Libra: A Benchmark for Time Series Forecasting Methods

André Bauer, Marwin Züfle, Simon Eismann, Johannes Grohmann, Nikolas Herbst, Samuel Kounev
University of Würzburg, Germany
firstname.lastname@uni-wuerzburg.de

ABSTRACT

In many areas of decision making, forecasting is an essential pillar. Consequently, there are many different forecasting methods. According to the "No-Free-Lunch Theorem", there is no single forecasting method that performs best for all time series. In other words, each method has its advantages and disadvantages depending on the specific use case. Therefore, the choice of the forecasting method remains a mandatory expert task. However, expert knowledge cannot be fully automated. To establish a level playing field for evaluating the performance of time series forecasting methods in a broad setting, we propose Libra, a forecasting benchmark that automatically evaluates and ranks forecasting methods based on their performance in a diverse set of evaluation scenarios. The benchmark comprises four different use cases, each covering 100 heterogeneous time series taken from different domains. The data set was assembled from publicly available time series and was designed to exhibit much higher diversity than existing forecasting competitions. Based on this benchmark, we perform a comprehensive evaluation to compare different existing time series forecasting methods.

CCS CONCEPTS

• General and reference → Evaluation; **Design**; **• Applied computing → Forecasting**.

KEYWORDS

Time Series Forecasting, Benchmarking, Evaluation

ACM Reference Format:
André Bauer, Marwin Züfle, Simon Eismann, Johannes Grohmann, Nikolas Herbst, Samuel Kounev. 2021. Libra: A Benchmark for Time Series Forecasting Methods. In *Proceedings of the 2021 ACM/SPEC International Conference on Performance Engineering (ICPE '21), April 19–23, 2021, Virtual Event, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3427921.3450241

1 INTRODUCTION

Time series forecasting is an essential pillar in many decision-making disciplines [18]. Accordingly, time series forecasting is also an established as active field of research and different methods have been proposed. Thus, the question arises if there is a single method that forecasts best for all time series. The "No-Free-Lunch Theorem" [35], initially formulated for optimization problems, denies the possibility of such a method. It states that improving one aspect typically leads to a degradation in performance for some other aspect. In other words, forecasting methods have their advantages and drawbacks depending on the considered time series.

Besides consulting an expert, the selection of the most appropriate approach can be guided by reviewing experimental results performed in either established benchmarks or scientific papers. By benchmark, we refer to an instrument used to evaluate and/or compare systems or methods based on certain properties [34]. Moreover, we consider benchmarks as a key instrument for improvement and competition. In the context of time series forecasting, the key concerns are the forecast accuracy and the time-to-result. Although there are forecasting competitions that can be considered as benchmarks, like the well-known M-Competitions[1], these are barely applied in scientific works [5]. More precisely, a recent survey found that publications on time series forecasting typically consider only a small set of (mostly related) methods and evaluate their performance on a small number of time series with only a few error measures while providing no information on the time-to-result of the different methods [5]. In other words, the quality of the evaluations suffers due to this methodology and, therefore, fails to guide the choice of an appropriate method for a particular use case.

To approach the problem of limited comparability between existing forecasting methods, we pose ourselves the following research questions:

- **RQ 1:** *How to automatically compare different forecasting methods on a level playing field?*
- **RQ 2:** *To what extent does the underlying data set differ from existing data sets?*

Towards addressing the research questions, we propose Libra—a forecasting benchmark—that automatically evaluates and ranks forecasting methods based on their performance in a diverse set of evaluation scenarios. The benchmark comprises four different use cases, each covering 100 heterogeneous time series taken from different domains. The data set was assembled from publicly available time series and exhibits much higher diversity than existing forecasting competitions.

The contributions of this paper comprise answering the two research questions addressed above and publishing the available source code of Libra on GitHub[2], as well as the acquired data set on Zenodo[3]. In addition, we provide a reproducible CodeOcean capsule[4] containing an example run for demonstration.

ICPE '21, April 19–23, 2021, Virtual Event, France

ACM ISBN 978-1-4503-8194-9/21/04…$15.00
https://doi.org/10.1145/3427921.3450241

[1] M-Competitions: https://mofc.unic.ac.cy/history-of-competitions/

[2] GitHub: https://github.com/DescartesResearch/ForecastBenchmark

[3] Zenodo: http://doi.org/10.5281/zenodo.4399959

[4] CodeOcean: https://doi.org/10.24433/CO.3240518.v1

The rest of the paper is structured as follows: We first introduce terms as well as definitions for understanding this paper and review related work in Section 2. Then, we face RQ 1 and present Libra in Section 3. In Section 4, we address RQ 2 and compare the proposed data set with existing forecasting competitions. In Section 5, we benchmark existing forecasting methods with Libra before concluding this paper.

2 BACKGROUND

In this section, we briefly present the basic terms and definitions related to time series and time series forecasting in Section 2.1 and Section 2.2, respectively. In Section 2.3, we outline how to asses the accuracy of a forecast. Finally, we present the state-of-the-art on forecasting competitions in Section 2.4.

2.1 Time Series

A *univariate time series* is an ordered collection of values of a quantity obtained over a specific period or since a certain point in time. In general, observations are recorded in successive and equidistant time steps (e.g., hours). Mathematically, if $y_t \in \mathbb{R}$ is the observed value at time t and T a discrete set of equidistant time points, a univariate time series is defined by

$$Y := \{y_t : t \in T\}\,. \tag{1}$$

Correlated or dependent observations can be stored together to form a *multivariate time series*. That is, there are multiple observed values for each point in time. Not to narrow the spectrum of methods and to cover classical frameworks, we focus on univariate time series.

A time series can also be seen as a composition of trend, seasonal, cycle, and irregular components [18]. The long-term development in a time series (i.e., upwards, downwards, or stagnate) is called *trend*. The presence of recurring patterns within a regular period in the time series is called *seasonality*, whereas the length of the seasonal pattern is called *frequency*. Rises and falls within a time series without a fixed frequency are called *cycles*. The remaining part of the time series that is not described by trend, seasonality, or cycles is called the *irregular component*. Note that there are time series where some components are absent.

2.2 Time Series Forecasting

Based on the underlying statistical properties of the time series, mathematical models can be developed for providing a plausible description of the observed data. These mathematical models are integrated into forecasting methods and can then be used to estimate the future development of the time series. Hence, a forecast of the time series can be conducted based on the historical data. More formally, we define the forecast for the time $n + k$ based on the historical observations $y_1, \ldots, y_n$ as

$$\hat{y}_{n+k|n} := f(y_n, \ldots, y_1, k) \tag{2}$$

with n and k being positive integers and f being the forecasting method capturing the time series model. While performing the forecast, it can be distinguished between *one-step-ahead* and *multi-step-ahead* forecasting. As the name indicates, when performing a one-step-ahead forecast, only the next value $\hat{y}_{n+1|n}$ is forecast, i.e., $k = 1$. In terms of a multi-step-ahead forecast, the values $\hat{y}_{n+1|n}, \ldots, \hat{y}_{n+h|n}$ are forecast, where h is the *forecast horizon* and represents the number of values that are forecast based on the historical data. In other words, the h values are forecast at once without updating the model with new data.

2.3 Assessing Forecast Accuracy

In principle, a forecast can either be evaluated *a-priori* or *a-posteriori*. In the case of an a-posteriori evaluation, the forecast's accuracy can only be quantified once the future values are available because the data is usually absent at the time of the forecast. In contrast, in an a-priori evaluation, the forecasting method is assessed before the actual forecast is carried out. For this reason, an estimator is required for the forecast accuracy. The straightforward solution is to use the error of the model fitting as an indicator for the forecast accuracy. However, this approach is unreliable, e.g., due to overfitting. Consequently, a more common practice is to split the time series into a *training set* and *test set* [18]. The training set is used to estimate the parameters of a forecasting method to fit the model to the data. Based on this model, a forecast is performed and then compared against the test set. Since the test data are not used for model fitting, this practice should provide a reliable indicator. Typically, the first 80% of a time series is used as the training set and the remaining 20% is used for evaluation, i.e., as the test set. According to R. Hyndman and G. Athanasopoulos [18], there are three types of error measures: (i) *Scale-dependent error measures*, (ii) *percentage error measures*, and (iii) *scaled error measures*. A detailed distinction between different error measures and corresponding discussions can be found in the works of M. Shcherbakov et al. [32], R. Adhikari and R. Agrawal [1], or R. Hyndman and A. Koehler [19].

2.4 Related Forecasting Competitions

In the last decades, several papers have been published in which forecasting methods have been evaluated either on a small or large scale. However, based on our former review [5], we found that most of the articles reviewed consider only a small set of (mostly related) methods and evaluate their performance on a small number of time series with only a few error measures while providing no information on the execution time of the different methods. To this end, we focus in this work on benchmarking forecasting methods in large scale. More precisely, we review forecasting competitions. Moreover, we restrict the scope on competitions involving multiple participants.

