Drilling down into Chronos TS

Alejandra Solarte Usctegui
Data Science and Engineering
Politecnico di Torino
Turin, Italy
s321944@studenti.polito.it

Alejandro Mesa Gmez
Data Science and Engineering
Politecnico di Torino
Turin, Italy
s306638@studenti.polito.it

Mateo Rivera Monsalve
Data Science and Engineering
Politecnico di Torino
Turin, Italy
s320923@studenti.polito.it

This work builds on CHRONOS, a pre-trained probabilistic framework that treats univariate time series as language. We extend CHRONOS in two directions. First, to handle covariates we compress multivariate inputs with Principal Component Analysis and with a transformer auto-encoder, feed the reduced series to CHRONOS, and reconstruct the full signal, achieving accurate hourly air-quality forecasts. Second, we adopt LoRAan efficient parameter fine-tuning methodto specialise the 46 M-parameter Chronos-T5-Small while updating only 0.6 % of its weights, matching full fine-tuning on eight public datasets. Finally, a systematic context-versus-horizon study shows that data aggregation tightens WQL but worsens MASE, guiding optimal split choices.

***Index Terms*—Time Series Forecasting, Chronos Framework, Low-Rank Adaptation, Covariates Time Series, Forecast Horizon Sensitivity, WQL, MASE, context length, prediction length.**

The source code associated with this report is available on GitHub: *https://github.com/cosmolejo/DeepNLP_Forecasting*

I. Problem statements

A. Specializing CHRONOS with LoRA: Parameter-Efficient Fine-Tuning in time series

Chronos transforms time series into token sequences through scaling and quantization, enabling their processing by large language models (LLMs) without the need to modify their architecture. This allows the direct use of LLMs, such as those based on the T5 architecture, to model complex patterns in temporal data [1].

Given the high computational cost of training LLMs from scratch [2], a common strategy is to specialize pretrained models through fine-tuning, which reduces data and computation requirements. This technique allows for the reuse of previously learned knowledge and is effective in adapting models to specific tasks with less labeled data [3]. However, even the smallest Chronos model (Chronos-T5 Mini) contains 20 million parameters [1], making it costly to adjust all of them.

Therefore, parameter-efficient fine-tuning (PEFT) techniques are used. In particular, LoRA (Low-Rank Adaptation)proposed by Hu et al. [4] and mentioned in the Chronos article as a viable alternative for reducing the cost of fine-tuningintroduces trainable low-rank adapters in certain layers of the model, significantly decreasing the number of updated parameters [1].

Inspired by the experiment described in the original article (Fine-tuning section, p. 13) [1], we replicated the fine-tuning of the Chronos-T5 Small model (46 million parameters) on new sets of unseen time series. Unlike the original approach, we used LoRA to perform the fine-tuning, and we compared the performance of the fine-tuned model with the base model. The goal is to verify whether LoRA allows for performance improvements with lower computational cost.

B. Extending CHRONOS' capabilities for multivariate forecasting

CHRONOS focuses primarily on univariate prediction, where observations are scalar (i.e., $x_i \in \mathbb{R} \forall i$). Real-world forecasting tasks, most of the time, involve predicting multivariate time series or incorporating exogenous information (covariates), whose influence is crucial for precise prediction. The wide variability in the number of covariates or multivariate dimensions between different tasks presents a significant challenge for training a generalist model. The current Chronos framework does not offer a direct and native solution for multivariate forecasting; therefore, to overcome this limitation and extend our model's capabilities to multivariate domains, this project proposes an innovative methodology based on dimensionality reduction:

- Dimensionality Reduction with PCA and Prediction with Chronos: The first strategy involves applying Principal Component Analysis (PCA) to reduce the dimensionality of multivariate time series to a single univariate series to be the input for Chronos, which will generate future predictions. Afterward, we perform a reconstruction of the original multivariate time series using the predicted data.
- Autoencoder as an Alternative Approach: As a robust alternative to PCA, we analyse the viability of an Autoencoder for dimensionality reduction. The Autoencoder will learn a compressed latent representation of the multivariate time series, and then, we will use this latent space as input for the model. The process of prediction and following reconstruction of the multivariate time series will be equivalent to that used with PCA.
- Evaluation and Comparison with Existing Multivariate Methods: To validate the suitability and performance of our proposal, we will compare our experiments with the performance of other established and robust methods in

multivariate time series prediction. This comparison will allow us to estimate the effectiveness of our approach and determine whether our proposal is a reliable method to perform multivariate forecasting.

C. Analysis of the behaviour of the errors and granularity of the datasets

The selection of the amount of data corresponding to the context length and prediction horizon for the use of the Chronos model is a fundamental part of the modeling given to the variation of the errors. This improves or diminishes the accuracy of the predictions depending on which configuration of the split is chosen to do it.

Also, it is important to understand that the split of the data depends on the level of granularity of the dataset and how strong are the changes in its behavior. For instance, according to the configurations tested, the Weighted Quantile Loss (WQL) could take values between 0.05 to 0.35, when the forecast window is long or the training slice is too small. Because of this, the improvement of the accuracy of the predictions made by Chronos can increase.

Furthermore, implementing an appropriate selection of the split between train and test would yield benefits such as the adequate estimation of the training set according to how many data is going to be need to collect. Reducing with this the cost to collect and process unnecessary data, improving the efficiency of the process

II. Methodology

A. LoRA's implementation

1) Overview of LoRA's architecture: To enable fine-tuning of large language models (LLMs) with minimal computational cost, Hu et al. (2021) [4] proposed the *Low-Rank Adaptation* (LoRA) technique. Rather than updating all weights of a pre-trained model, LoRA introduces trainable low-rank matrices into targeted componentstypically the linear projections within attention and feed-forward layers. These additional parameters allow the model to adapt effectively while keeping the original weights frozen.

LoRA has demonstrated competitive performance across various NLP tasks, significantly lowering the resource requirements of model adaptation.

2) LoRA's modules: The LoRA mechanism operates through a modular integration strategy that enables efficient adaptation, as illustrated in Figure 3 in the LoRA extension appendix:

- **Low-rank adapters**: In selected layers, a parallel pathway consisting of two trainable matrices is added:
  - $A \in \mathbb{R}^{d\times r}$: projects the input x to a lower-dimensional space.
  - $B \in \mathbb{R}^{r\times d}$: maps it back to the original space.

