sARIMA	1.43E−01	LSTM	2.25E−01	1.63E−01	1.52E−01
SD. Mean Under-Accuracy Share	1.71E−01	2.03E−01	1.51E−01	1.55E−01	1.63E−01	1.60E−01	1.56E−01	1.47E−01	1.34E−01	1.65E−01	2.18E−01	Random Forest	1.34E−01	XGBoost	2.18E−01	1.66E−01	1.60E−01
Avg. Mean Over-Accuracy Share	1.04E+00	1.61E+00	5.12E−01	5.88E−01	3.35E−01	4.07E−01	6.29E−01	8.61E−01	7.82E−01	4.91E−01	5.34E−01	TBATS	3.35E−01	ETS	1.61E+00	7.08E−01	5.88E−01
SD. Mean Over-Accuracy Share	2.42E+00	7.08E+00	1.15E+00	1.78E+00	6.93E−01	9.41E−01	1.92E+00	1.68E+00	2.02E+00	1.28E+00	1.17E+00	TBATS	6.93E−01	ETS	7.08E+00	2.01E+00	1.68E+00
Avg. Normalized Time	1.37E+04	5.02E+02	1.25E+05	1.00E+00	7.61E+03	1.04E+01	1.01E+04	6.11E+02	1.58E+03	1.04E+03	6.54E+00	sNaive	1.00E+00	sARIMA	1.25E+05	1.46E+04	1.04E+03
SD. Normalized Time	3.19E+04	5.11E+02	5.10E+05	0.00E+00	9.86E+03	1.11E+01	9.54E+03	7.93E+02	4.28E+03	4.28E+03	4.51E+00	sNaive	0.00E+00	sARIMA	5.10E+05	5.19E+04	4.28E+03