One of the best-known forecasting competition series is the Makridakis competitions or also known as M-Competitions. Before the first M-Competition was launched in 1982, S. Makridakis and M. Hibon [27] assembled 111 time series and compared different methods. Then, in the first M-Competition [25], S. Makridakis et al. compared forecasting methods on 1001 time series. In contrast to its predecessor, the second M-Competition [26] considered only 26 time series. However, this competition lasted almost four years, as participants, starting in 1987, received real-time data and feedback on their submitted forecasts as new data became available. The final forecast was then submitted in the last year. To extend and replicate the former M-Competitions, S. Makridakis and M. Hibon ran their third competition in 1998. The M3-Competition [28] contained 3003 time series. To show the potential of neural networks in terms of forecasting, S. Crone et al. [9] used 111 time series from the M3-Competition. A few years later, the Tourism competition was held in 2010 and contained 1311 time series [4]. To raise the importance of energy forecasting and have a sound benchmark, the Global Energy Forecasting Competition comprising 28 time series was first held in 2012 [15]. Two years later, the second Global Energy Forecasting Competition was held [16]. In contrast to the first edition, this competition comprised 15 time series while the data was updated monthly in a rolling manner. The last published competition is the M4-Competition [29]. S. Makridakis et al. provided 100,000 time series.

The existing data sets are designed for a specific use case, are very homogeneous, or are based on certain assumptions. For instance, most time series from the M3-Competition have a length of less than 100 data points, or the time series from M4-Competition have short frequencies. Consequently, both competitions cannot be used when evaluating forecasting methods for time series with high frequencies.

3 THE LIBRA FORECASTING BENCHMARK

In this section, we start with the design overview of Libra. Then, we introduce the time series data set in Section 3.2. Afterwards, we explain the evaluation types of the benchmark in Section 3.3. Finally, we highlight the applied performance measures.

3.1 Design Overview and Use Cases

To establish a level playing field for evaluating the performance of forecasting methods in a broad setting, we propose Libra, a forecasting benchmark that automatically evaluates and ranks forecasting methods based on their performance in a diverse set of evaluation scenarios. Figure 1 illustrates the workflow of Libra. First, the user implements the forecasting method. Then, the forecasting method is deployed within the benchmark. Afterward, the user specifies for which use case the deployed forecasting method should be evaluated. More precisely, the benchmark comprises four different use cases (see Section 3.2), each covering 100 heterogeneous time series taken from different domains. Moreover, the user has to select the evaluation type (see Section 3.3).

For each time series within the domain, the benchmark splits the time series into a training and test time series. The split depends on the evaluation type. Then, the training time series and the forecast horizon are passed to the forecasting method. Based on this input, the forecasting method performs a forecast and submits it to the benchmark. Based on the forecast and the test time series, the benchmark calculates different forecast error measures (see Section 3.4) and records the time-to-result of the forecasting method. After each time series is forecast, the benchmark creates a report that contains a detailed overview and ranking compared to the state-of-the-art methods. The overview shows the average and the standard deviation of the collected measures. The state-of-the-art methods in competition comprise ETS [20], GPyTorch [14], NNetar [17], random forest [7], sARIMA [6], sNaïve, SVR [11], TBATS [23], Theta [3], and XGBoost [8]. For details on the methods see Section 5.1. Note that the results (i.e., forecast error measures and normalized time-to-result) of these methods were conducted beforehand and saved within the benchmark.

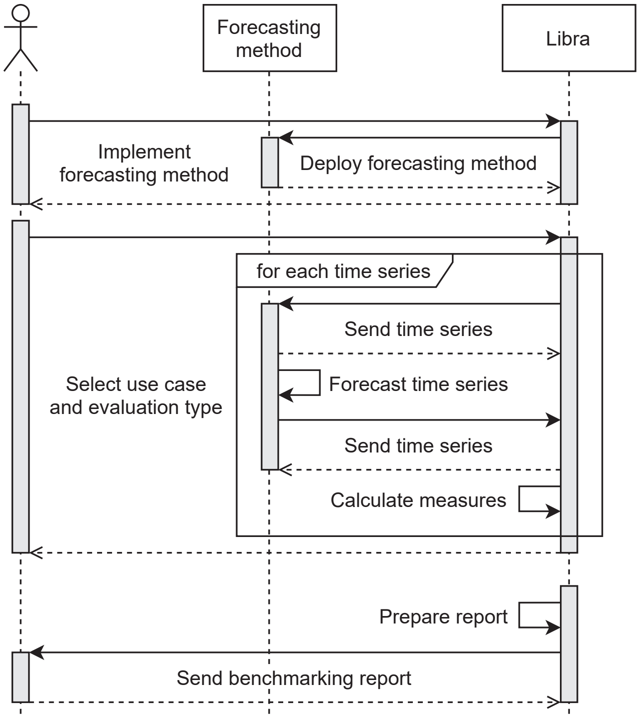

Figure 1: Sequence diagram for the usage of Libra.

3.2 Time Series Data Set

Numerous data sets are available online: The M-Competitions (such as M3[5] or M4[6]), the website Kaggle[7], R packages, and many more. Usually, the data sets are designed for a specific use case, are very homogeneous, or are based on certain assumptions. For instance, the M3-Competition contains 3,003 time series from different domains. However, most time series have a high degree of similarity and a length of less than 100 data points. Although, for instance, the M4-Competition comprises 100,000 time series, these time series are also quite similar and have low frequencies (1, 4, 12, and 24). A detailed analysis of the M-Competitions and other forecasting competitions can be found in Section 4.

In general, many domains have time series with more data points and/or higher frequencies than covered by the M-Competitions. For example, the popular FIFA'98 trace [2] has a frequency of 96, or in some self-aware systems, data are sampled each second leading to a frequency of 3600. That is, both competitions are not suitable for benchmarking forecasting methods for such domains. In other words, the currently available time series competitions are not exhaustive enough as they focus on only a few domains. Furthermore, it is hard to compare forecasting methods if they have been evaluated in different domains. Accordingly, a data set is required that

[5] M3 competition: https://forecasters.org/resources/time-series-data/m3-competition/
[6] M4 competition: https://www.mcompetitions.unic.ac.cy/the-dataset/
[7] Kaggle: https://www.kaggle.com/

covers a wide range of domains with different characteristics for generalizable forecasting results and enhancing the comparability between forecasting methods.

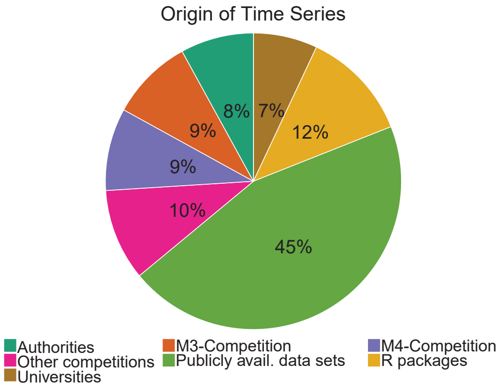

Figure 2: Time series origins used in the Libra benchmark.

Consequently, we assembled a highly diverse data set comprising 400 real-world time series[8] to enable a comprehensive and realistic evaluation. The time series are publicly available and originate from 50 different sources, including also time series from M3 and M4. Figure 2 shows the distribution of origins of the time series, which we combined into different groups for the sake of clarity. The groups are authorities from different countries, the M3-Competition, the M4-Competition, other competitions (e.g., Kaggle), various universities, R packages, and other publicly available data sets (not assignable to the other groups).

During the configuration of the benchmarking process, the user has to specify the use case and the evaluation type (see Section 3.1). More precisely, the user can choose between four different use cases, namely

(1) *Economics* (gas, sales, unemployment, . . .),
(2) *Finance* (stocks, sales prices, exchange rate, . . .),
(3) *Human access* (calls, SMS, Internet, . . .), and
(4) *Nature and demographics* (rain, birth, death, . . .).

Therefore, the data set is split into four domains, each covering 100 time series. The length and frequency distributions of the use cases are shown as cumulative distribution functions in Figure 3 and Figure 4. In addition, Figure 5 shows how the time series length and frequency are related in each use case. Note that the horizontal axes in the figures are depicted in log-scale and each x-axis shows different ranges for better readability.

3.3 Applied Evaluation Types

To quantify the forecast accuracy of a forecasting method, three different evaluation types are implemented in Libra: (i) *One-step-ahead forecasts*, (ii) *multi-step-ahead forecasts*, and (iii) *rolling origin forecasts*. For the first type, the forecasting method receives all values of the time series except the last one, which must be forecast. To evaluate forecast based on more values, the second type splits the time series in 80% training and 20% test. In other words, the forecast method has to forecast 20% of the time series at once. However, the first evaluation types perform an “arbitrary” split, resulting in forecasts that are sensitive to occurrences that may only occur in that particular split. To stabilize the assessment of forecasting methods, the third evaluation type is based on *rolling origin* [18]. The evaluation based on rolling origin is the time series equivalent of cross-validation from the field of machine learning. The term *origin* refers in this context to the training set of the time series, which is successively enlarged. In other words, this technique allows obtaining multiple forecasts, each on the increasing training set of a single time series. Typically, the origin is increased by 1, leading to many forecasts for long time series. Consequently, Libra offers a modified version of this rolling origin approach.