  Initialized as:

$$A \sim \mathcal{N}(0, \sigma^2), \quad B = 0$$

  The resulting transformation becomes:

$$W_{\text{LoRA}}(x) = W(x) + BAx$$

  where W are the frozen base weights.
- **Parameter freezing**: The original weights remain unchanged throughout training. This preserves the models foundational representations and prevents overfitting, particularly beneficial when labeled data is scarce.
- **Adapter-only training**: Only the added matrices A and B are updated. Optimization is carried out using the cross-entropy loss function as proposed in the Chronos framework:

$$\ell(\theta) = -\sum_{h=1}^{H+1}\sum_{i=1}^{|\mathcal{V}_{\text{ts}}|} \mathbf{1}_{(z_{C+h+1}=i)} \log p_\theta(z_{C+h+1} = i \mid \mathbf{z}_{1:C+h})$$

  This loss is averaged over all samples in a batch during training. It follows the same principle as language modeling in NLP, where the model learns to predict the next token given a context, but applied here to quantized time series data.
- **Inference integration**: During prediction, the outputs of each modified layer reflect the sum of pre-trained knowledge and low-rank adjustments, delivering adapted behavior with minimal overhead.

The modularity and efficiency of LoRA enable its practical use in constrained environments, extending the adaptability of large models without compromising their pre-trained advantages.

B. Multivariate Extension

The solution pipeline comprises three stages:

1) **Dimensionality Reduction**:

$$\text{PCA:} \quad X \xrightarrow{\text{PCA}} z_{1:T}$$
$$\text{Autoencoder (AE):} \quad X \xrightarrow{\text{Encoder}} z_{1:T}$$

   where $z_{1:T}$ is a low-dimensional (univariate) representation.
2) **Forecasting**:

$$z_{1:T} \xrightarrow{\text{Chronos}} \hat{z}_{T+1:T+H}$$

3) **Reconstruction**:

$$\text{PCA:} \quad \hat{z}_{T+1:T+H} \xrightarrow{\text{PCA}^{-1}} \hat{X}_{T+1:T+H}$$
$$\text{AE:} \quad \hat{z}_{T+1:T+H} \xrightarrow{\text{Decoder}} \hat{X}_{T+1:T+H}$$

Validation Strategy

We benchmark against native multivariate models to evaluate:

- Autogluon multivariate implementation for Chronos
- XGBoost

1) PCA: Principal Component Analysis (PCA) [5] is a widely used dimensionality reduction technique for simplifying complex data sets. Its main objective is to transform a set of possibly correlated variables into a new set of uncorrelated variables, called principal components. The first principal component captures the greatest amount of variance in the data, the second component captures the greatest remaining variance orthogonal to the first, and so on. This process allows the most important patterns in the data to be identified, projecting them into a lower-dimensional space while retaining as much of the original information (variance) as possible.

Let $X_{\text{train}} \in \mathbb{R}^{T \times d}$ be the original multivariate time series and $z_{\text{train}} = \text{PCA}(X_{\text{train}}) \in \mathbb{R}^T$ its first principal component. Given Chronos' forecast $\hat{z}_{\text{test}} \in \mathbb{R}^H$ for future steps, we construct the extended series $z_{\text{full}} = [z_{\text{train}}; \hat{z}_{\text{test}}] \in \mathbb{R}^{T+H}$. Since Chronos preserves the statistical properties of the training data (as demonstrated in §5.5 of [1]), $\hat{z}_{\text{test}}$ follows the same distribution as z_{train}. The inverse PCA operation:

$$\hat{X}_{\text{test}} = \text{PCA}^{-1}(z_{\text{full}})[T+1 : T+H, :] \in \mathbb{R}^{H \times d} \quad (1)$$

yields valid reconstructions because: (1) the principal components remain invariant when new data conforms to the original distribution [5], and (2) the temporal dependencies are preserved through Chronos' autoregressive generation (§3.3 of [1]).

2) Autoencoder: An alternative approach employs an *Autoencoder* (AE) with a unidimensional latent space $z_t \in \mathbb{R}$. The AE architecture (detailed in Appendix V-B2) consists of:

- An **Encoder** $E : \mathbb{R}^d \to \mathbb{R}$ that compresses multivariate inputs $x_t \in \mathbb{R}^d$ to latent representations $z_t = E(x_t)$
- A **Decoder** $D : \mathbb{R} \to \mathbb{R}^d$ that reconstructs $\hat{x}_t = D(z_t)$

Analogous to the PCA pipeline, we:

1) Encode the training series: $z_{\text{train}} = E(X_{\text{train}}) \in \mathbb{R}^T$
2) Forecast future latent states with Chronos: $\hat{z}_{\text{test}} = \text{Chronos}(z_{\text{train}}) \in \mathbb{R}^H$
3) Concatenate and decode: $\hat{X}_{\text{test}} = D([z_{\text{train}}; \hat{z}_{\text{test}}])[T+1 : T+H]$

This reconstruction is valid because:

- The AE's latent space preserves temporal dynamics of the original data and the predicted data will follow the same distribution.
- The decoder D generalizes to predicted latents $\hat{z}_{\text{test}}$ when they adhere to the training distribution [6]

C. Errors Analysis

The study trains, predicts and evaluates three timeseries-Finance, Industry and Demographyto examine how small adjustments in context length and prediction length influence the two main metrics used in the original Chronos paper WQL and MASE.

These two metrics are used with AutoGluon where the Weighted Quantile Loss (WQL) is used to measure the probabilistic model according to if the prediction is in the correct side of the truth value and the Mean Absolute Seasonal Error MASE is used to measure if the prediction is better, equal or worst to the seasonal naive rule. This metrics are multiplied by -1 so the bigger number (the closest to 0) is the best one.

AutoGluon˙Project

For every temporal granularity, it is first split the 20% of observations as the base forecast horizon. From the remaining history it is build a 5×5 grid: the context factor retains $100\%, 95\%, 75\%, 50\%$ or 25% of the original training window, while the prediciton length factor asks the model to predict those same five percentages of that base horizon. This factorial design produces 25 splits per series and granularity, allowing to analyze the effects of shrinking history and stretching horizon.

Chronos evaluator requires the full context plus at least one extra step. Then, it is therefore assemble a composite test slide that joint the reduced history to the required forecast window, then reorder its last rows to match the predictors index so that Chronos strict ordering check passes. AutoGluon returns sign-flipped MASE and WQL scores, which are stored along with the exact train and test sizes.

With respect to the Finance Dataset, high volatility makes errors depend mainly on test size where the shorter the horizon, the better the daily and monthly scores; in the yearly view, performance improves only when the training window is maximised.

For the Industry Dataset, with lower variability, error still is highly correlated on test size. Daily MASE is worst when both train and horizon are large and best when the prediciton length is $\approx 1\%$ of the series. Monthly MASE gets worst overall but the best part is when context is close to the present, while WQL is always better than the daily worst case and improves as train and test windows converge.

