Comparison of methodologies for forecasting multiple univariate time-series

Abstract

Time-series analysis is a useful branch of statistics for understanding trends and patterns in the data over time and for forecasting future events. This paper looks into the latter, i.e. forecasting, and examines how well a deep learning architecture (the Transformer model) performs in the context of forecasting multiple univariate time-series, as compared to benchmark methods and a clustering method based on data mining and statistical techniques. The benchmark models include Näive Drift, Holt's exponential smoothing, and ARIMA. The forecasting accuracy of the examined models is evaluated by masking a horizon of the latest years in all time-series, forecasting estimates for that horizon, and computing the error in the predictions compared to the ground truth using symmetrical Mean Absolute Percentage Error (sMAPE). Overall, we find very good performance by all benchmark methods and for the Transformer method, with median sMAPEs across all time-series below 2 %. The cluster model shows a median sMAPE of 2.98 %, however with much higher variation than the other models. The time-series used in this paper are generally highly uniform with stable trajectories, which likely contributed to the high performance of benchmark methods and the low performance of the cluster model. Additionally, the Transformer appears to produce better forecasts for the few time-series that are more difficult to predict. This points towards the advantages of cross-learning in global models. Advantages and limitations of the various approaches to forecasting multiple, univariate time-series are presented and discussed, as well as considerations for how to improve results and the comparative methodology.

Link to Github Repository with analysis code:

https://github.com/evasahlholdt/datascienceexam

Introduction

In order to obtain confidence in methods used for forecasting future events, it is crucial to evaluate and compare the performance of various approaches on real data. A prominent avenue for this type of research are forecasting competitions such as the NN3 and NN5 competitions, Kaggle's forecasting competitions, and in particular the M competitions (Godahewa et al., 2021). These competitions aim to empirically identify, compare, and improve the most accurate forecasting methods by allowing interested analysts to apply their methods of choice in order to solve a given forecasting challenge (Makridakis et al., 2022). These competitions allow for comparing model performance across a diverse range of applications, and the insights derived from them have had valuable influence on the domain of forecasting from time-series (Hyndman, 2021). Forecasting accuracy is measured in terms of different error metrics, which generally aim to capture how much a forecast deviates from the ground truth. Over the course of the five M competitions, which have been conducted over the past 40 years, the most successful forecasting methods have generally been quite simple statistical methods such as simple exponential smoothing (Makridakis et al., 1982) and the theta method (Assimakopoulos & Nikolopoulos, 2000). It was not until the M4 competition in 2018 that a few select models, employing more advanced algorithms that combine statistical and machine learning methods, surpassed the performance of simple forecasting models. In contrast, the results from the M5 competition in 2022 seriously demonstrated the capacity of machine- and deep learning methods, which outperformed the best statistical benchmarks with more than 20 % accuracy (Makridakis et al., 2022). Whilst the latest M competition apply a complex, multivariate time-series dataset, which is arguably more difficult to model with statistical methods than univariate time-series, there is reason to believe that machine- and deep learning models also can provide improved forecasting accuracy for univariate time-series when enough data is available, as compared to their statistical counterparts. In the following, we will introduce the challenge of modelling multiple univariate time-series, and present both a model mixing approaches from data mining and statistics, and a deep learning approach based on the Transformer architecture to this problem.

Univariate time-series and the challenge and prospects of multiple series (Eva)

When dealing with time-series data in forecasting, univariate series are the simplest cases to work with. Whilst multivariate time-series are multidimensional with several interrelated time-series variables, univariate time-series consist only of the series of observations which are to be predicted (Naik et al., 2022). Throughout a long research tradition, numerous approaches have been developed, each tailored to the specific characteristics of the data (Hyndman, 2021). Despite the rapid increase in data availability and the development of more complex multivariate models, forecasting based on univariate time-series remains ubiquitous and relevant in many applications. However, a common constraint when forecasting based on univariate time-series is that the only information available is that which is nested in the time-series themselves. Thus, explanatory variables, which may correlate with or causally affect the target time-series, are not explicitly considered - such as e.g. modelling holiday seasons in a sales forecast. Nonetheless, forecasting from univariate time-series can be quite precise when the patterns in the series are relatively stable and the historical data is sufficient to encapsulate on-going patterns (Hyndman, 2021).

Generally, for both uni- and multivariate time-series, the typical analytical approach is to apply local methods, i.e., analysing each series individually. As local methods are well-established, and are generally simple to work with and to interpret, they remain popular in real-world applications. Further, they often perform reasonably well, especially when the data exhibits stable trends or seasonality in the series (Hyndman, 2021). A central advantage of local methods in time-series analysis is that the forecast will be specialised uniquely to the time-series at hand. However, this specialisation can be a double-edged sword, for example in cases where the historical data of the particular time-series does not contain the information necessary to produce valid forecasts (Szabłowski, 2023). Furthermore, various issues are introduced as the numbers and scale of individual time series increase. For instance, when forecasting time-series individually, preliminary data exploration is essential for selecting the most suitable local method for the current analysis. This could include examining the sample for trends, seasonality and identifying and addressing anomalies, outliers, and missing data. For large amounts of time-series, this process becomes extensive (Szabłowski, 2023). These are central arguments relevant for the establishment of *global models,* which are trained on a multitude of time-series. Global models are relatively novel, arising in unison with the development of machine- and deep

learning methods (Han et al., 2021). In the case of multiple univariate time-series, their primary benefit is that they can extrapolate upon a single series based on the patterns and representations learned from seeing many other series. This can circumvent the aforementioned issues in fitting individual time-series locally. However, it is generally not as straight-forward to train, apply, and interpret results from global models as compared to local methods. Therefore, when developing and implementing complex or novel forecasting methodologies, it is standard procedure to assess how well simple, local methods (so-called benchmarks) perform on a shared forecasting task (Hyndman, 2021).. This way, one can both establish a baseline for how much error can be expected, and motivate whether it is worth implementing a more complex approach in the hope of increasing performance. Here, we compare the performance of baseline approaches applied locally with that of a global deep learning approach, as well as an approach combining data mining methods and simpler statistical methods. Specifically, the benchmark models selected in this paper include 1) Naïve drift, 2) Holt's exponential smoothing, and 3) ARIMA.

Näive drift (Liv)

The Näive Drift method is the simplest forecasting method applied in the current analysis. Whilst the Näive method assigns each forecast the same value as the most recent observation, Näive Drift combines this approach with an estimate of the average change over time (i.e. a general increase or decrease, a “drift” or “trend”) in a time-series throughout the historical observations. Thus, the Näive Drift method considers both the immediate and distant past, and is a suitable choice for a very simple benchmark method for linear time-series. For historical data of length T, a time horizon (number of time steps) h and the current observation y, the forecast $\hat{y}$ is given by:

$$\hat{y}_{T+h} = y_T + h\left(\frac{y_T - y_1}{T-1}\right)$$

Holt’s exponential smoothing (Liv)

As Näive Drift, Holt’s exponential smoothing is a suitable method for forecasting time-series with trend (Hyndman, 2021). It is an extension of simple exponential smoothing (SES), which balances between leveraging the most recent observations and taking into account the distant past, somewhat as a compromise between the Näive method (which discards all but the most recent information) and the mean method (which discards all information about temporal relations). In SES, the forecast is computed by calculating a mean of past observations which is

weighted according to recency, so that the most recent observations contribute more to the weighted average (by exponentially decreasing the weights according to distance in time). Thus, the past is "smoothed" out, according to the smoothing parameter α (0 ≤ 1), where α ≈ 0 weights the distant past more, whilst α ≈ 1 weights the recent observations more. Holt's exponential smoothing extends upon this principle by incorporating an additional smoothing parameter for trend β, so that the forecast $\hat{y}$ is given by addition of two smoothing equations (with the first for the level ℓ equivalent to the SES equation), for a time point *t* and a forecasting horizon *h*:

$$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$$

$$b_t = \beta\ (\ell_t - \ell_{t-1}) + (1-\beta\ )b_{t-1}$$

$$\hat{y}_{t+h|t} = \ell_t + hb_t$$

ARIMA (Liv)