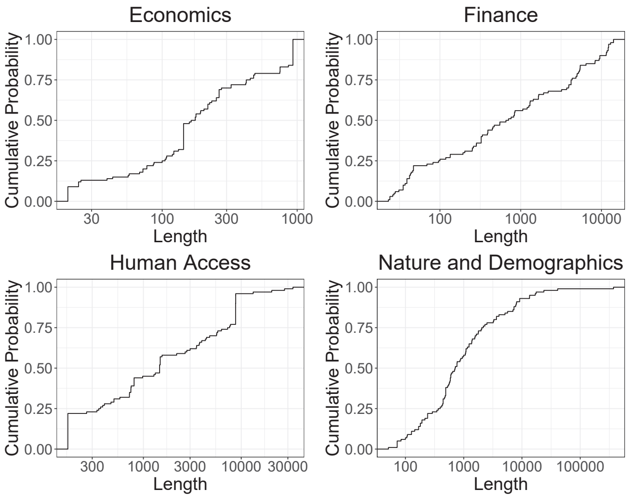

Figure 3: Distribution of the time series lengths in each use case.

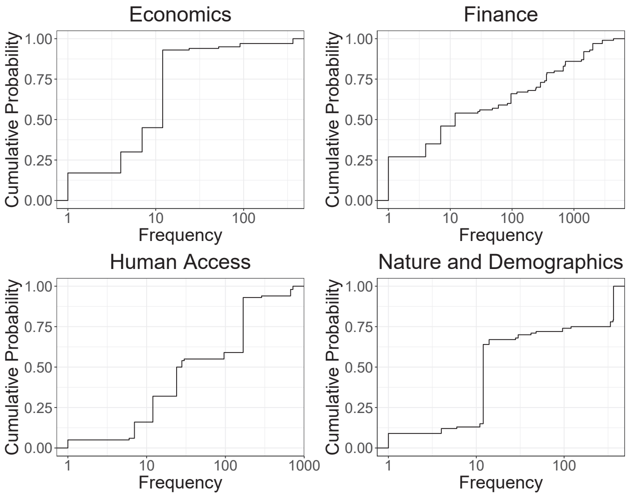

Figure 4: Distribution of the time series frequencies in each use case.

[8] The time series are available at Zenodo: http://doi.org/10.5281/zenodo.4399959.

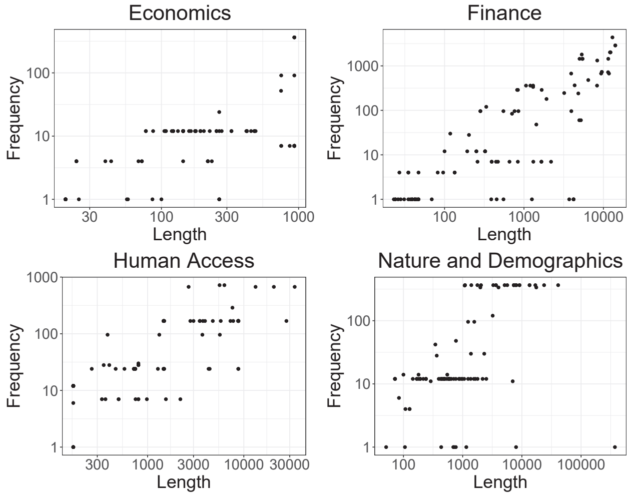

Figure 5: Relationship between time series length and frequency in each use case.

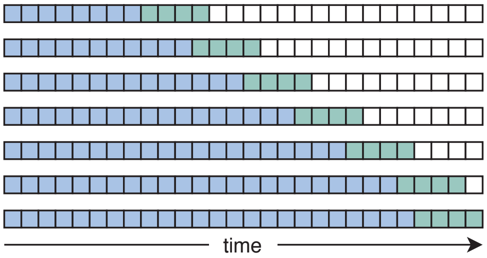

Figure 6: Concept of rolling origin forecast in Libra.

The core idea of the modified rolling origin is illustrated in Figure 6. The blue squares represent observations from the training set, the green squares observations from the test set, and the white squares the remaining observations from the time series. With every iteration except the last one, the training set is increased by a fixed number of observations. In the last iteration, the training set is enlarged so that both sets together cover the entire time series. More specifically, the starting point is set either to 40% of the time series or at two times the frequency of the time series plus 1, depending on which is greater. Note that the frequencies are already known. As the horizon is 20% of the time series length, the endpoint is set to 80% of the time series. Then, the range between the starting point and endpoint is divided into 100 parts of equal size. As the length of each split may not be an integer, the length is rounded up and the resulting integer is referred to as step. That is, the end indices of the origin (i.e., training data) begin with the starting point and are successively increased by the step to the endpoint. Therefore, the end indices contain $q + 1$ but a maximum of 101 points, where $step = \lceil r/100 \rceil$ and $q = \lceil r/step \rceil$. Mathematically, the end indices include the following points

$$\{y_{end}\} \cup \bigcup_{i=0,\dots,q} \{y_{start+i\cdot step}\}, \tag{3}$$

where y_t is the observation of the time series at time t, y_{start} the starting point, and y_{end} the endpoint. For each iteration, the forecasting method performs a multi-step-ahead forecast (i.e., the next 20% of the time series) based on the current training set (starting with the first observation of the time series and ends with the current end index). Finally, the calculated measures per forecast are averaged.

3.4 Applied Performance Measures

For the assessment of the forecasting method, our benchmark uses forecast error measures and the time-to-result of the forecasting method. For the benchmarking report, the average and standard deviation of each measure is output. Since the time depends on the underlying operating system and hardware, each time series is forecast by the deployed forecasting method and also by sNaïve[9]. Then, the time-to-result of the deployed method is normalized by the reference time of the sNaïve forecast to have comparable time-to-result measures. Note that the time-to-result for a time series reflects the duration in which the forecasting method receives the time series, estimates the parameters, creates the model, and performs the forecast. In general, it is impossible to prove the correctness of a measure; instead, it is a joint agreement on how to quantify the given property. To counter the weaknesses of a specific error measure, it is advantageous to take more than one of these measures into account when evaluating forecasts. Since different measures allow different insights and thus, a better understanding of the forecast can be obtained. Consequently, the forecasting benchmark implements six forecast error measures. Based on a recent survey [5] and in order to use common measures, the benchmark incorporates the *symmetrical mean absolute percentage error* [24] (sMAPE) e_{sM} and the *mean absolute scaled error* [19] (MASE) e_{MA}. Both measures are independent of the scale and can be used across different time series. In order to give useful insights into the forecasting method (e.g., tendency to under- or over-estimation the data), the benchmark also considers the *mean wrong-estimation shares* [5] ρ and the *mean wrong-accuracy shares* [5] δ. We choose those measures as all of them come with a deterministic mathematical expression, are simplistic as they can be described each in a compact sentence or formula, and are either unit-less (or have the unit of the time series) or a normalized ratio (percentage) with the values lying in the interval $[0; \infty)$, where 0 is the optimal value. Formally, the error measures can be calculated as

$$e_{\mathrm{sM}} := \frac{200\%}{k} \sum_{t=1}^{k} \frac{|y_t - \hat{y}_t|}{|y_t + \hat{y}_t|}, \tag{4}$$

$$e_{\mathrm{MA}} := \frac{\frac{1}{k}\sum_{t=1}^{k}|y_t - \hat{y}_t|}{\frac{1}{n-m}\sum_{i=m+1}^{n}|h_i - h_{i-m}|}, \tag{5}$$

$$\rho_U := \frac{1}{k} \cdot \sum_{t=1}^{k} max(sgn(y_t - \hat{y_t}), 0), \tag{6}$$

$$\rho_O := \frac{1}{k} \cdot \sum_{t=1}^{k} max(sgn(\hat{y_t} - y_t), 0), \tag{7}$$

[9] The sNaïve forecast repeats past observations so that each forecast value is equal to the corresponding observation from the last seasonal period.

$$\delta_U := \begin{cases} \frac{1}{k \cdot \rho_U} \cdot \sum_{t=1}^{k} \frac{max(y_t - \hat{y}_t, 0)}{|y_t|}, & \rho_U > 0, \\ 0, & otherwise, \end{cases} \tag{8}$$

$$\delta_O := \begin{cases} \frac{1}{k \cdot \rho_O} \cdot \sum_{t=1}^{k} \frac{max(\hat{y}_t - y_t, 0)}{|y_t|}, & \rho_O > 0, \\ 0, & otherwise, \end{cases} \tag{9}$$

where k is the forecast horizon, y_t the actual value at time t, $\hat{y}_t$ the forecast value at time t, m the length of the period ($m = 1$ for non-seasonal time series), n the length of the history, and h_i the historical values at time i.

4 COMPARISON WITH OTHER FORECASTING COMPETITIONS

In this section, we start with the description of the comparison of Libra with prominent forecasting competitions. Then, we compare Libras data set against the data sets from the forecasting competitions regarding their time series characteristics in Section 4.2. Finally, we investigate the similarity of the data sets.