For the Demography dataset, errors magnitude correlates most with test size as it is possible to observe in 13 and 14 the best MASE error occurs when the context is close to the current date as it is possible to see in 15, 18 and 19. The prediction lenth with the minimal configuration always gets the best results as **??**, **??**, **??** and 20 .

These patterns follow directly from the metrics: WQL rewards the full prediction interval, so monthly aggregation stripping daily noise lets the model wrap tighter, better-centred bands, lowering WQL. MASE, judging only the point forecast, rises because aggregation amplifies mean bias while the seasonal naive baseline strengthens; point estimates drift farther from that baseline even as the WQL measured intervals look better.

III. Experiments

A. Chronos with LoRA

As discussed throughout this report, the adaptation of Chronos-T5 (Small) using LoRA aims to match the performance of fully fine-tuned models, where all parameters are updated. To enable a fair comparison, LoRA ideally should be applied across the complete set of 27 datasets included in Benchmark II, as described in the original Chronos paper [1].

However, due to computational constraints, we selected a representative and diverse subset of 8 datasets. These cover multiple real-world categories including banking, retail, healthcare, finance, transport, and economics ensuring heterogeneity in structure, scale, and forecasting horizons.

Forecast horizons (H) are defined per dataset as shown in Table I of the LoRA extension appendix.

All datasets used in this subset are publicly available via the official Chronos repository on Hugging Face:
https://huggingface.co/datasets/autogluon/chronos_datasets

Each dataset underwent a time series-specific train-test split, where the training set excludes the last H observations, which were held out for evaluation.

Chronos-T5 (Small) was fine-tuned using LoRA. Hyperparameters were defined in the configuration file LoRA/scripts/training/configs/chronos-t5-small-lora.yaml

The adaptation used a low-rank configuration with $r = 8$ and $\alpha = 16$, targeting the Query (q) and Value (v) projection modules within the models attention mechanism. Out of a total of $46{,}449{,}152$ parameters, only $294{,}912$ were trainable under LoRArepresenting approximately 0.6349% of the model's full capacity.

After training, the adapted model was evaluated on all Benchmark II datasets. Results were compared with the fully fine-tuned Chronos-T5 (Small) model and its non-adapted version.

We adopt the same metrics used in the Chronos paper:

- Mean Absolute Scaled Error (MASE)
- Weighted Quantile Loss (WQL)

In both cases, lower values indicate better forecasting performance.

The fine-tuning of Chronos-T5 (Small) using LoRA required approximately 4 hours of computation time on the described hardware. The evaluation phase, which involved forecasting across all 27 datasets from Benchmark II, took an additional 7 hours.

Figure 4 (LoRA extension appendix) presents the aggregated relative scores for Chronos-T5 (Small) across the full Benchmark II, evaluated under three configurations: Zero Shot (pre-trained, non-adapted), LoRA PEFT with rank $r = 8$, and full fine-tuning of all model parameters.

On the MASE benchmark, LoRA achieved a score of 0.839, slightly outperforming Zero Shot (0.841) and approaching the performance of the fully fine-tuned model (0.760). Similarly, on the WQL benchmark, LoRA reached 0.656, improving upon Zero Shot (0.667) and getting closer to the fine-tuned baseline (0.597).

These results suggest that LoRA-based adaptation yields meaningful gains over Zero Shot performance, while requiring significantly fewer trainable parameters.

The experimental evidence confirms that LoRA adaptation can improve the predictive capabilities of Chronos-T5 (Small) without the computational cost of full fine-tuning. While the fine-tuned model still outperforms LoRA across both metrics, the performance gap is narrow, especially considering that LoRA updates only 0.63% of the models parameters.

This efficiency-performance trade-off reinforces the viability of LoRA as a low-cost alternative for time series forecasting tasks, particularly in resource-constrained environments. Additionally, the fact that LoRA consistently improves upon the Zero Shot baseline across both MASE and WQL metrics indicates that even limited adaptation introduces meaningful specialization to the model.

B. Multivariate Extension

For these experiments, we employed a multivariate time series dataset of moderate complexity, as data quality plays a crucial role when using classical dimensionality reduction techniques. The dataset consists of hourly air quality measurements from an array of 5 metal oxide chemical sensors deployed in an urban environment. Specifically, it contains 9,358 instances of sensor responses recorded over one year (March 2004 to February 2005) in a polluted area of an Italian city, with co-located reference measurements for CO, NMHC, Benzene, NOx, and NO_2 concentrations [7]. For the purpose of our experiments we will use the feature 'C6H6(GT)' as the target variable for prediction. Some preprocessing was done in order to deal with the missing data.

1) PCA: To handle the exogenous variables in our model, we conducted several experiments with dimensionality reduction techniques. Initially, we tried a direct reduction of all variables (including the target variable) to a single principal component. However, this approach resulted in a significant loss of information about the target variable, negatively affecting the predictive power of the model.

In a second approach, we implemented a two-step strategy:

1) Separate reduction of exogenous covariates using PCA
2) Application of PCA on the resulting components together with the target variable to obtain a single final component

This approach proved to be significantly more effective. The prediction using this single component adequately preserved the seasonality patterns present in the original data, as evidenced in figure 5. When comparing the predictions with the actual values (Figure 6), we observe that the model generates values very close to the theoretical ones, with minimal errors at most time points.

A particularly encouraging result was the reconstruction capability when applying the inverse PCA transformation. As shown in Figure 7, the reconstructed values of the target variable maintain high fidelity to the original data, preserving both magnitude and temporal dynamics.

These results suggest that this staged dimensional reduction technique may be a promising strategy for extending Chronos' capabilities in handling multivariate time series, allowing for:

- Preservation of relevant information from the target variable
- Efficient handling of exogenous covariates
- Maintenance of key temporal patterns
- Accurate reconstruction of the original variables

2) AutoEncoder: We made the same process using the Autoencoder, however getting a little bit unexpected results. In Figure 8, it can be seen that the technique preserves the seasonal patterns of the original data. However, when comparing the predictions with the theoretical values (Figure 9), the model shows lower accuracy than that achieved with PCA, particularly in the magnitude of the predicted values. The reconstruction of the target variable also failed to achieve the same level of fit as the previous method, generating a signal with higher magnitude with respect to the theoretical, getting approximately 1 point of separation in the maximum distance point. However, we can observe that the Autoencoder did manage to preserve the temporal patterns of the series, therefore, we can say that the approach remains valid for applications where temporal dynamics are more relevant than absolute values.