The ARIMA (AutoRegressive Integrated Moving Average) model is the most advanced baseline method applied in the current approach. It is a popular method applied across various domains, as it can be adapted for different use cases and because it considers both short-term and long-term dependencies in time-series data (Hyndman, 2021). It combines an autoregressive (AR) component (which considers the relationship between an observation and its lagged value), a differencing (I) component (which transforms the data to achieve stationarity), and a moving average (MA) component (which models the error terms as a linear combination of past errors). In short, the autoregressive parameter (AR) determines the number of lagged terms to include in the model and their respective coefficients. Each lagged term contributes to the prediction based on its coefficient and the value at the corresponding lag. The differencing parameter *d* controls how many degrees of differencing are involved. The MA parameter *q* determines the size of the moving average window, i.e. specifies the number of lagged error terms (residuals) included in the model. Thus, it is included to capture fluctuations and noise which is not explained by the autoregressive component. The full equation (where y_t' is the differentiated time-series) for the non-seasonal ARIMA model is thus given by:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

The cluster model (Liv)

In this paper, we implement a semi-supervised clustering approach developed to efficiently forecast multiple univariate time series (Banik, 2022). The core concept behind this approach (henceforth referred to as the cluster model) is to reduce computational load and processing time by clustering temporally similar time series and leveraging the general patterns in each cluster to make individual forecasts. Time-series clustering is an unsupervised data mining method used to arrange time-series into groups according to their similarity, which for univariate time-series refers to their temporal dynamics (Yeshchenko et al., 2019). The aim is to group time-series in a way that maximises similarity within each cluster and minimise it between clusters. This is achieved using a clustering algorithm, of which the optimal choice depends on the type of data used for analysis. The cluster model was presented using the bisecting K-means algorithm for clustering, which draws on aspects from K-means clustering as well as Divisive Hierarchical Clustering. In this algorithm, the data is divided into two sub-clusters (bisected) via the K-means algorithm. This is a centroid-based clustering algorithm, which assigns data points into clusters by minimising the sum of squared distances between them and a randomly selected cluster centroid (Dabbura, 2022). After the time-series have been assigned to a cluster, K-means updates the centroid for each cluster by averaging over the values currently assigned to the cluster. This process is repeated until the cluster centroids are no longer updated and all time-series have been assigned to a cluster that does not change for each new iteration. The algorithm then selects the sub-cluster with the largest sum of squared distances for further splitting via K-means. The process of splitting and selecting continues until the desired number of clusters (K) is obtained. Since the bisecting K-means algorithm gradually increases the number of clusters, rather than attempting to optimise a pre-defined number of clusters all at once, it is considered more efficient than the traditional K-means algorithm when dealing with large data sets (Martins, 2023). After clustering the time-series, a time series representative of the general temporal dynamics of the series contained in each cluster is calculated. This is done by taking the median across all series in the cluster. A forecast is generated to teach representative time-series using a methodology depending on the specific characteristics of the series. This way, general temporal

components such as trend and seasonality present in the cluster are captured efficiently. To extract forecasts for each local series in the cluster, the representative forecast is added to forecasts produced on adjusted series for each individual series. These adjustment series are calculated by subtracting the values of the representative from each actual individual time-series. The final individual forecasts are made using straightforward approaches such as e.g. simple linear modelling.

Global modelling with multiple time-series (Eva)

Before the current acceleration in applying deep learning in context of time-series forecasting, global modelling using e.g. recurrent neural networks (RNNs) did not show particularly promising results, as revealed in the previously mentioned M competitions (Gamboa, 2017; Makridakis et al., 2022). This picture is changing, and the importance of developing more powerful models is increasing in line with the rise of big data (Naik et al., 2022). While there is a diverse range of global models applicable to analysis of multiple time-series, they all share some common characteristics. Global models benefit from being able to learn latent representations across individual time-series, making it possible to leverage general patterns in local predictions via cross-learning (Makridakis et al., 2022). This can be an advantage for example when forecasting short or incomplete series (Orlandi, 2022). Also, a previously trained global model can be applied to forecast novel time-series via transfer-learning. It should however be mentioned that cross- and transfer-learning require that the time-series share some characteristics in order to obtain meaningful forecasts, particularly in relation to temporal dynamics. This relates to another limitation of global models, which is that these architectures generally demand a substantial amount of data in order to extract meaningful and reliable representations between time-series (Naik et al., 2022). A sufficient quantity of data is also important to avoid overfitting, as most deep learning architectures are designed to handle complex relationships between many features. This is especially true for analysis of multiple univariate time series, as the lack of correlating features demands that all feature representations are extracted directly from the target values (Szabłowski, 2023). Out of the increasingly diverse range of machine- and deep learning models applicable in a univariate time-series setting, we focus on a relatively recent architecture which has attracted a lot of attention especially in the field of natural language processing, namely the Transformer (Wen et al., 2023).

Transformer models in time-series analysis (Eva)

The Transformer is a deep neural network architecture released in 2017 by a team from Google Research (Vaswani et al., 2017). It is a generative sequence-to-sequence model, which creates a sequential output given a certain input sequence. It consists of an encoder stack, which takes an input sequence and produces a context vector (representation) of the essential information in the input. This is then used in a decoder stack, which generates an output sequence (target) by autoregression, i.e. by generating one element at a time and using each generated element to produce the next. The Transformer architecture, initially designed for natural language processing (NLP), introduced the use of multi-head attention mechanisms for encoding both internal dependencies within input sequences (self-attention) and external dependencies between input and output sequences (encoder-decoder attention). In short, self-attention in the encoder stack captures relationships and dependencies between elements in the input sequence, by assigning weighted attention scores and determining the relative importance of each element pair. This allows the model to attend to all positions in the input sequence, learning contextualised representations for each position. During the autoregressive generation process in the decoder stack, self-attention enables the model to attend to each generated output element in the context of the full generated sequence, which is important in the generation of the next element. The encoder-decoder attention mechanism allows the model to leverage the encoded representations of the input sequence during the generation of the output sequence. This enables the model to capture relevant aspects of the input and produce a novel output sequence based on individual element representations and their relationships in a multi-dimensional space (Vaswani et al., 2017). It also facilitates handling long-range interdependencies between elements. Further, the multi-head attention structure is parallelizable, which significantly increases efficiency by reducing the amount of sequential processing compared to e.g. RNNs.