4.1 Comparison Setup

During the last decades, some forecasting competitions have been established (e.g., the M-Competitions). To anwser RQ 2, i.e., showing that the assembled data set of Libra exhibits a high heterogeneity, we compare Libra set with prominent forecasting competitions. More precisely, we compare our assembled data set with publicly available competitions, namely M1 [25], M3 [28], M4 [29], NN3 [9], NN5[10], NNGC1[11], and Tourism [4]. To investigate a time series, one can either examine the observations or the time series characteristics describing the time series. The next two sections investigates the latter to compare our data sets. More precisely, we examine the distribution of the frequencies, the lengths, and 25 time series characteristics, which were proposed by R. Hyndman et al. [21], Y. Kang et al. [22], and B. Fulcher et al. [13].

4.2 Time Series Characteristics

First, we investigate the frequency distribution of the individual data sets shown in Table 1. The competitions M1, M3, M4, NN3, and Tourism comprise only time series with a low frequency. That is, the frequencies range between 1 to 24. More precisely, the M1, M3, and Tourism competitions support only the frequencies 1, 4, and 12, while the M4-Competition additionally supports the frequency of 24. The NN3 and NN5 competitions both support only a single frequency, while the NNGC1 competition supports only the frequencies 365 and 7,305. In contrast, our data set supports 37 different frequencies starting from 1 to 4,368.

Table 1: Frequency distribution within each data set.

Frequency	Libra	M1	M3	M4	NN3	NN5	NNGC1	Tourism
Min.	1	1	1	1	12	365	365	1
1st Qu.	7	4	1	1	12	365	365	1
Median	12	12	4	4	12	365	7,305	4
3rd Qu.	168	12	12	12	12	365	7,305	12
Max.	4,368	12	12	24	12	365	7,305	12

Next, we investigate the length distribution as shown in Table 2. For instance, the time series from the NN5 competition all have the same length, with 791 observations. The median length of the time series within the M1, M3, NN3, and Tourism competitions is less than or equal to 134. In contrast, the median length of our time series is 570. While examining the interquartile range, our data set has a range of about 3,000 observations, while the other data sets have less than 300 observations. Furthermore, our data set also contains the longest time series with 372,864 observations. In comparison, the longest time series from the M4-Competition, which has the longest time series among the other competitions, comprises only 9,993 observations. To sum up, our data set shows the highest diversity in terms of the time series' lengths.

Table 2: Length distribution within each data set.

Length	Libra	M1	M3	M4	NN3	NN5	NNGC1	Tourism
Min.	20	15	20	19	68	791	502	11
1st Qu.	169	44	44	56	69	791	747	27
Median	570	68	69	106	134	791	902	110
3rd Qu.	2,074	85	133	252	144	791	1,026	199
Max.	372,864	150	144	9,933	144	791	1,742	333

Besides the lengths and frequencies distribution, we investigate time series characteristics (e.g., C1: spectral entropy of the time series [30], C10: first auto-correlation coefficient of the time series, or C16: non-linearity of the time series [33]) proposed by different scientific works [13, 21, 22]. The description of the time series characteristics and the calculated characteristics of our data set as well as each forecasting competition can be found at Zenodo[12]. For the investigation, we apply min-max scaling to all time series characteristics, taking all time series of all considered data sets into account. This means that for each time series characteristic, the minimum (0) and the maximum (1) can be located in different data sets. The resulting distribution of each data set is illustrated as a spider-chart in Figure 7. Each chart contains 25 edges, each representing a time series characteristic. For each time series characteristic, the red dot represents the largest value in the data set, the green dot the average value, and the blue dot the smallest value. Based on these charts, our data set exhibits a higher diversity of time series characteristics than all other competitions except the M4-Competition. Our data set has the minimum value for 9 out of 25 time series characteristics and the maximum value for 10 out of 25 time series characteristics. The M4-Competition covers the remaining minima and maxima. However, the M4-Competition comprises 100,000 time series. Therefore the likelihood of having a time series with a minimum/maximum for a time series characteristic is higher than in our data set.

4.3 Distance between Time Series

To determine how similar time series are to each other in the individual data sets, we calculate the distance between them. If we think of the geometric representation of a time series, we could, for example, consider the Euclidean distance or dynamic time warping. However, the first metric can only be calculated if the time series

[10] NN5 competition: http://www.neural-forecasting-competition.com/NN5/

[11] NNGC1 competition: http://www.neural-forecasting-competition.com/

[12] Time series characteristics at Zenodo: http://doi.org/10.5281/zenodo.4115345

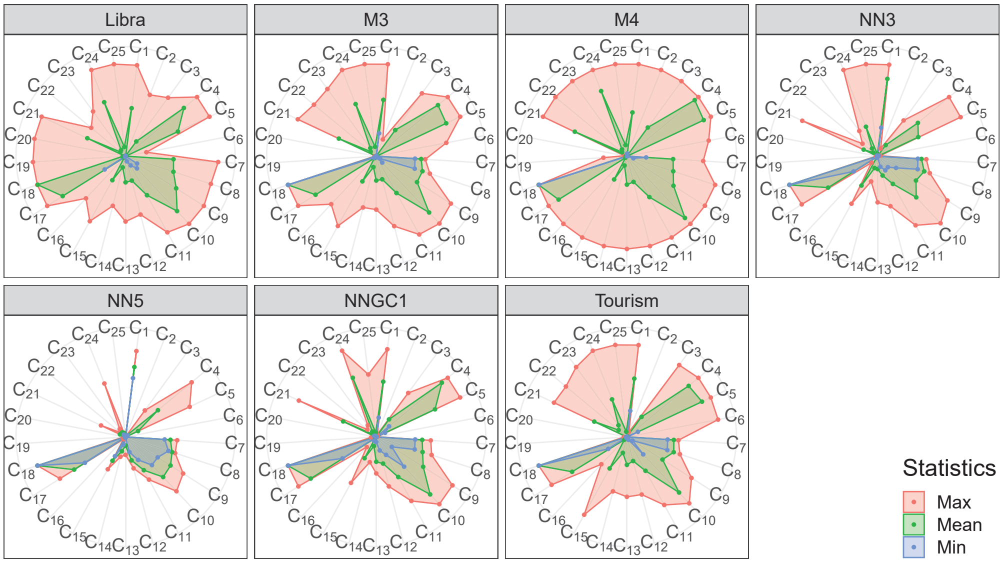

Figure 7: Distribution of time series characteristics per investigated data set.

are of equal length and the latter is difficult to interpret. To this end, we adopt the idea of the zoomed ranking [10] approach applied to time series. More precisely, the distance between two time series is equal to the L_1-norm of their describing time series characteristics. Mathematically, the distance between two time series Y_i and Y_j is defined as [10]

$$d(Y_i, Y_j) := \sum_{m=1}^{q} \frac{|c_{Y_i,m} - c_{Y_j,m}|}{\max\limits_{k=\{1,\dots,s\}/\{i\}} c_{Y_k,m} - \min\limits_{k=\{1,\dots,s\}/\{i\}} c_{Y_k,m}}, \qquad (10)$$

where $Y_1, \dots, Y_s$ are all time series from all considered data sets with s being a positive integer, Y_i and Y_j origin from the same data set, and $c_{Y_i,1}, \dots, c_{Y_i,q}$ are the describing time series characteristics of time series Y_i with q being a positive integer, i.e., 25. The distance between two time series lies in the interval $[0, \infty)$, where 0 indicates that the time series are equal in terms of their describing characteristics. Consequently, the higher the distance, the more heterogeneous are both time series.

Table 3: Distance between time series within each data set.

Distance	Libra	M1	M3	M4	NN3	NN5	NNGC1	Tourism
Min.	0.003	0.000	0.000	0.000	0.237	0.000	0.018	0.033
1st Qu.	2.697	2.125	2.001	1.669	1.632	0.681	2.077	2.072
Median	3.866	2.950	2.968	2.473	2.797	0.992	2.867	2.899
Mean	4.104	3.186	3.304	2.808	3.112	1.087	2.900	3.024
3rd Qu.	5.296	4.027	4.331	3.594	4.291	1.405	3.660	3.846
Max.	27.898	10.785	10.876	13.583	8.758	3.235	8.395	9.092

To analyze the different data sets, we calculate the distance between each pair of time series within the data set and report the average distance as well as the respective distribution in Table 3. The time series in our data set have an average distance of 4.104, while the time series in the other data have an average distance between 1.087 and 3.304. Moreover, the M-Competitions and the NN5 competitions have time series that are identical regarding their describing time series characteristics. Except for the minimal value, our data set exhibits the highest average time series distance for all other quartiles. As for the maximum value, our data set's average distance is at least twice as far as in all other competitions. In summary, our data set exhibits the higher average distance and thus the greater heterogeneity between time series.

5 BENCHMARKING FORECASTING METHODS

In this section, we benchmark existing forecasting methods with Libra. We first introduce the benchmarking setup. Then, we benchmark the methods on the different use cases in Section 5.2 – 5.5. Afterward, we report the overall performance of the methods in Section 5.6. Finally, we sum the results and discuss threats to validty.

5.1 Benchmarking Setup

For having a broad and representative forecasting method competition stored in Libra, we compare different methods from different fields. The competitors can be grouped into (i) "classical" time series forecasting methods and (ii) regression-based machine learning methods. For each category, we consider a set of representative methods. The classical forecasting methods are listed and briefly described below:

- *ETS* [20] builds on the concept of exponential smoothing and is a framework that automatically retrieves the components (trend, season, and error) of a time series and determines the relationships (additive, multiplicative, or not present) between the components.
- *sARIMA*[13] [6] extends the ARIMA model by adding a seasonal counterpart to each component (autoregressive model

[13] In the experiments, we use auto.arima [18] to find the most suitable model for the time series automatically.

for the past values, moving average for the past forecast errors, and time series differencing for stationarity).