3) Comparison wint classic multivariate techniques : Since the initial publication of the article describing the base model, some adjustments have been proposed by third parties to extend its capabilities. A notable attempt at multivariate generalization was developed by the AutoGluon project **AutoGluon˙Project** where they implement a tabular model to generate Chronos inputs. However, as shown in Figure 11, this approach has a significantly higher margin of error compared to our dimensional reduction proposals. When compared to XGBoost, a classic method known to have good quality predictions for multivariate series, the results in Figure 12 show that the it achieves perfect accuracy on the test dataset. This results where expected, as XGBoost has good performance for small datasets such as the one used in this tests. However, in scenarios with more extensive and complex time series, Chronos could outperform this model thanks to its attention mechanisms that let it keep track complex seasonal patterns in the data that other models would fail to learn or will get directly overfitted.

C. Errors Analysis

It was used the **M4 Daily** subset given to the fact that it was also used in the original paper of Chronos and then isolated three long univariate seriesone each from the *Finance*, *Industry* and *Demographic* categories (ids `T002046`, `T002014`, `T001611`). Each series was converted to a long-format DataFrame with columns `item_id`, `timestamp`, `target` in order to set the required format by Chronos. For granularity experiments it was resampled the raw daily data to monthly (`MS`) and yearly (`YS`) frequencies. Timestamps were normalised to midnight to avoid index-mismatch errors and the index was sorted after every slice.

Additionally, each split was train to Chronos using the lightweight `bolt_small` preset, so hyper-parameters and network capacity remained constant across runs. Two models were trained per splitone optimised for *Weighted Quantile Loss* (WQL) and the other for *Mean Absolute Scaled Error* (MASE)thereby measuring distributional quality and point accuracy under identical data conditions.

Also, Chronos evaluator requires the full context plus at least one extra step beyond the forecast horizon. Because of this, it was rebuilt a composite *test slide* that combined the trimmed history with the appropriate forecast window. To satisfy Chronos strict index-ordering check, the final forecasted rows of this slide were explicitly reordered to match the predictors own index before invoking `evaluate`. AutoGluon returns sign-flipped scores (larger = better); these values, along with the exact train and test sizes, were stored in a cumulative results table.

Finally, all the set of runs were finished in approximately 20 minutes.

IV. Conclusions

One of the most notable achievements of this project is that the Chronos framework now supports LoRA-based adaptation without becoming LoRA-dependent. Users may continue to leverage all features of the original Chronos repository as designed, while now having the added option to perform parameter-efficient fine-tuning using Hugging Face's `peft` library with customizable configurations.

Implementing LoRA fine-tuning during summer posed thermal constraints on personal hardware. Training sessions pushed the CPU and GPU temperatures to peaks of 92°C, requiring real-time monitoring to avoid system throttling or failure.

Designing modular systems allows optional extensions (like LoRA) without disrupting core functionality.

The dimensionality reduction method, particularly PCA, proves to be an effective strategy for adapting Chronos to multivariate problems. Although it requires training specific models for each signalwhich partially reduces its generalist capacitythe process is simple and feasible even on limited hardware, facilitating its practical implementation.

PCA stands out for its accuracy and, unlike other approaches, offers transparency: as an explainable model, it allows us to understand how much each variable contributes to the final result. This not only improves interpretability but also helps optimize data preprocessing.

Although techniques such as AutoGluon or XGBoost have their advantages, the combination of PCA + Chronos achieves a balance between performance, efficiency, and clarity, being especially useful when prioritizing the capture of complex temporal patterns without sacrificing computational accessibility. This is a promising line of work to take Chronos beyond univariate series.

We can observe that the smaller the test size, the smaller is the error, where the test error should be approximately of 1%. For high granularity level of the data, the error can improve with the closer possible context. Also, when the granularity of the data is bigger like Yearly it is most relevant in the closest context.

Prediction-window length matters more than history length: very short horizons give the best scores.

References

[1] A. F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. Pineda Arango, S. Kapoor, J. Zschiegner, D. C. Maddix, M. W. Mahoney, K. Torkkola, A. Gordon Wilson, M. Bohlke-Schneider, and Y. Wang, "Chronos: Learning the language of time series," *Transactions on Machine Learning Research*, 2024, ISSN: 2835-8856. [Online]. Available: https://openreview.net/forum?id=gerNCVqqtR.

[2] A. W. Services. (2023). What is llm (large language model)? [Online]. Available: https://aws.amazon.com/what-is/large-language-model/.

[3] D. Bergmann, *What is fine-tuning?* Mar. 2024. [Online]. Available: https://www.ibm.com/think/topics/fine-tuning.

[4] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9.

[5] I. T. Jolliffe, *Principal component analysis*. Springer Science+Business Media, LLC, 2010.

[6] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *CoRR*, vol. abs/1906.02691, 2019. arXiv: 1906.02691. [Online]. Available: http://arxiv.org/abs/1906.02691.

[7] S. Vito, *Air quality*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C59K5F, 2008.

[8] J. Belveze, *Julesbelveze/time-series-autoencoder: Pytorch dual-attention lstm-autoencoder for multivariate time series*, Oct. 2024. [Online]. Available: https://github.com/JulesBelveze/time-series-autoencoder/tree/master?tab=readme-ov-file.

V. Appendix

A. Hardware and Software specializations

1) LoRA:

- **Hardware:** All experiments were conducted on a personal laptop: Predator Helios 300. The system is equipped with an NVIDIA GeForce GTX 1660Ti GPU (6GB VRAM), CUDA version 12.9, an Intel Core i7-9750 processor @ 2.60 GHz, and 16 GB of RAM.
- **Software and libraries:** Development was performed using Visual Studio Code and Python version 3.13.2. Required dependencies were installed via the command: `pip install --editable ".[training]"` from the Chronos GitHub repository. Key packages include:
  - `torch==2.7.0+cu128` PyTorch with CUDA support.
  - `datasets==2.21.0` and `transformers==4.51.3` Hugging Face ecosystem for tokenization and model APIs.
  - `gluonts==0.16.1` probabilistic time series forecasting utilities.
  - `numpy==1.26.4`, `pandas==2.2.3` numerical and data manipulation.
  - `peft==0.16.0` LoRA implementation used to adapt Chronos-T5.

2) Multivariate Extension: All experiments ran in a dedicated conda environment (python 3.12.10, GPU-enabled PyTorch 2.3) with chronos 0.6, autogluon.timeseries 1.3.1, pandas, matplotlib/seaborn, and Jupyter.