As natural language is essentially a special case of sequential data, the architecture can be generalised to other types of sequential data (Wen et al., 2023). In forecasting time-series data, the model is prompted to generate a forecast sequence following the time axis of the training sequence. The attention mechanisms in the model enables learning of the relationships between observations at different time steps. This is achieved by updating attention weights through backpropagation, which determines the relevance of observations (in NLP, this teaches the model

e.g. which words frequently co-occur). Consequently, the model captures temporal dependencies, recognizes and prioritises important patterns in the data, and extrapolates upon them during forecast generation (Wen et al., 2023). Current implementations of the Transformer in a time-series forecasting context have shown mixed results, with some implementations showing state-of-the-art performance (e.g. Wu et al., 2020; Lim et al., 2021; Tang & Matteson, 2021) whilst others find simple methods to outperform the Transformer (e.g. Zeng et al., 2022). Given the fact that the architecture is still young, implementations and modifications in a time-series forecasting context remain under development. For instance, in NLP, applications often benefit greatly from fine-tuning models that are pre-trained on tremendous amounts of data. Similarly, pre-trained large time-series models might hold great promise (Wen et al., 2023). Much of this work relies on the original "vanilla" Transformer architecture, but alternative configurations better suited for different cases of time-series are being explored (Wen et al., 2023). Lastly, the current implementation from the Darts library does not fully exploit the output sequences during the autoregressive process, but considers only the last value in the input sequence for the generation of the next target. Ideally, the generated target sequences should be added as inputs for next-step generation, which remains work in progress (Herzen et al., 2022).

Data applied in current paper **(Liv)**

We base our analysis on a dataset of expected life spans for humans across multiple independent time-series, obtained from Our World in Data, combined from various sources (see Roser et al., 2013). The original dataset consists of 257 independent time-series ranging from year 1543 to 2021, with one estimate of life expectancy per year. Life expectancy measures the average age of death in a population and is thus a metric capturing the mortality rates in a country (Roser et al., 2013). The population groups in the current data mainly consist of country- and territory level groups, why the estimate is useful to assess population health in a country and inequalities between countries. In general, life expectancies have been increasing globally over the past century, alongside industrialization and medical development (Roser et al., 2013). This dataset was selected for the purpose of the current analysis as it represents a simple case of multiple, independent, univariate time-series of relative homogeneity. This is considered suitable and easy to interpret in the context of both local- and global modelling approaches.

Methods

Analysis platform **(Eva)**

The current analysis is performed using Python 3.10 and relies primarily on the library Darts (Herzen et al., 2022). Darts is developed for time-series analysis and includes machinery for the full pipeline related to time-series analysis. It contains various methods and models including traditional statistical approaches as well as PyTorch-based machine- and deep learning approaches, and supports analysis of multiple time-series. We leverage Darts' pipeline for large parts of our analyses. Additionally, we utilise the libraries Seaborn and Matplotlib for visualisations.

Data specifications and exploratory data analysis **(Eva)**

Whilst the original dataset consists of data ranging back to 1543, initial exploration revealed that most time-series have sparse and incoherent estimates especially before the 20th century. Therefore, to avoid missing values and increase the likelihood that the estimates included are reliable, we exclude all observations before the year 1950. Further, to avoid a hierarchical structure, we exclude grouped time-series (e.g. continents). Thus, our dataset consists of 237 individual time-series at a country- and territory-level with 72 observations each. To inspect our data, we compute a profile report of the data structure using pandas-profiling for time-series (3.6.6). The report confirms that there is no missing data nor duplicates. As we only have yearly estimates, no seasonality is detected. Additionally, to get an idea of the overall autocorrelation (the relationship between a current measure and the past measure) in our time-series, we assess the autocorrelation for each time-series individually and compute the median, mean and standard deviation across all time-series. Here, we find a median autocorrelation of 1, with mean 0.99 and sd 0.03. An autocorrelation of 1 indicates a perfect positive correlation between a time-series and the lagged time-series, i.e. an increasing linear trend, which is confirmed by visualising the overall trend of the time-series (figure 1). A low standard deviation implies that generally, most time-series show the same behaviour.

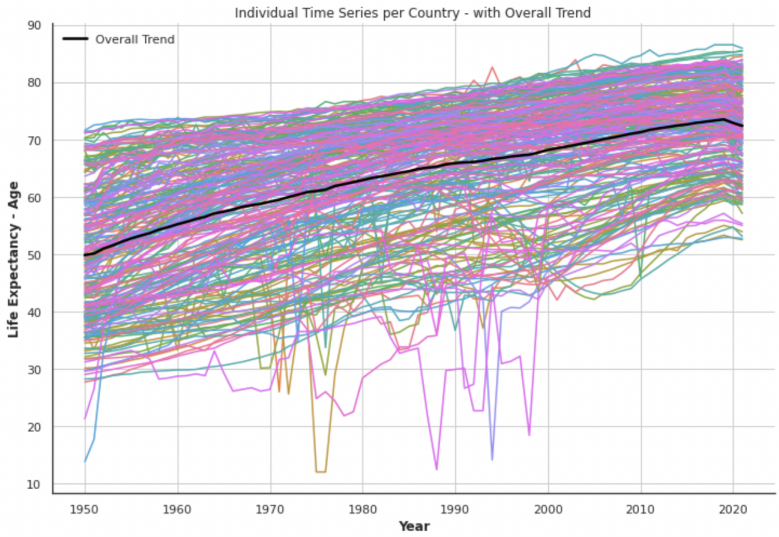

Figure 1: All individual time-series plotted together. The plot demonstrates the general trend of global life expectancy, which exhibits a predominantly linear pattern.

Additional preprocessing and data splitting **(Liv)**

To enable analysis through the Darts library, the data is transformed into TimeSeries objects. In order to split our data into training- and testing sets, we choose a simple approach of splitting the data by year, as the temporal dependencies in time-series data exclude the option of randomised splitting. For all local methods, the training data includes all observations before year 2012 (62 years) and the test data includes all observations after year 2012 (ten years). Thus, our forecasting horizon (i.e. the number of years into the future we produce forecasts) is ten years. For the Transformer model, we create an additional split for validation to enable hyperparameter tuning. Thus, the training set includes all data before 2002 (52 years), the validation set all data between 2002 and 2011 (ten years), and the test set all data from 2012 (ten years). Thus, our training set consists of 86 % of the data for the local methods, whilst it consists of 70 % for the Transformer model. Further, due to the architecture of the Transformer and for optimization reasons, the target values (life expectancies) are scaled to between 0 and 1 using Darts' Scale() function.

Evaluation and error metrics (sMAPE) **(Liv)**

Evaluation of model performance is done by fitting the model on the training split and generating forecasts for each time series for the forecasting horizon equal to the length of the testing set (ten

years). This makes it possible to compare the model predictions to the ground truth. For evaluation, we apply the symmetric mean absolute percentage error (sMAPE), which with a time-series of actual values *y* and a time-series of predicted values $\hat{y}$ both of length *T* for each time point *t* is given by:

$$200 \cdot \frac{1}{T} \sum_{t=1}^{T} \frac{|y_t - \hat{y}_t|}{|y_t| + |\hat{y}_t|}$$

Thus, sMAPE computes a percentage measure of the error between actual and predicted time points. In our case, sMAPE is a suitable error metric as we do not have estimates close to zero (which can cause division issues), and because it is symmetrical, i.e. treats over- and underestimations equally (in contrast to e.g. MAPE) (Lewinson, 2020). To establish overall performance across all time-series, we compute the median sMAPE as well as the mean and standard deviation for each method. Further, to make it possible to explore what may underlie the cases in which the produced forecasts are inaccurate, we extract the time-series which yields the highest sMAPEs.