- *sNaïve* repeats past observations for the entire forecast horizon. More precisely, each forecast value is equal to the corresponding observation from the last period.
- *TBATS* [23] extends ETS using a trigonometric representation based on Fourier series for the seasonal part of the time series and an autoregressive-moving average model for the error corrections. Further, the time series is transformed with Box-Cox transformations.
- *Theta* [3] first checks whether the time series is seasonal, and if so, de-seasonalizes the time series. The de-seasonalized time series is then split into a short- and a long-term component. Finally, the forecast is the weighted forecast of both components.

The competing methods from the field of machine learning are outlined in the following:

- *GPyTorch* [14] is a Gaussian process library. The basic idea for modeling the time series is to apply Gaussian process inference based on black-box matrix-matrix multiplication.
- *NNetar* [17] is a feed-forward neural network with one hidden layer. The model is trained with lagged values of the time series, while the number of lags and the number of nodes in the hidden layer are automatically selected.
- *Random forest* [7] is an ensemble of decision trees based on bagging, i.e., the trees are generated in parallel, fully grown, and independent of each other. To reduce overfitting, each tree is trained on a random sample of the features. In the following, we refer to this method as *RF*.
- *SVR* [11] is based on the same principle as SMVs, i.e., finding separation lines to group the data into different classes, with the extension to predict numerical values.
- *XGBoost* [8] is an ensemble of decision trees based on gradient tree boosting, i.e., the trees are growing sequentially with knowledge from their preceding tree. To reduce overfitting, XGBoost applies regularization objects, shrinkage, and feature subsampling.

Each method received a single time series as input if not stated otherwise. Note that using one time series at a time is a major difference to the M4-Competition where it is possible to use the complete training data set (i.e., all time series) for training the algorithms. In our experiments, each time series was divided into history and test, where the split depends on the evaluation type. Based on the history, each method learned a model that was used for forecasting future values of the time series (i.e., the test part) at once with a single execution. This forecasting procedure (i.e., receiving the time series, estimating the parameters, building the model, and forecasting the time series) was repeated ten times for each time series as some methods are not deterministic. Consequently, the reported measures were determined on the average values of each time series.

As input for the forecasting task, the classical time series forecasting methods and NNetar received the time series and the respective frequency. The regression-based machine learning methods except NNetar got the time series and a synthetic seasonal pattern (a vector with the indices modulus the frequency) as input. To achieve a reliable forecast for GPyTorch, we shifted each time series linearly to obtain a mean value of zero [31]. Note that we used all methods "out-of-the-box" since the results of the M3-Competition have shown that the methods were kept simple and, on average, complex models do not necessarily perform better [28]. That is, there was no parameter tuning and the methods were used with their default settings. Recall the "No-Free-Lunch Theorem" [35], stating that improving a method for one aspect leads to deterioration of another aspect.

To compare and quantify the performance of the different forecasting methods, we use forecast error measures (see Section 3.4) and the time-to-result. The time-to-result for a time series reflects the duration in which the forecasting method receives the time series, estimates the parameters, creates the model, and performs the forecast. Note that the time-to-result is normalized to be independent of the underlying hardware. Table 4 lists a brief description of each measure.

Table 4: Overview of the applied measures.

Name	Description
e_{sM}	Forecast accuracy based on the sMAPE. The measures $\overline{e}_{\text{sM}}$, and $\sigma_{e_{\text{sM}}}$ reflect the average error and the standard deviation of the error.
e_{MA}	Forecast accuracy based on the MASE. The measures $\overline{e}_{\text{MA}}$ and $\sigma_{e_{\text{MA}}}$ reflect the average error and the standard deviation of the error.
ρ_U, ρ_O	Mean wrong-estimation share as the relative number of forecast values that under- or overestimate the actual values.
δ_U, δ_O	Mean wrong-accuracy share as the mean percentage error during under- or overestimating the actual values.
t_{sN}	The time-to-result normalized by the time required by sNaïve. The measures $\overline{t}_{sN}$ and $\sigma_{t_{sN}}$ reflect the average time and the standard deviation of the time.

5.2 Economics Use Case

Table 5 and 6 show the results of the multi-step-ahead forecasting competition on the economic use case for the classical time series forecasting methods and the regression-based machine learning methods, respectively. Each row shows a measure and the columns correspond to the methods in competition. The best values (the lower, the better) are highlighted in bold. The most accurate forecasting method based on $\overline{e}_{\text{sM}}$ is ETS (13.02%) followed by sNaïve (13.28%). The most accurate machine learning method is NNetar (16.49%) followed by GPyTorch (17.32%). Concerning $\overline{e}_{\text{MA}}$, the most accurate forecasting method is sARIMA (0.53) followed by TBATS (0.59) and the most accurate machine learning method is XGBoost (0.87) followed by GPyTorch (0.91). For both error measures, all forecasting methods (except sNaïve for $\overline{e}_{\text{MA}}$) exhibit a higher accuracy than the most accurate machine learning method.

As there are different superior methods for both error measures, we investigate measures describing the forecast. More precisely, ρ_O or ρ_U either reflects whether the forecasting method over- or underestimates the future vales. The methods sARIMA, TBATS, and

Table 5: Comparison of classical time series forecasting methods on the economics use case.

Measures	ETS	sARIMA	sNaïve	TBATS	Theta
$\overline{e}_{\text{sM}}$ [%]	**13.02**	14.88	13.28	14.19	13.40
$\sigma_{e_{\text{sM}}}$ [%]	16.26	25.80	**12.12**	21.06	21.42
$\overline{e}_{\text{MA}}$	0.66	**0.53**	0.90	0.59	0.77
$\sigma_{e_{\text{MA}}}$	1.41	**0.97**	2.20	0.99	1.85
ρ_U [%]	55.52	49.33	67.86	49.71	60.07
ρ_O [%]	44.48	50.67	32.08	50.29	39.93
δ_U [%]	**7.44**	7.45	10.18	9.51	9.05
δ_O [%]	$1.58 \cdot 10^2$	94.40	**90.43**	$1.06 \cdot 10^2$	$2.34 \cdot 10^2$
$\overline{t}_{sN}$	$3.40 \cdot 10^2$	$7.29 \cdot 10^3$	**1.00**	$2.63 \cdot 10^3$	6.04
$\sigma_{t_{sN}}$	$2.55 \cdot 10^2$	$2.36 \cdot 10^4$	**0.00**	$1.86 \cdot 10^3$	18.15

Table 6: Comparison of regression-based machine learning methods on the economics use case.

Measures	GPyTorch	NNetar	RF	SVR	XGBoost
$\overline{e}_{\text{sM}}$ [%]	17.32	**16.49**	19.80	25.08	17.58
$\sigma_{e_{\text{sM}}}$ [%]	**13.33**	16.86	18.27	29.23	19.89
$\overline{e}_{\text{MA}}$	1.04	0.91	1.69	2.20	**0.87**
$\sigma_{e_{\text{MA}}}$	2.06	**2.0**	5.71	8.14	2.16
ρ_U [%]	71.80	52.92	73.72	72.57	70.85
ρ_O [%]	28.20	47.08	26.28	27.43	29.15
δ_U [%]	12.11	**7.85**	13.52	16.65	26.04
δ_O [%]	**1.05**$\cdot 10^2$	$2.84 \cdot 10^2$	$1.51 \cdot 10^2$	$1.39 \cdot 10^2$	$2.01 \cdot 10^2$
$\overline{t}_{sN}$	$3.39 \cdot 10^3$	$1.18 \cdot 10^2$	62.26	13.09	**6.41**
$\sigma_{t_{sN}}$	$3.00 \cdot 10^3$	$1.19 \cdot 10^2$	77.32	39.08	**16.90**

NNetar exhibit almost no tendency ($\rho_O \sim \rho_U$), while the remaining methods tend to underestimate (ρ_U > 50%). Especially, the methods sNaïve, GPyTorch, RF, SVR, and XGBoost underestimate a time series on average more than 2/3 of the forecast horizon. However, while underestimating a time series, all methods are more accurate than overestimating a time series.

In terms of the time-to-result, the forecasting methods sNaïve (1.00) and Theta (6.04) are by far the fastest methods. Note that the time-to-result of sNaïve was used to normalize the recorded times and thus, sNaïve has a value of 1.00. The remaining forecasting methods are between 100 and 1,000 times slower than sNaïve. For instance, sARIMA, which is the most accurate method concerning $\overline{e}_{\text{MA}}$, is the slowest method and also shows by far the most remarkable variation for the time-to-result. The fastest machine learning method is XGBoost (6.41) followed by SVR (13.09).