Hardware: MSI Thin GF63 12VF, 16 Gib RAM, Processor: 12th Gen Intel Core i7-12650H 16, GPU: NVIDIA GeForce RTX 4060, CUDA 12.7

3) Errors Analysis: All experiments ran in a dedicated conda environment (python 3.12.10, GPU-enabled PyTorch 2.3) with chronos 0.6, autogluon.timeseries 1.3.1, pandas, matplotlib/seaborn, and Jupyter. To ensure exact reproducibility it was froze the environment (pip freeze ¿ requirements.txt) and executed every notebook cell with a fixed random seed (torch.manual$_seed(123); rn.seed(123)$).

B. Description of the main models

1) Chronos:

- **Overview of Chronos' architecture**
  The general architecture of Chronos consists of three main stages: tokenization, training, and inference. As illustrated in Figure 1, in the tokenization phase, time series are transformed into discrete sequences of tokens through scaling and quantization processes. These tokens allow temporal information to be represented in a format that is understandable by language models.
  During training, Chronos models learn to predict future sequences using cross-entropy-based losses on the tokens. Finally, in the inference stage, autoregressively sampled token trajectories are generated, which are then descaled and dequantized to produce probabilistic prediction values of the series.

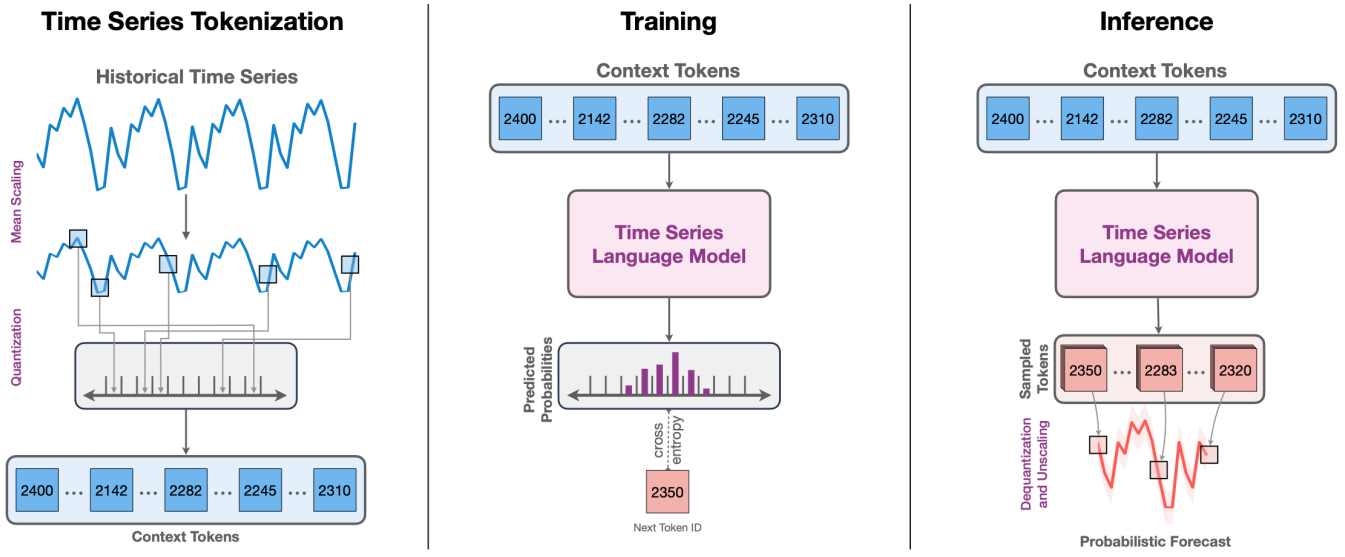

Fig. 1. Summary of the processing flow in Chronos: tokenization of time series, training through token prediction, and autoregressive inference and probabilistic decoding.

- **Chronos' modules**
  Chronos is structured into three fundamental modules that articulate the processing and modeling of time series:
  - **Temporal tokenizer**: Responsible for converting time series into discrete sequences of tokens. This process is performed through scaling and quantization, which allows numerical data to be represented in a vocabulary that the language model can handle. The tokens capture the original temporal dynamics of the series.
  - **Probabilistic language model**: Based on T5-type architectures, this module receives sequences of tokens and learns to predict the next sequence using cross-entropy loss. It can be of the encoder-decoder or decoder-only type, depending on the Chronos model configuration.
  - **Decoder and sampler**: In the inference phase, this module performs autoregressive token generation, sampling multiple future trajectories. It then converts the generated tokens into real values through an inverse process of downscaling and dequantization, allowing probabilistic predictions to be obtained about the evolution of the time series.

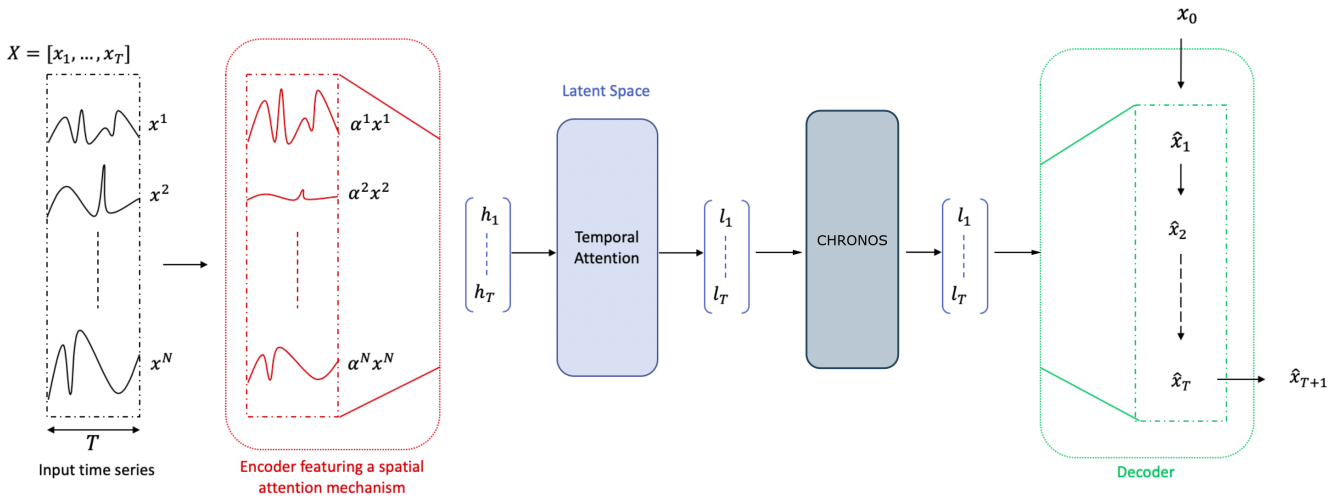

Fig. 2. Overall architecture of the autoencoder implementation, the latent space is used as input for CHRONOS and its output gets decoded to recover the original data