Model specifications

Local methods **(Liv)**

The three baseline methods applied in the current analysis are all selected to accommodate forecasting of time-series with trend, but without seasonality. Näive Drift does not include any particular configuration. To apply Holt's exponential smoothing, we apply Dart's built-in exponential smoothing function (which is based on Holt-Winters exponential smoothing technique), but set seasonality mode to "none". We use an additive trend, which is standard when the trend in a data sample is linear. Lastly, we add damping to the model, which is a particularly useful parameter for time series with a trend that is expected to decrease over time (Hyndman, 2021). The damping parameter flattens the forecasts as time progresses, and thus helps to avoid overestimating the life expectancy in our data. The ARIMA model demands a bit more careful configuration, with appropriate configuration of the autoregressive component (AR, or *p)*, the integration component (I, or *d)* and the moving average component (MA, or *q*). As we find an overall autocorrelation of 1 across our time-series, we specify $p = 1$, which means that a current value is estimated as a linear combination of its immediate past value (lag 1) with a coefficient of

1. Further, we specify $q = 0$, as an autocorrelation of 1 indicates that the residuals are already considered in the linear relationship included in the autoregressive component. Lastly, we specify $d = 1$ as we assume that one level of differentiation is sufficient to account for the trend in the series.

Clustering Model **(Liv)**

The Bisecting K-means algorithm was used to cluster the time series. The optimal number of clusters, K, to use in the algorithm was determined by calculating the sum of squared distances (inertia) between cluster centroids and individual data points for different values of K. The elbow plot in *Figure 2* demonstrates that the rate of decrease in sum of squared distances significantly diminishes beyond K = 4. For this reason, and because too many clusters introduce a heightened risk of decreased generalizability and overfitting, four clusters were deemed suitable for the dataset. These clusters are visualised in *Figure 3*. The algorithm was set to use the biggest inertia as its bisecting strategy, over the less computationally heavy option of bisecting the largest cluster. Once all clusters had been obtained, the representative time series were fitted with Holt's exponential smoothing model, incorporating a damped trend to ensure the forecasts maintain a realistic development. Lastly, the adjustment series were modelled using a simple linear model.

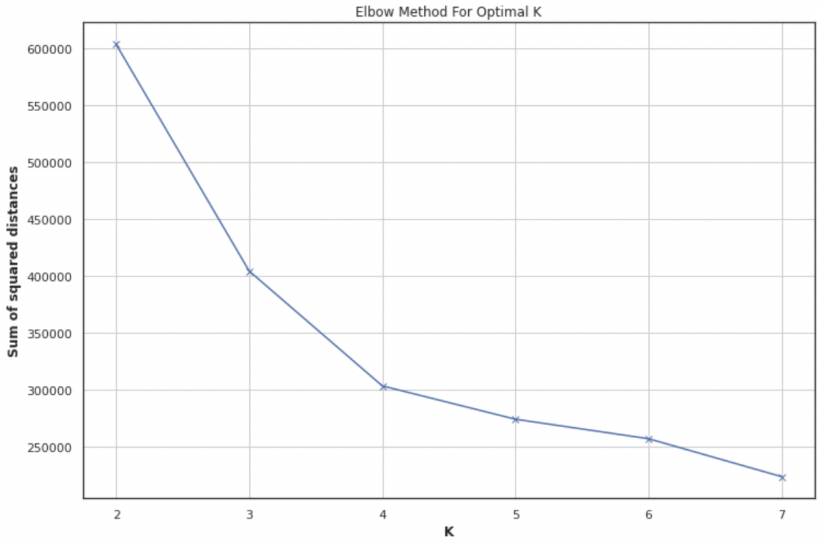

Figure 2: Elbow plot showing the sum of squared distances for different values of K. The inertia decreases steadily until K = 4.

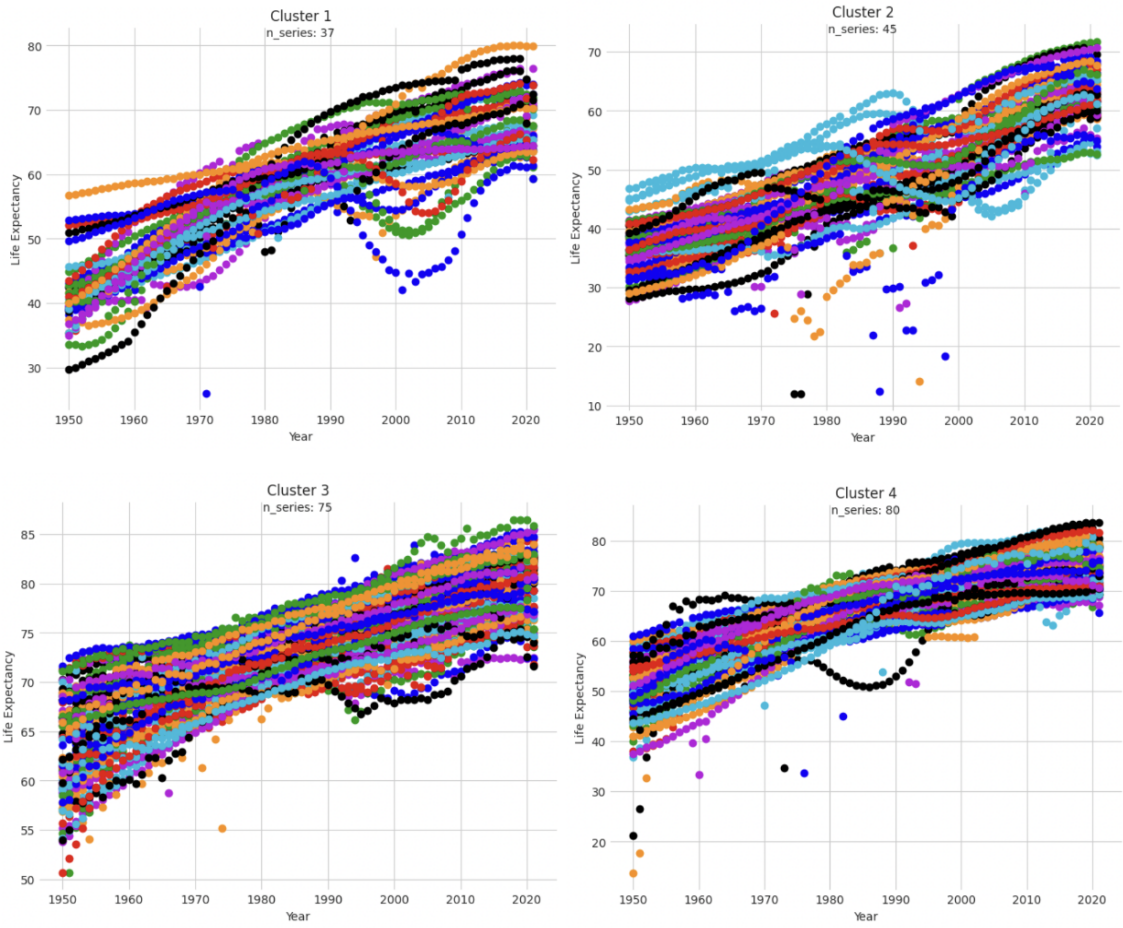

Figure 3: The four clusters obtained with the bisecting K-means algorithm. Note that cluster 1 and 2 include less homogenous, and fewer, time series than cluster 3 and 4.