As Libra offers three different evaluation types, we compare $\overline{e}_{\text{sM}}$ and $\sigma_{e_{\text{sM}}}$ for all evaluation types. For the sake of brevity, Table 7 comprises only ETS, sARIMA, GPyTorch, and NNetar as they are the most accurate methods of their field according to the multi-step-ahead-based $\overline{e}_{\text{sM}}$. Each row shows a measure and each column one forecasting method. As the rolling origin performs several multi-step-ahead forecasts to stabilize the forecast error and to avoid arbitrary splits, $\overline{e}_{\text{sM}}$ for multi-step-ahead and rolling origin are almost similar. In contrast, the one-step-ahead forecast produces completely different results. For instance, sARIMA, which is the second most accurate method based on the multi-step-ahead forecast, exhibits a high $\sigma_{e_{\text{sM}}}$ ($2.13 \cdot 10^3$%) and thus has the highest $\overline{e}_{\text{sM}}$ (227.01%). In other words, while having reliable forecasts on long forecast horizons, sARIMA is prone to error on short forecast horizons. In the following sections, we focus on multi-step-ahead forecasts.

Table 7: Comparison of the most accurate forecasting methods on the economics use case considering all evaluation types.

Type	Measures	ETS	sARIMA	GPyTorch	NNetar
One-step	$\overline{e}_{\text{sM}}$ [%]	9.37	227.01	13.03	25.45
	$\sigma_{e_{\text{sM}}}$ [%]	22.59	$2.13 \cdot 10^3$	25.91	49.73
Multi-step	$\overline{e}_{\text{sM}}$ [%]	13.02	14.88	17.32	16.49
	$\sigma_{e_{\text{sM}}}$ [%]	16.26	25.80	13.33	16.86
Rolling origin	$\overline{e}_{\text{sM}}$ [%]	12.14	16.35	16.82	17.25
	$\sigma_{e_{\text{sM}}}$ [%]	11.27	30.49	11.76	15.73

5.3 Finance Use Case

The results of the multi-step-ahead forecasting competition on the finance use case are listed in Table 8 showing the classical time series forecasting methods and in Table 9 showing the regression-based machine learning methods. Each row shows a measure and each column a method. The most accurate values (the lower, the better) are highlighted in bold. Among the forecasting methods, ETS (17.59%) is based on $\overline{e}_{\text{sM}}$ the most accurate method followed by sARIMA (17.83%). The most accurate machine learning method is XGBoost (19.25%) followed by RF (23.39%). In contrast to all other methods, NNetar ($1.33 \cdot 10^3$) is by far the most inaccurate method and exhibits a standard deviation with a value of $4.14 \cdot 10^4$. However, these high values are introduced by a single time series. According to $\overline{e}_{\text{MA}}$, sARIMA (1.58) is the most accurate forecasting method followed by ETS (1.88). Again, XGBoost is the most accurate machine learning method (2.13) followed by GPyTorch (2.36). Like the economics use case, all forecasting methods are more accurate than the machine learning methods.

Table 8: Comparison of classical time series forecasting methods on the finance use case.

Measures	ETS	sARIMA	sNaïve	TBATS	Theta
$\overline{e}_{\text{sM}}$ [%]	**17.59**	17.83	24.40	17.95	18.00
$\sigma_{e_{\text{sM}}}$ [%]	17.96	22.04	24.31	**17.44**	18.14
$\overline{e}_{\text{MA}}$	1.88	**1.58**	2.25	1.94	1.96
$\sigma_{e_{\text{MA}}}$	5.32	**4.07**	5.89	5.29	5.24
ρ_U [%]	66.75	64.83	69.88	67.96	70.28
ρ_O [%]	33.25	35.17	30.10	32.04	29.72
δ_U [%]	12.40	**12.24**	17.22	12.71	13.07
δ_O [%]	18.56	25.32	19.79	**15.48**	20.07
$\overline{t}_{sN}$	$1.69 \cdot 10^2$	$2.35 \cdot 10^6$	**1.00**	$5.35 \cdot 10^3$	8.58
$\sigma_{t_{sN}}$	$2.09 \cdot 10^2$	$1.75 \cdot 10^7$	**0.00**	$6.50 \cdot 10^3$	16.72

In contrast to the economics use case, all methods tend heavily to underestimate a time series. More precisely, the forecast of each

Table 9: Comparison of regression-based machine learning methods on the finance use case.

Measures	GPyTorch	NNetar	RF	SVR	XGBoost
$\overline{e}_{\text{sM}}$ [%]	31.64	$1.33{\cdot}10^3$	24.94	30.44	**19.25**
$\sigma_{e_{\text{sM}}}$ [%]	32.65	$4.14{\cdot}10^4$	23.39	30.69	**19.46**
$\overline{e}_{\text{MA}}$	2.36	2.39	3.62	4.80	**2.13**
$\sigma_{e_{\text{MA}}}$	**5.80**	6.89	11.64	16.59	5.92
ρ_U [%]	70.48	69.85	70.78	69.99	70.66
ρ_O [%]	29.01	30.15	29.22	30.01	29.34
δ_U [%]	20.37	14.97	17.33	20.42	**13.99**
δ_O [%]	17.89	29.24	17.21	**15.50**	22.06
$\overline{t}_{sN}$	$9.95{\cdot}10^3$	$6.78{\cdot}10^2$	$1.64{\cdot}10^3$	$3.60{\cdot}10^2$	**7.27**
$\sigma_{t_{sN}}$	$1.16{\cdot}10^4$	$8.47{\cdot}10^2$	$3.16{\cdot}10^3$	$6.23{\cdot}10^2$	**17.24**

method is, on average, at least 64% of the horizon below the actual values. Moreover, all methods are exhibiting almost the same accuracy during under- and overestimating. In terms of $\overline{t}_{sN}$, sNaïve (1.00) and Theta (8.58) are the fastest forecasting methods and XGBoost (7.27) and SVR ($3.60{\cdot}10^2$) are the fastest machine learning methods. Also, in this use case, sARIMA, which is the most accurate method according to $\overline{e}_{\text{MA}}$, is, on average, more than one million times slower than sNaïve.

5.4 Human Access Use Case

The results for the multi-step-ahead forecasting competition on the human access use case are shown in Table 10 (classical time series forecasting methods) and Table 11 (regression-based machine learning methods). Each column shows a method and each row a measure, where the most accurate values (the lower, the better) are highlighted in bold. The most accurate forecasting method based on $\overline{e}_{\text{sM}}$ is sNaïve (23.60%) followed by sARIMA (27.95%). XGBoost (28.45%) and GPyTorch (29.92%) are the most accurate machine learning methods. In terms of $\overline{e}_{\text{MA}}$, XGBoost (0.55) and GPyTorch (0.65) are the most accurate machine learning methods, while sARIMA (0.50) and sNaïve (0.51) swap places. In contrast to the economic and finance use cases, the forecasting methods do not outperform the machine learning methods concerning both accuracy measures.

Table 10: Comparison of classical time series forecasting methods on the human access use case.

Measures	ETS	sARIMA	sNaïve	TBATS	Theta
$\overline{e}_{\text{sM}}$ [%]	46.64	27.95	**23.60**	42.51	31.20
$\sigma_{e_{\text{sM}}}$ [%]	65.27	54.99	**34.91**	$1.46{\cdot}10^2$	89.65
$\overline{e}_{\text{MA}}$	1.42	**0.50**	0.51	0.63	0.65
$\sigma_{e_{\text{MA}}}$	2.86	**0.54**	0.89	1.03	1.58
ρ_U [%]	45.68	48.73	53.61	49.40	50.03
ρ_O [%]	54.32	51.27	45.56	50.60	49.97
δ_U [%]	$2.46{\cdot}10^2$	52.79	**17.63**	23.55	21.02
δ_O [%]	$7.73{\cdot}10^3$	$2.26{\cdot}10^3$	$2.02{\cdot}10^3$	**1.24**$\cdot 10^3$	$6.31{\cdot}10^3$
$\overline{t}_{sN}$	$6.90{\cdot}10^2$	$9.05{\cdot}10^5$	**1.00**	$1.48{\cdot}10^4$	15.33
$\sigma_{t_{sN}}$	$1.11{\cdot}10^3$	$4.58{\cdot}10^6$	**0.00**	$1.52{\cdot}10^4$	25.44

Table 11: Comparison of regression-based machine learning methods on the human access use case.