2) Autoencoder: The AutoEncForecast model ilustrated in Figure 2 combines encoder-decoder architecture with attention mechanisms for time series processing. The original architecture, used to implement the model comes from the project of Jules Belveze, publically available on GitHub [8] Key components:

- **Core Structure**
  - **Encoders**:
    - Basic: LSTM Linear(1D)
    - Attn: LSTM + Feature Attention Linear(1D) + Denoising
  - **Decoders**:
    - Basic: Linear(expansion) LSTM Linear
    - Attn: Linear(expansion) + Temporal Attention LSTM 2xLinear
  - **Main Model**:
    - Flexible encoder/decoder combinations
    - Latent space operations (encode/decode)
    - Automatic CPU/GPU handling
- **Key Features**
  - Dual attention mechanisms (input & temporal)
  - Fixed-length sequence processing (configurable)
  - Optional denoising during training
  - 1D latent space representation
- **Data Flow**
  1) **Encoding**: $X_{seq} \xrightarrow{\text{LSTM+Attn}} z \in \mathbb{R}^1$

2) **Decoding**: $z \xrightarrow{\text{Expand}} \hat{h} \xrightarrow{\text{LSTM+Attn}} \hat{y}$

Where z represents the latent encoding and $\hat{y}$ the reconstructed/predicted output.

C. LoRA extension appendix

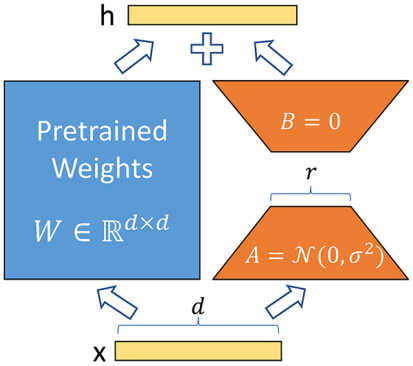

Fig. 3. Adaptation process with LoRA: the pre-trained weights W are combined with a low-rank projection given by the matrices A and B, where A is initialized randomly and B is set to zero. Adapted from Hu et al. (2022) [4].

TABLE I
SELECTED BENCHMARK II DATASETS USED IN LORA FINE-TUNING

Name	Num. series	Category	H (Prediction length)
CIF 2016	72	Banking	12
Car Parts	674	Retail	12
Hospital	767	Healthcare	12
FRED-MD	107	Economics	12
Exchange Rate	8	Finance	30
Covid Deaths	266	Healthcare	30
Traffic	862	Transport	24
M3 (Monthly)	1428	Various	18

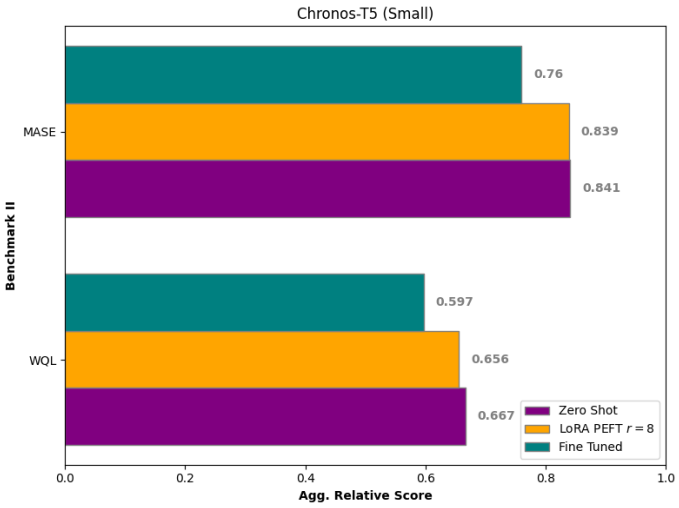

Fig. 4. Chronos-T5 (Small) results on Benchmark II: comparison across Zero Shot, LoRA (PEFT $r = 8$), and full fine-tuning using MASE and WQL metrics (lower is better).