Transformer model **(Eva)**

A model's hyperparameters refer to the parameter values explicitly applied during model training to control the training process (Özateş, 2022). Correct configuration of the hyperparameters is thus important to increase the accuracy of the model. We initially configure the hyperparameters of the Transformer to accommodate training on multiple univariate time-series based on advice from the modelling community's experiences with this use case, generally lowering the default values used in the usual NLP settings to avoid risk of overfitting, as univariate time-series have much fewer features than natural language (Onnen, 2022). Whilst model performance can benefit from optimization by searching the space of different hyperparameter configurations, this process is demanding in time and processing power (Kamsetty, 2020). Therefore, we perform

hyperparameter optimization for only selected hyperparameters, namely the length of the sequence of time steps given as input- and output chunks, the number of expected features in the transformer encoder/decoder inputs (i.e. input dimensionality), and the number of heads in the multi-head attention mechanism (which matters in relation to the input dimensionality, as the relation between the two decides how many attention heads are employed for each feature). We perform this search using the library Optuna, which is based on Bayesian optimization processes to sample the hyperparameter space in machine learning settings to identify the hyperparameters leading to the lowest error on validation data (Akiba et al., 2019). After tuning, we implement the optimal hyperparameters when training the final model.

Results

Local methods (Liv)

The figure below (*Figure 4)* shows the distribution of sMAPEs as well as median, mean, and standard deviation of the sMAPEs over a ten years forecasting horizon for each individual time-series for 1) the Näive Drift method, 2) Holt's exponential smoothing technique, and 3) the ARIMA method. The first thing to note is that all methods show good forecasting performance. Näive Drift (ND) produces a median error of 1.69 % and a mean error of 2.06 with a standard deviation of 1.69. Exponential smoothing (ES) produces a median error of 1.07 % and a mean error of 1.43 % with a standard deviation of 1.65. ARIMA produces a median error of 1.55 % and a mean error of 1.94 % with a standard deviation of 1.86. It generally appears that the majority of the individual forecasts are well below a 5 % error for all methods, whilst a few time-series seem to be difficult to predict. For forecasts below a 5 % error, the mean error is below 2 % for all methods (ND: 1.85 %, ES: 1.27 %, ARIMA: 1.7 %), indicating that most time-series are well-suited for the baseline methods applied (i.e. follow a linear trend without too much noise). For forecasts with errors above 5 %, the picture is more diverse. Here, we find errors up to 20.84 % (ES), and the mean and standard deviations of forecasts above a 5 % error show that they are significantly harder to predict than the majority of time-series below 5 % (ND: 9.03 %, 3.37; ES: 11.15 %, 5.25; ARIMA: 10.02 %, 4.75). However, these cases of hard-to-predict time-series are few (only a handful of the 257 series) for all methods.

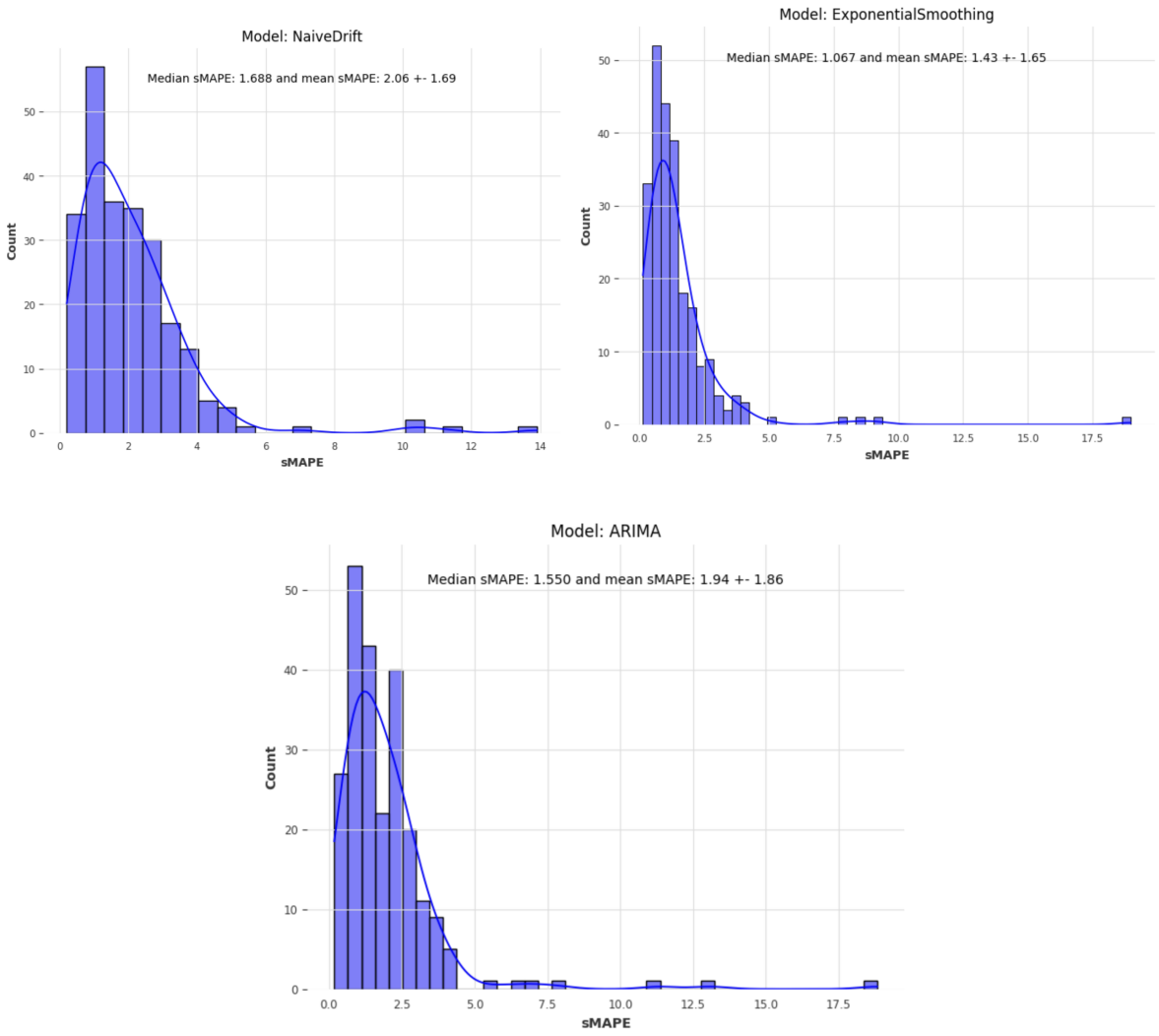

Figure 4: Distribution of sMAPEs as well as its corresponding median, mean, and standard deviation over a ten years forecasting horizon for each of the local models applied.

Cluster model (Liv)

The figures below show the overall and cluster-level performance of the cluster model. Overall, the median sMAPE value is 2.98 %, with a standard deviation of 5.6. *Figure 5* reveals that most errors lie between 0% and 8 %, and that a few odd series contribute to the high standard deviation. *Figure 6* further illustrates performance in each cluster. It becomes evident that there is a difference in performance between the two first and the two last clusters, with much higher standard deviations in cluster one and two (~ 8) compared to cluster three and four (~ 2 to 3). The median sMAPE in cluster three (1.99 %) and four (2.77 %) are also more comparable to the

benchmarks errors. Of all clusters, cluster three performs best, with a mean of 2.46 % and standard deviation of 1.66.