Measures	GPyTorch	NNetar	RF	SVR	XGBoost
$\overline{e}_{\text{sM}}$ [%]	29.92	33.09	75.60	$1.47{\cdot}10^2$	**28.45**
$\sigma_{e_{\text{sM}}}$ [%]	50.22	67.99	$6.17{\cdot}10^2$	$1.08{\cdot}10^3$	**42.73**
$\overline{e}_{\text{MA}}$	0.65	0.66	0.82	1.01	**0.55**
$\sigma_{e_{\text{MA}}}$	1.02	0.91	1.44	1.83	**0.87**
ρ_U [%]	52.69	48.58	43.79	47.38	52.35
ρ_O [%]	47.13	51.42	56.21	52.62	47.65
δ_U [%]	19.85	24.81	**18.32**	23.03	68.87
δ_O [%]	**1.95**$\cdot 10^3$	$2.59{\cdot}10^3$	$3.82{\cdot}10^3$	$3.59{\cdot}10^3$	$2.58{\cdot}10^3$
$\overline{t}_{sN}$	$1.41{\cdot}10^4$	$9.60{\cdot}10^2$	$2.40{\cdot}10^3$	$1.06{\cdot}10^3$	**6.72**
$\sigma_{t_{sN}}$	$1.11{\cdot}10^4$	$1.01{\cdot}10^3$	$4.90{\cdot}10^3$	$2.20{\cdot}10^3$	**13.30**

Regarding the under- and overestimating, all methods behave differently than in the economic and finance use case. More precisely, all methods show almost no tendency to under- or overestimating the future values (i.e., $\rho_O \sim \rho_U$). However, the methods are, on average, far more accurate during underestimation than overestimation. Regarding $\overline{t}_{sN}$, the fastest forecasting method is sNaïve (1.00) followed by Theta (15.33) and the fastest machine learning method is XGBoost (6.72) followed by NNetar ($9.60{\cdot}10^2$). Also, in this use case, sARIMA is the most accurate method according $\overline{e}_{\text{MA}}$, but exhibits the highest mean value ($9.05{\cdot}10^5$) and the highest standard deviation ($4.58{\cdot}10^6$) for the time-to-result.

5.5 Nature and Demographics Use Case

The multi-step-ahead forecasting results of the last use case, nature and demographics, are shown in Table 12 for the classical time series forecasting methods and in Table 13 for the regression-based machine learning methods. Each row reflects a measure and each column a method. The most accurate values (the lower, the better) are highlighted in bold. The most accurate forecasting method based on $\overline{e}_{\text{sM}}$ is TBATS (19.85%) followed by Theta (21.79%), while the most accurate machine learning method is GPyTorch (24.00%) followed by RF (26.53%). Among the forecasting methods, sARIMA (0.31) and TBATS (0.36) are the most accurate methods regarding $\overline{e}_{\text{MA}}$. XGBoost (0.47) and GPyTorch (0.48) are the most accurate machine learning methods. Similar to the human access use case, the forecasting methods and machine learning methods exhibit a comparable accuracy for both measures.

As in the human access use case, all methods neither tend to under- nor overestimate the future vales (i.e., $\rho_U \sim \rho_O$). However, while underestimating a time series, all methods are more accurate than overestimating a time series. On average, the fastest forecasting methods are sNaïve (1.00) and Theta (1.04) and the fastest machine learning methods are XGBoost (6.54) and SVR ($1.04{\cdot}10^2$). The remaining methods are at least 100 times slower than sNaïve. For example, sARIMA is 125,000 times slower than sNaïve.

5.6 Overall Evaluation

In this section, we investigate the overall multi-step-ahead performance of the forecasting methods. Table 14 shows the performance of the classical time series forecasting methods on all use cases and

Table 12: Comparison of classical time series forecasting methods on the nature and demographics use case.

Measures	ETS	sARIMA	sNaïve	TBATS	Theta
$\overline{e}_{\mathrm{sM}}$ [%]	38.54	21.87	26.00	**19.85**	21.79
$\sigma_{e_{\mathrm{sM}}}$ [%]	$1.05\cdot10^{2}$	28.47	39.54	**22.52**	24.41
$\overline{e}_{\mathrm{MA}}$	0.83	**0.31**	0.42	0.36	0.40
$\sigma_{e_{\mathrm{MA}}}$	2.75	**0.28**	0.64	0.53	0.58
ρ_U [%]	44.95	46.18	53.38	51.01	50.39
ρ_O [%]	55.05	53.82	46.39	48.99	49.61
δ_U [%]	15.12	**14.34**	16.51	15.21	14.80
δ_O [%]	$1.60\cdot10^{2}$	51.15	58.83	**33.51**	40.68
$\overline{t}_{sN}$	$5.02\cdot10^{2}$	$1.25\cdot10^{5}$	**1.00**	$7.61\cdot10^{3}$	1.04
$\sigma_{t_{sN}}$	$5.17\cdot10^{2}$	$5.11\cdot10^{5}$	**0.00**	$1.03\cdot10^{4}$	19.31

Table 13: Comparison of regression-based machine learning methods on the nature and demographics use case.

Measures	GPyTorch	NNetar	RF	SVR	XGBoost
$\overline{e}_{\mathrm{sM}}$ [%]	**24.00**	28.79	26.53	31.79	30.11
$\sigma_{e_{\mathrm{sM}}}$ [%]	33.53	45.96	**27.13**	69.59	45.73
$\overline{e}_{\mathrm{MA}}$	0.48	0.52	0.57	0.54	**0.47**
$\sigma_{e_{\mathrm{MA}}}$	0.81	0.75	0.98	1.27	**0.74**
ρ_U [%]	51.83	42.75	48.00	50.82	48.79
ρ_O [%]	48.17	57.25	52.00	49.18	51.21
δ_U [%]	16.02	15.23	**15.04**	16.92	17.84
δ_O [%]	62.92	86.13	78.17	**49.07**	53.36
$\overline{t}_{sN}$	$1.01\cdot10^{4}$	$6.11\cdot10^{2}$	$1.58\cdot10^{3}$	$1.04\cdot10^{2}$	**6.54**
$\sigma_{t_{sN}}$	$1.02\cdot10^{4}$	$8.39\cdot10^{2}$	$4.72\cdot10^{3}$	$4.50\cdot10^{3}$	**14.60**

Table 15 the results of the regression-based machine learning methods on all use cases. Each row shows a measure and each column a method. The best values (the lower, the better) are highlighted in bold. Considering all use cases, the most accurate forecasting method based on both accuracy measures is sARIMA (20.63%; 0.73). For the machine learning methods, XGBoost (23.85%; 1.00) is the most accurate method regarding both accuracy measures.

Table 14: Comparison of classical time series forecasting methods on all use cases.

Measures	ETS	sARIMA	sNaïve	TBATS	Theta
$\overline{e}_{\mathrm{sM}}$ [%]	28.95	**20.63**	21.82	23.62	21.10
$\sigma_{e_{\mathrm{sM}}}$ [%]	64.57	35.63	**30.07**	76.00	48.95
$\overline{e}_{\mathrm{MA}}$	1.20	**0.73**	1.02	0.88	0.94
$\sigma_{e_{\mathrm{MA}}}$	3.43	**2.17**	3.27	2.82	2.97
ρ_U [%]	53.22	52.27	61.18	54.52	57.69
ρ_O [%]	46.78	47.73	38.53	45.48	42.31
δ_U [%]	70.44	21.71	15.38	15.25	**14.48**
δ_O [%]	$2.02\cdot10^{3}$	$6.07\cdot10^{2}$	$5.46\cdot10^{2}$	**3.48**$\cdot10^{2}$	$1.65\cdot10^{3}$
$\overline{t}_{sN}$	$4.25\cdot10^{2}$	$8.48\cdot10^{5}$	**1.00**	$7.59\cdot10^{3}$	10.10
$\sigma_{t_{sN}}$	$6.65\cdot10^{2}$	$9.08\cdot10^{6}$	**0.00**	$1.08\cdot10^{4}$	20.46

Table 15: Comparison of regression-based machine learning methods on all use cases.

Measures	GPyTorch	NNetar	RF	SVR	XGBoost
$\overline{e}_{\mathrm{sM}}$ [%]	25.72	$3.52\cdot10^{2}$	36.72	58.63	**23.85**
$\sigma_{e_{\mathrm{sM}}}$ [%]	35.40	$2.07\cdot10^{4}$	$3.10\cdot10^{2}$	$5.41\cdot10^{2}$	**34.67**
$\overline{e}_{\mathrm{MA}}$	1.13	1.12	1.68	2.14	**1.00**
$\sigma_{e_{\mathrm{MA}}}$	**3.23**	3.71	6.64	9.45	3.27
ρ_U [%]	61.70	53.53	59.07	60.19	60.66
ρ_O [%]	38.13	46.47	40.93	39.81	39.34
δ_U [%]	17.09	**15.71**	16.05	19.26	31.69
δ_O [%]	**5.34**$\cdot10^{2}$	$7.48\cdot10^{2}$	$1.02\cdot10^{3}$	$9.47\cdot10^{2}$	$7.13\cdot10^{2}$
$\overline{t}_{sN}$	$9.39\cdot10^{3}$	$5.92\cdot10^{2}$	$1.42\cdot10^{3}$	$6.19\cdot10^{2}$	**6.73**
$\sigma_{t_{sN}}$	$1.03\cdot10^{4}$	$8.39\cdot10^{2}$	$3.85\cdot10^{3}$	$2.56\cdot10^{3}$	**15.59**

By taking all use cases into account, we can investigate the overall tendency of the methods. All methods tend to underestimate the future vales ($\rho_O > 50\%$). However, methods like ETS, NNetar, sARIMA, and TBATS exhibit only a slight difference between ρ_O and ρ_U. In contrast, the remaining methods underestimate, on average, almost 3/5 of the future values. Also, during the underestimation, the methods are, on average, more accurate than overestimating a time series. In terms of the time-to-result, XGBoost (6.73) is the fastest method of the machine learning methods, while sARIMA ($8.48\cdot10^{5}$) is among all methods the slowest method. Moreover, all methods (except XGBoost and Theta) are at least 100 times slower than sNaïve.