D. Multivariate extension graphs

E. Errors Analysis

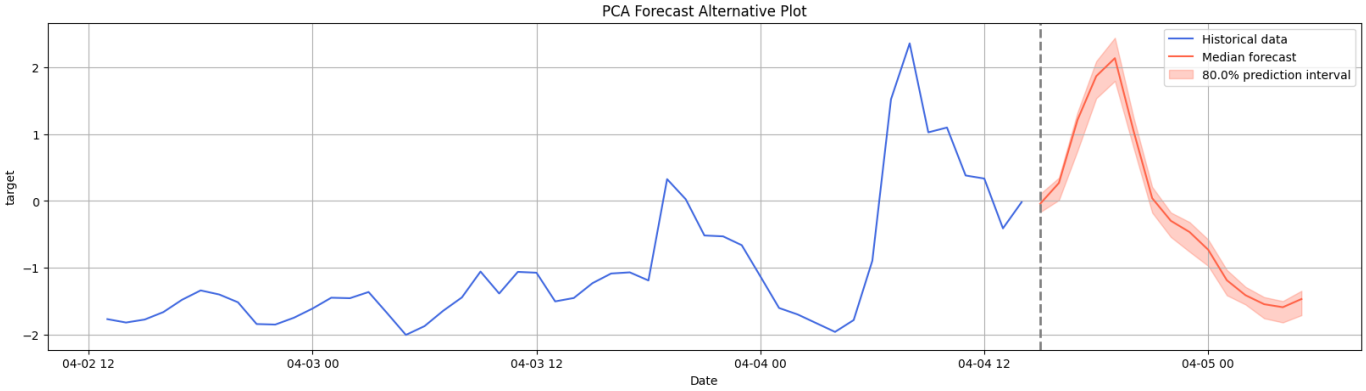

Fig. 5. CHRONOS forecasting using PCA dimensionality reduction

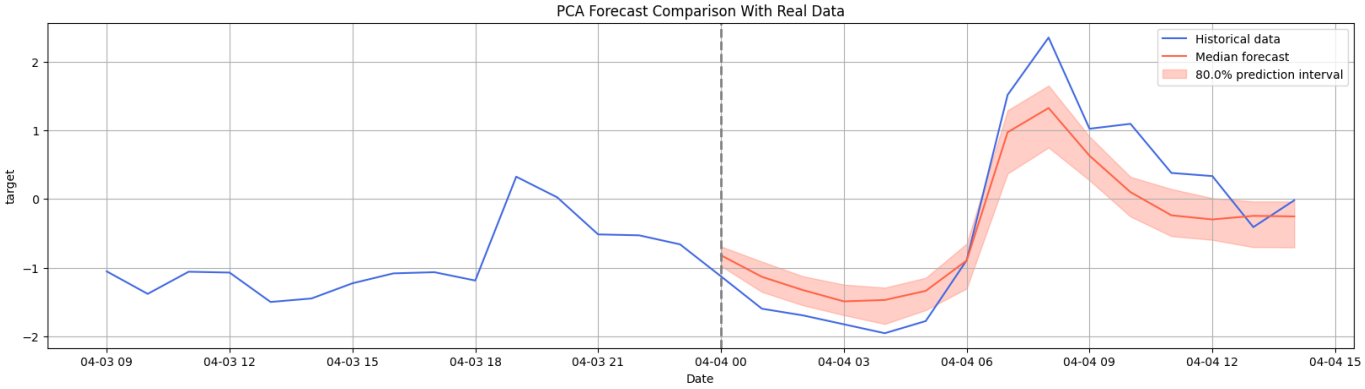

Fig. 6. PCA forecasting compared with ground truth

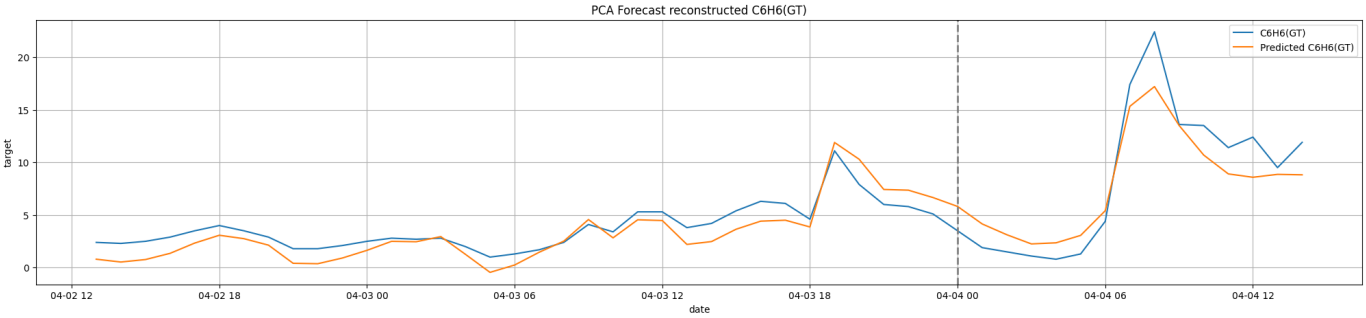

Fig. 7. PCA Reconstruction of target variable

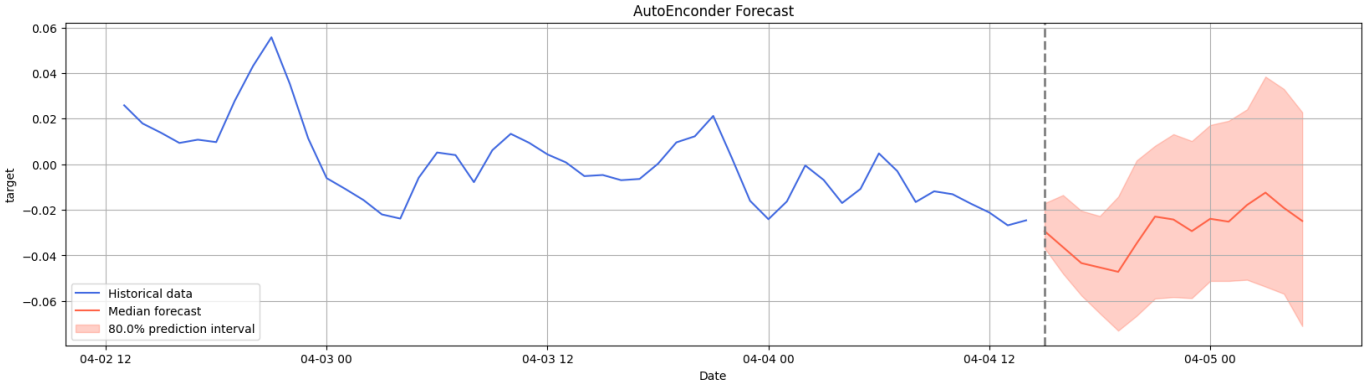