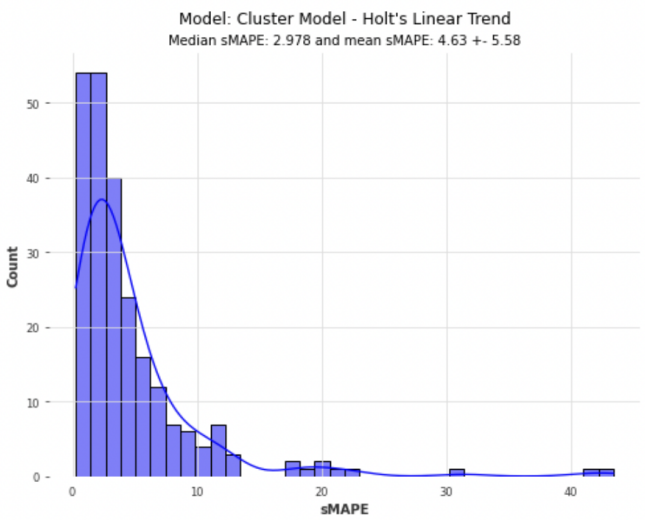

Figure 5: Distribution of sMAPE values across all clusters in the cluster model. The overall mean sMAPE for the cluster model is 4.71, and the overall median is 3.04.

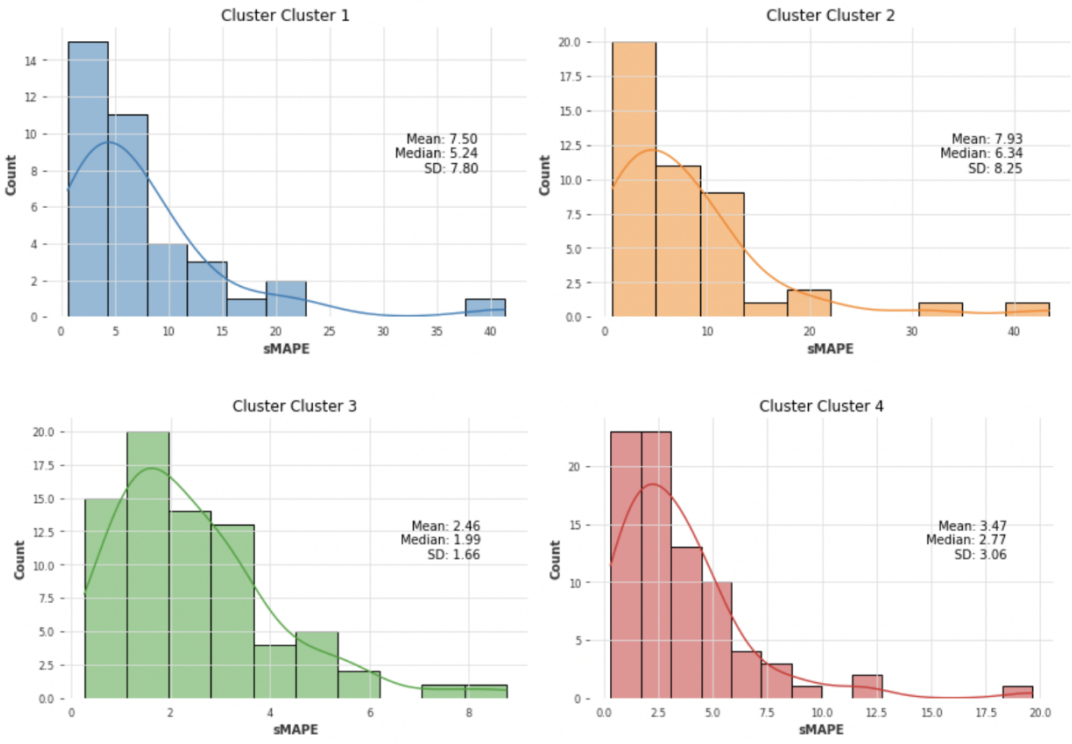

Figure 6: Distribution of sMAPE values per cluster. sMAPE values range from around 0% to around 40% in the first two clusters. sMAPE values in cluster 3 range from around 0 % to around 9%. In cluster 4, the sMAPE values range from around 0% to around 20%.

Transformer model (Eva)

Overall, the Transformer model shows the second-best performance in terms of overall error, with a median sMAPE of 1.31 % and a mean of 1.45 % with a standard deviation of 0.69 (*Figure 7*). Exponential smoothing slightly outperforms the Transformer at these metrics (median: 1.07 %, mean: 1.43 %, standard deviation: 1.65). However, the Transformer shows the best performance in regards of variance, as no forecasts are more than 6.71 % from the ground truth, and only six forecasts produce errors above 3 % (in contrast to 44, 17, and 30 forecasts with errors exceeding 3 % for Näive Drift, exponential smoothing, and ARIMA, respectively).

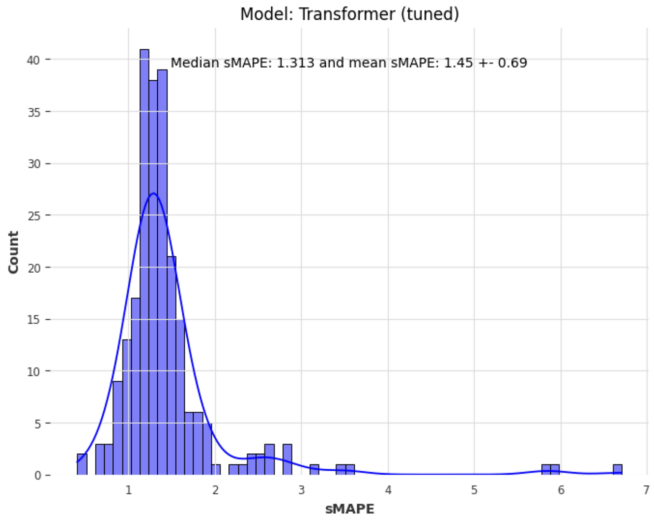

Figure 7: Distribution of sMAPEs as well as its corresponding median, mean, and standard deviation over a ten years forecasting horizon for the Transformer model.

Exploratory comparison (Liv)

Highest error	Näive Drift	Holt's exponential smoothing	ARIMA	Transformer	Cluster model
1	Eswatini, 13.91 %	Libya, 20.84 %	Libya, 17.74 %	Syria, 6.71 %	Eswatini, 43.40 %
2	Syria, 11.19 %	Zimbabwe, 9.94 %	Syria, 11.77 %	Eswatini, 5.91 %	Zimbabwe, 41.32 %
3	Zimbabwe, 10.43 %	Syria, 9.88 %	Haiti, 10.97 %	Zimbabwe, 5.80 %	Lesotho, 31.40 %

4	Lesotho, 10.09 %	South Africa, 9.37 %	South Africa, 7.61 %	Namibia, 3.56 %	Botswana, 22.81 %
5	Yemen, 6.99 %	Somalia, 7.07 %	Yemen, 5.94 %	Malawi, 3.46 %	Zambia, 21.73 %

Table 1: The table includes the five countries with the highest calculated sMAPE values for each of the 5 models assessed. The countries' sMAPE values are listed in descending order for each model.

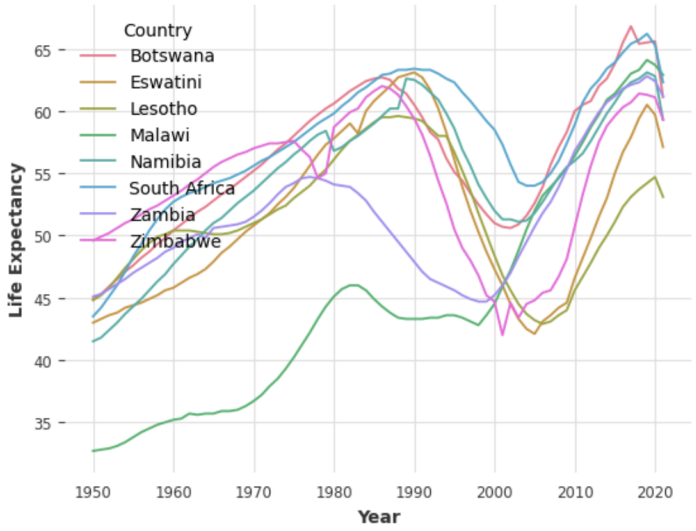

Figure 8: Trajectories for some of the time-series with the highest errors in produced forecasts.