5.7 Summary of the Results and Threats to Validity

While comparing the forecast error, the machine learning methods exhibit a higher forecast error than the classical forecasting methods. This observation is in line with the high forecast error of pure machine learning methods in the M4-Competition [29]. However, the machine learning methods are faster than the classical forecasting methods. Considering the individual methods over all four use cases, no method performs best for all use cases (recall the "No-Free-Lunch Theorem" [35]). For the classical forecasting methods, sARIMA is, on average, the most accurate method, although it is not the best method for any use case. In terms of time-to-result, however, sARIMA is, on average, almost one million times slower than sNaïve. In contrast, the most accurate machine learning method XGBoost is on average 6.73 times slower than sNaïve. Moreover, XGBoost is, for two use cases, the best machine learning method. The baseline method sNaïve achieves only good forecasts when a strong seasonality within a time series is present.

Although we compare the methods on a broad competition comprising a wide range of domains containing 400 different time series, the results may not be generalizable to all time series from all domains. Moreover, we use all methods "out-of-the-box" with their respective default setting. Thus, the individual methods' performance may be different if parameter tuning would have been performed before the forecasting task. We also investigate (i) whether the differences for the observed forecast accuracy are statistically significant and (ii) whether the differences for the measured time-to-result are statistically significant. Consequently, we apply the

Friedman test [12] that is a non-parametric statistical test. More precisely, the test ranks the forecasting methods for each time series separately and compares the methods' average ranks. If there is a tie, average ranks are assigned. Based on this test, we formulate the following hypotheses:

$$H_{0,1}: \text{The methods perform equally regarding the forecast error.}$$
$$H_{0,2}: \text{The methods perform equally regarding the time-to-result.}$$

We conduct both hypotheses with a significance level of 1%. The resulting p-values $p_1 < 2{\cdot}10^{-16}$ and $p_2 < 2{\cdot}10^{-16}$ indicate that both hypotheses can be rejected. Thus, the differences in the exhibited performance of the forecasting methods are statistically significant.

6 CONCLUSION

In this paper, we address RQ 1 *"How to automatically compare different forecasting methods on a level playing field?"* by introducing Libra—a forecast benchmark[14]—that automatically evaluates and ranks forecasting methods based on their performance in a diverse set of evaluation scenarios. The benchmark comprises four different use cases, each covering 100 heterogeneous time series taken from different domains. Moreover, we showed that the assembled data set has a higher diversity than established forecasting competitions such as the well known M-Competitions (RQ 2 *"To what extent does the underlying data set differ from existing data sets?"*). Lastly, we use Libra to perform a broad evaluation of state-of-the-art forecasting methods. In summary, based on the provided data set and the automatic evaluation procedure, the Libra contributes to enhance the comparability of forecasting methods. The benchmarking results for different forecasting methods enable the selection of the most appropriate forecasting method for a given use case.

REFERENCES

[1] Ratnadip Adhikari and R. K. Agrawal. 2013. An Introductory Study on Time Series Modeling and Forecasting. *CoRR* abs/1302.6613 (2013).

[2] Martin Arlitt and Tai Jin. 2000. A Workload Characterization Study of the 1998 World Cup Web Site. *IEEE Network* 14, 3 (2000), 30–37.

[3] V Assimakopoulos and Konstantinos Nikolopoulos. 2000. The theta model: a decomposition approach to forecasting. *International journal of forecasting* 16, 4 (2000), 521–530.

[4] George Athanasopoulos, Rob J Hyndman, Haiyan Song, and Doris C Wu. 2011. The tourism forecasting competition. *International Journal of Forecasting* 27, 3 (2011), 822–844.

[5] André Bauer, Marwin Züfle, Nikolas Herbst, Albin Zehe, Andreas Hotho, and Samuel Kounev. 2020. Time Series Forecasting for Self-Aware Systems. *Proc. IEEE* 108, 7 (7 2020), 1068–1093.

[6] G.E.P. Box and G.M. Jenkins. 1970. *Time series analysis: forecasting and control*. Holden-Day.

[7] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[8] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *ACM Special Interest Group on Knowledge Discovery in Data 2016*. ACM, 785–794.

[9] Sven F Crone, Michele Hibon, and Konstantinos Nikolopoulos. 2011. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of forecasting* 27, 3 (2011), 635–660.

[10] Patrícia Maforte Dos Santos, Teresa Bernarda Ludermir, and Ricardo Bastos Cavalcante Prudencio. 2004. Selection of time series forecasting models based on performance information. In *Fourth International Conference on Hybrid Intelligent Systems*. IEEE, 366–371.

[11] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. 1997. Support vector regression machines. In *Advances in neural information processing systems*. 155–161.

[12] Milton Friedman. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association* 32, 200 (1937), 675–701.

[13] Ben D Fulcher, Max A Little, and Nick S Jones. 2013. Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of the Royal Society Interface* 10, 83 (2013), 20130048.

[14] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. 2018. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. In *Advances in Neural Information Processing Systems*.

[15] Tao Hong, Pierre Pinson, and Shu Fan. 2014. Global energy forecasting competition 2012. *International Journal of Forecasting* 30, 2 (2014), 357–363.

[16] Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli, Rob J Hyndman, et al. 2016. Probabilistic energy forecasting: Global Energy Forecasting Competition 2014 and beyond. *International Journal of Forecasting* 32, 3 (2016), 896–913.

[17] Rob Hyndman, George Athanasopoulos, Christoph Bergmeir, Gabriel Caceres, Leanne Chhay, Mitchell O'Hara-Wild, Fotios Petropoulos, Slava Razbash, Earo Wang, and Farah Yasmeen. 2018. *forecast: Forecasting functions for time series and linear models*. http://pkg.robjhyndman.com/forecast R package version 8.4.

[18] Rob J Hyndman and George Athanasopoulos. 2017. *Forecasting: principles and practice*. OTexts. OTexts.org/fpp

[19] Rob J Hyndman and Anne B Koehler. 2006. Another look at measures of forecast accuracy. *International journal of forecasting* 22, 4 (2006), 679–688.

[20] Rob J Hyndman, Anne B Koehler, Ralph D Snyder, and Simone Grose. 2002. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting* 18, 3 (2002), 439 – 454.

[21] Rob J Hyndman, Earo Wang, and Nikolay Laptev. 2015. Large-scale unusual time series detection. In *2015 IEEE international conference on data mining workshop*. IEEE, 1616–1619.

[22] Yanfei Kang, Rob J Hyndman, and Kate Smith-Miles. 2017. Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting* 33, 2 (2017), 345–358.

[23] Alysha M. De Livera, Rob J. Hyndman, and Ralph D. Snyder. 2011. Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing. *J. Amer. Statist. Assoc.* 106, 496 (2011), 1513–1527.

[24] Spyros Makridakis. 1993. Accuracy measures: theoretical and practical concerns. *International journal of forecasting* 9, 4 (1993), 527–529.

[25] Spyros Makridakis, A Andersen, Robert Carbone, Robert Fildes, Michele Hibon, Rudolf Lewandowski, Joseph Newton, Emanuel Parzen, and Robert Winkler. 1982. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of forecasting* 1, 2 (1982), 111–153.

[26] Spyros Makridakis, Chris Chatfield, Michele Hibon, Michael Lawrence, Terence Mills, Keith Ord, and LeRoy F Simmons. 1993. The M2-competition: A real-time judgmentally based forecasting study. *International Journal of Forecasting* 9, 1 (1993), 5–22.

[27] Spyros Makridakis and Michele Hibon. 1979. Accuracy of forecasting: An empirical investigation. *Journal of the Royal Statistical Society: Series A (General)* 142, 2 (1979), 97–125.

[28] Spyros Makridakis and Michele Hibon. 2000. The M3-Competition: results, conclusions and implications. *International journal of forecasting* 16, 4 (2000), 451–476.

[29] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2018. The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting* 34, 4 (2018), 802–808.

[30] Albert H Nuttall and G Clifford Carter. 1982. Spectral estimation using combined time and lag weighting. *Proc. IEEE* 70, 9 (1982), 1115–1125.

[31] Carl Edward Rasmussen and Christopher KI Williams. 2006. *Gaussian processes for machine learning*. Vol. 2. MIT press Cambridge, MA.

[32] Maxim Vladimirovich Shcherbakov, Adriaan Brebels, Nataliya Lvovna Shcherbakova, Anton Pavlovich Tyukov, Timur Alexandrovich Janovsky, and Valeriy Anatol'evich Kamaev. 2013. A survey of forecast error measures. *World Applied Sciences Journal* 24, 24 (2013), 171–176.

[33] Timo Teräsvirta, Chien-Fu Lin, and Clive WJ Granger. 1993. Power of the neural network linearity test. *Journal of time series analysis* 14, 2 (1993), 209–220.

[34] Jóakim v. Kistowski, Jeremy A. Arnold, Karl Huppler, Klaus-Dieter Lange, John L. Henning, and Paul Cao. 2015. How to Build a Benchmark. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering* (Austin, Texas, USA) *(ICPE '15)*. ACM, 333–336.

[35] D. H. Wolpert and W. G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (Apr 1997), 67–82.

[14] Available at GitHub: https://github.com/DescartesResearch/ForecastBenchmark