Table 1 demonstrates that for all methods, there are certain time-series that appear difficult to forecast accurately. For example, the countries included in *Figure 8* are amongst the series with the highest errors across multiple methods, baseline as well as for the Transformer and cluster model. The figure illustrates how these series show a similar pattern, with a significant decrease in life expectancy in the decades immediately before the forecasting horizon (at year 2012).

Discussion

(Eva)

Before all, it is important to note that it is difficult to draw general conclusions based on the current analysis. Generally, there is very little error in all produced forecasts apart from in the clustering model. This is likely due to the relatively short forecasting horizon and the fact that steadily increasing life expectancy across the globe makes most time series exhibit stable, linear

trajectories (Roser et al., 2013). Thus, there is little room for improvement, and any discussion will therefore concern only marginal differences in terms of model performance. Ideally, a comparative, methodological analysis would include empirical testing across a diverse range of datasets to establish a more generalizable image of performance, as it is increasingly being performed in the literature (e.g. using the Monash Forecasting Repository, Godahewa et al., 2021).

Whilst our overall finding that the local benchmark methods generally perform well in forecasting life expectancy, the overarching aim of this analysis was to establish which methods produce the best results across a multitude of independent time-series at once. In that regard, we find that whilst the baseline methods applied locally perform well for series which follow the global pattern (with errors generally below 5 %), their performance diminishes for time-series that show unexpected trajectories of observations (with errors between 5 and 21 %). Here, we find that the deep learning architecture applied is able to capture accurate predictions for highly predictable series (with the same precision as local methods), but also produces more accurate forecasts for time-series that are less linear. For instance, the transformer displays approximately half the error as compared to local methods on the hardest-to-predict series, with the highest error being 6.71 % for Syria, for which the best competing forecast (exponential smoothing) shows an error of 9.88 %. In contrast, the cluster model performs worst overall, both in terms of median error as well as for forecasts on non-linear series. Here, performance is significantly worse for two out of four clusters, which affect the estimates of the model's overall performance.

(Liv)

The results of the current analysis point towards a number of interesting considerations in relation to the problem of forecasting multiple univariate time-series. The main challenge when attempting to produce a multitude of forecasts is how to best accommodate the variation between time-series. Producing forecasts using local methods generally demands careful data inspection to select a suitable method. Whilst we see good performance on our local methods, which are chosen based on overall characteristics of all time-series, we observe that they prove insufficient for series with sudden changes in recent observations (such as high child mortality due to an epidemic), and for series that appear non-linear (such for regions with recurrent turmoil). For all

baseline methods, sudden changes in the recent past are problematic as they all weigh recent observations heavily when producing forecasts, and non-linear trajectories make it difficult for the methods to establish the correct regression coefficient for the trend component. Compared to local methods, the global modelling approach leaves more room for variance between time-series, if sufficient data with some shared statistical characteristics is provided for the model to learn which trajectories are similar and dissimilar. Given the improved forecasting accuracy in series with sudden events and non-linear trajectories as compared to the benchmarks, it appears that the deep learning approach has indeed managed to identify and generalise those slightly different characteristics. In contrast, it appears that the clustering method reduces performance as compared to the baseline methods, probably due to clustering of dissimilar series which skews the individual forecast when providing cluster-level information.

Cluster model **(Liv)**

The cluster model was included in this analysis as a model lying at the intersection of advanced data mining techniques and simpler statistical approaches. The significant underperformance as compared to benchmark methods is somewhat surprising, considering the promising results obtained in the original implementation (Banik, 2022) and the fact that it utilises the same exponential smoothing technique which produces the lowest median error overall in the current analysis. It appears that the issue is related to two of the obtained clusters, and the representative forecasts created at the cluster level. The cluster model's reduced accuracy overall could be attributed to the fact that cluster one and two consisted of a smaller number of time-series, and of time series with relatively large heterogeneity, resulting in representative forecasts that may not be particularly representative for the cluster. Given the more comparable performance in the more homogeneous clusters three and four, and the fact that there is a global pattern of general increase in life expectancy across the world, we may consider whether these results indicate that the current dataset is not fit for clustering. Clustering is helpful when there are different spatio-temporal characteristics amongst families of time-series across a dataset, likely arising based on similar underlying causal mechanisms (for instance, if there was a "great Asian war", we may see a benefit of clustering Asian countries together). Here, we seem to have but one general, global pattern, which is violated by single series for local reasons (such as the Syrian war). Thus, in cluster one and two, the model may attempt to identify common patterns amongst

time-series that exhibit different trajectories than the general pattern, but are not actually related. It is also worth noting that we did not explore whether using a different method for fitting the representative forecast may have improved performance in these clusters. Especially when there are significant differences between clusters, it is necessary to select the most appropriate method for a given cluster rather than simply fitting the same method across all clusters as is done here. To conclude, we do not find this approach suitable for the current forecasting task. However, we recognize that with careful configuration, this method may be useful for forecasting a large quantity of time-series of higher heterogeneity, especially when forecasters wish to reduce the load related to time consumption and computational power as compared to local methods, or when employing a simpler and more transparent methodology as compared to deep- or machine learning models.

Transformer model **(Eva)**

In the present analysis, we find an overall accurate forecasting ability of the Transformer model for multiple univariate time-series. However, the accuracy does not exceed the benchmark methods, which produce accurate forecasts as well. Thus, there is little room for improvement as compared to benchmark results, and it is therefore difficult to discuss how beneficial it is to implement a complex model such as the Transformer for the task of fitting multiple univariate time-series in general. Nonetheless, as presented, we do observe that the Transformer performs better in odd cases, especially when there are sudden events immediately before the forecasting horizon. This indicates that the ability of the model to learn from multiple time-series has had an effect upon forecasting accuracy in cases that are difficult for the benchmark methods. However, there are important considerations and limitations to take into account when implementing a deep learning architecture like the Transformer and when interpreting the results. We have already addressed some of the common issues, including the fact that for the model to produce sensible forecasts, it needs a relatively large amount of data for training. Here, it appears that the model received a sufficient amount of time-series to produce better forecasts for odd cases than benchmark methods. However, it is important to consider that there might be a relatively high risk that the model is overfitting the data, due to the complex structure of the architecture (Flovik, 2023). A general risk is also that the model may infer irrelevant correlations between observations due to the complex attention structure (Roger & Rasul, 2022). Addressing these

issues is difficult, as this modelling approach is highly intransparent compared to simpler methods. The mechanisms underlying e.g. exponential smoothing are straightforward to understand, and the reasons behind their produced forecast trajectories are therefore also easy to decompose. In contrast, it is difficult to assess why a model such as the Transformer produces a certain forecast, making it challenging to adapt the model accordingly. This is related to another point which we have previously touched upon, namely that careful configuration of the hyperparameters of the model is important to produce sensible forecasts (as well as to avoid overfitting). Hyperparameter tuning is not a straight-forward task, and even when performed, the influence of various hyperparameters is hard to interpret. A general point to make when considering the use of the Transformer and similar architectures for time-series forecasting is that many time-series forecasting tasks (including the current) are essentially cases of linear problems, whilst e.g. the Transformer is developed to overcome issues related to non-linearity (Zeng et al., 2022). Thus, for time-series with strong linear tendencies as the current, it may generally be more appropriate to apply linear methods, of which even simple techniques have also been found to outperform different versions of Transformer models on various forecasting tasks (Zeng et al., 2022). However, given time-series with complex, non-linear relationships between observations, and when patterns present over a long-range time axis are informative, deep learning does improve forecasting ability as compared to linear methods, as it was for instance demonstrated in the M5 competition (Makridakis et al., 2022). In the case of forecasting multiple time-series, it is also a considerable strength of deep learning architectures that they are able to apply cross-learning when producing the individual forecasts, which may be what we see the benefit of in the cases of hard-to-predict series. Further, in the context of applicability in e.g. a business context, it is also of remarkable use that training a global model eventually reduces the workload related to producing local forecasts, especially as it can be used in transfer learning for novel time-series.

Conclusion

In conclusion, the current work finds that a global Transformer model performs well at forecasting multiple, univariate time-series, even without extensive time-series specific modifications to the original NLP-architecture. However, we find comparable performance using simple, local benchmark methods, even with slightly higher overall precision using Holt's

exponential smoothing. Only in a few, specific cases where time-series deviate from the overall linear trend (mostly when there is a sudden decrease in estimates shortly before the forecasting horizon) does the Transformer model perform significantly better. This points toward the advantage of cross-learning in the global approach. We do not obtain satisfactory results using the cluster model approach, likely due to a high homogeneity across all time-series with local outliers rather than truly distinct groups of time-series across the dataset. Practically, it would be beneficial for the field to establish a standard, functioning pipeline to perform a more in-depth comparison of forecasting ability on individual time-series when forecasting many time-series in unison. For future empirical investigations of methods for multiple time-series, it would be illuminating to apply a range of diverse datasets for the task, as well as testing different modifications of the Transformer architecture and comparing the performance of the Transformer against other deep learning models using the same data. Additionally, it may be interesting to consider varying forecasting horizons, to establish how well different approaches perform on short- versus long term forecasting tasks.

Literature

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework* (arXiv:1907.10902). arXiv. https://doi.org/10.48550/arXiv.1907.10902

Assimakopoulos, V., & Nikolopoulos, K. (2000). The theta model: A decomposition approach to forecasting. *International Journal of Forecasting*, *16*(4), 521–530. https://doi.org/10.1016/S0169-2070(00)00066-2**.**

Banik, P. (2022, March 30). Fitting a univariate time series model to a cluster of series. *Walmart Global Tech Blog*. https://medium.com/walmartglobaltech/fitting-a-univariate-time-series-model-to-a-cluster-of-series-25cefd73f545

Dabbura, I. (2022, September 27). *K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks*. Medium. https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

Flovik, V. (2023, January 25). *How (not) to use Machine Learning for time series forecasting: Avoiding the pitfalls*. Medium. https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-19f9d7adf424

Gamboa, J. C. B. (2017). *Deep Learning for Time-Series Analysis* (arXiv:1701.01887). arXiv. http://arxiv.org/abs/1701.01887

Godahewa, R., Bergmeir, C., Webb, G. I., Hyndman, R. J., & Montero-Manso, P. (n.d.). *Monash Time Series Forecasting Archive*.

Han, Z., Zhao, J., Leung, H., Ma, K. F., & Wang, W. (2021). A Review of Deep Learning Models for Time Series Prediction. *IEEE Sensors Journal*, *21*(6), 7833–7848. https://doi.org/10.1109/JSEN.2019.2923982

Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., Pottelbergh, T. V., Pasieka, M., Skrodzki, A., Huguenin, N., Dumonal, M., Kościsz, J., Bader, D., Gusset, F., Benheddi, M., Williamson, C., Kosinski, M., Petrik, M., & Grosch, G. (2022). Darts: User-Friendly Modern Machine Learning for Time Series.

Journal of Machine Learning Research, *23*(124), 1–6.

Hyndman, R. J. (2021). *Forecasting: Principles and Practice (3rd ed)*. https://otexts.com/fpp3/

Kamsetty, A. (2020, October 6). Hyperparameter Optimization for 🤗Transformers: A guide. *Distributed Computing with Ray*. https://medium.com/distributed-computing-with-ray/hyperparameter-optimization-for-transformers-a-guide-c4e32c6c989b

Lewinson, E. (2020, November 1). *Choosing the correct error metric: MAPE vs. sMAPE*. Medium. https://towardsdatascience.com/choosing-the-correct-error-metric-mape-vs-smape-5328dec53fac

Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, *37*(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen, E., & Winkler, R. (1982). The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, *1*(2), 111–153. https://doi.org/10.1002/for.3980010202

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, *38*(4), 1346–1364. https://doi.org/10.1016/j.ijforecast.2021.11.013

Martins, C. (2023, March 21). *Understanding Bisecting K-Means: Hands-On with SciKit-Learn*. Medium. https://code.likeagirl.io/understanding-bisecting-k-means-hands-on-with-scikit-learn-550e69619db5

Naik, N. N., Chandrasekaran, K., Venkatesan, M., & Prabhavathy, P. (2022). Deep Learning-Based Prediction, Classification, Clustering Models for Time Series Analysis: A Systematic Review. In V. Goar, M. Kuri, R. Kumar, & T. Senjyu (Eds.), *Advances in Information Communication Technology and Computing* (pp. 377–390). Springer Nature. https://doi.org/10.1007/978-981-19-0619-0_34

Onnen, H. (2022, May 16). *Transformer Unleashed: Deep Forecasting of Multivariate Time Series in Python*. Medium. https://towardsdatascience.com/transformer-unleashed-deep-forecasting-of-multivariate-time-series-in-python-9ca729dac019

Orlandi, G. (2022, July 28). *Global deep learning for joint time series forecasting*. Medium. https://towardsdatascience.com/global-deep-learning-for-joint-time-series-forecasting-4b03bef42321

Özateş, M. N. (2022, November 23). Transformer Models Hyperparameter Optimization With the Optuna. *CARBON CONSULTING*. https://medium.com/carbon-consulting/transformer-models-hyperparameter-optimization-with-the-optuna-299e185044a8

Roger, N., & Rasul, K. (n.d.). *Probabilistic Time Series Forecasting with 🤗 Transformers*. Retrieved May 31, 2023, from https://huggingface.co/blog/time-series-transformers

Roser, M., Ortiz-Ospina, E., & Ritchie, H. (2013). Life Expectancy. *Our World in Data*. https://ourworldindata.org/life-expectancy

Szabłowski, B. (2023, April 26). *Forecast Multiple Time Series Like a Master*. Medium. https://towardsdatascience.com/forecast-multiple-time-series-like-a-master-1579a2b6f18d

Tang, B., & Matteson, D. S. (2021). Probabilistic Transformer For Time Series Analysis. *Advances in Neural Information Processing Systems*, *34*, 23592–23608. https://proceedings.neurips.cc/paper/2021/hash/c68bd9055776bf38d8fc43c0ed283678-Abstract.html

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., & Sun, L. (2023). *Transformers in Time Series: A Survey* (arXiv:2202.07125). arXiv. http://arxiv.org/abs/2202.07125

Yeshchenko, A., Di Ciccio, C., Mendling, J., & Polyvyanyy, A. (2019). *Comprehensive Process Drift Detection with Visual Analytics* (arXiv:1907.06386). arXiv. https://doi.org/10.48550/arXiv.1907.06386

Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2022). *Are Transformers Effective for Time Series Forecasting?* (arXiv:2205.13504; Version 2). arXiv. http://arxiv.org/abs/2205.13504