Fig. 8. CHRONOS forecasting using AutoEncoders dimensionality reduction

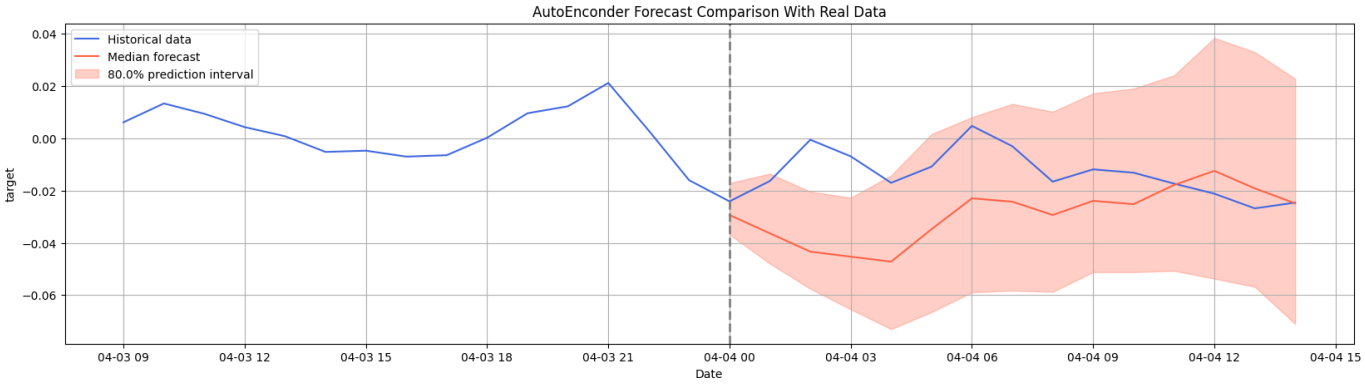

Fig. 9. AutoEncoder forecasting compared with ground truth

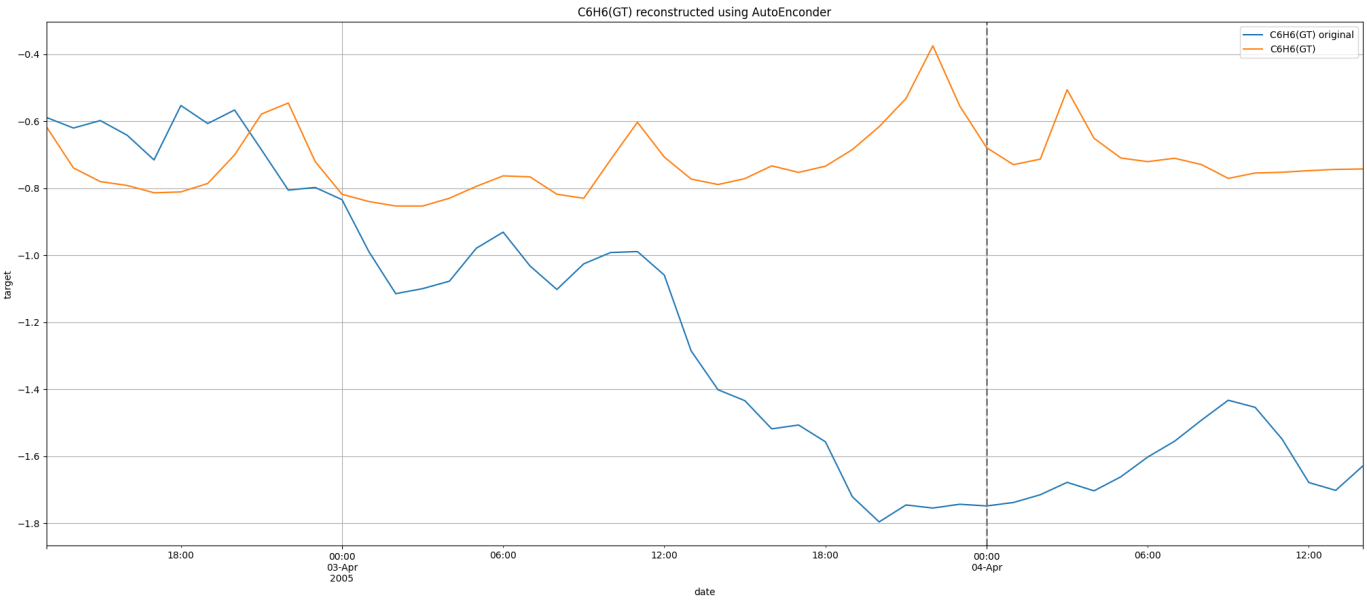

Fig. 10. AutoEncoder Reconstruction of target variable

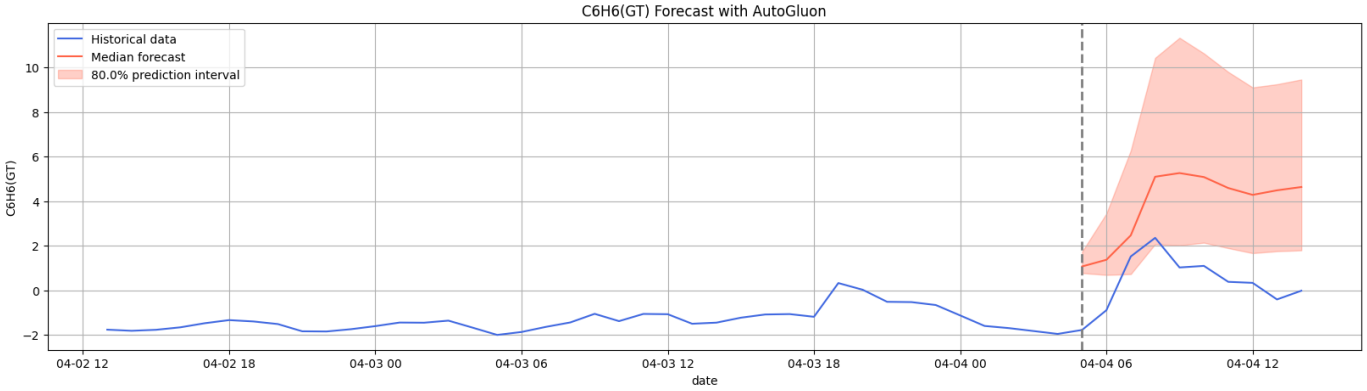

Fig. 11. Covariate forecasting with AutoGluon's extension for CHRONOS

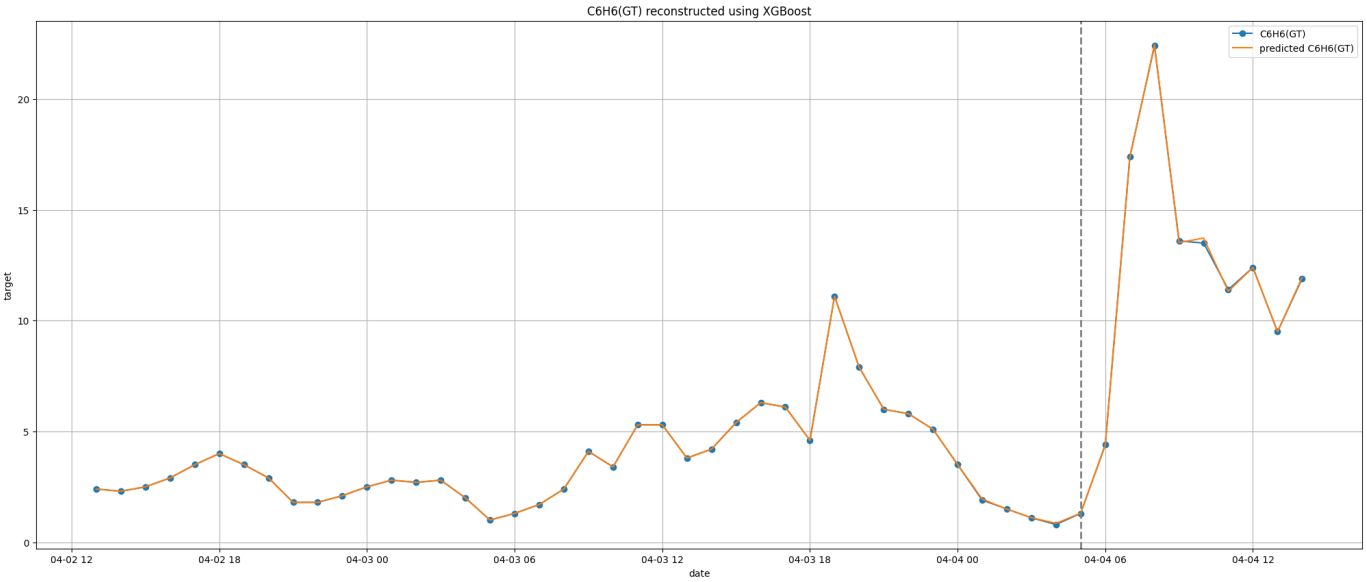

Fig. 12. XGBoost forecasting

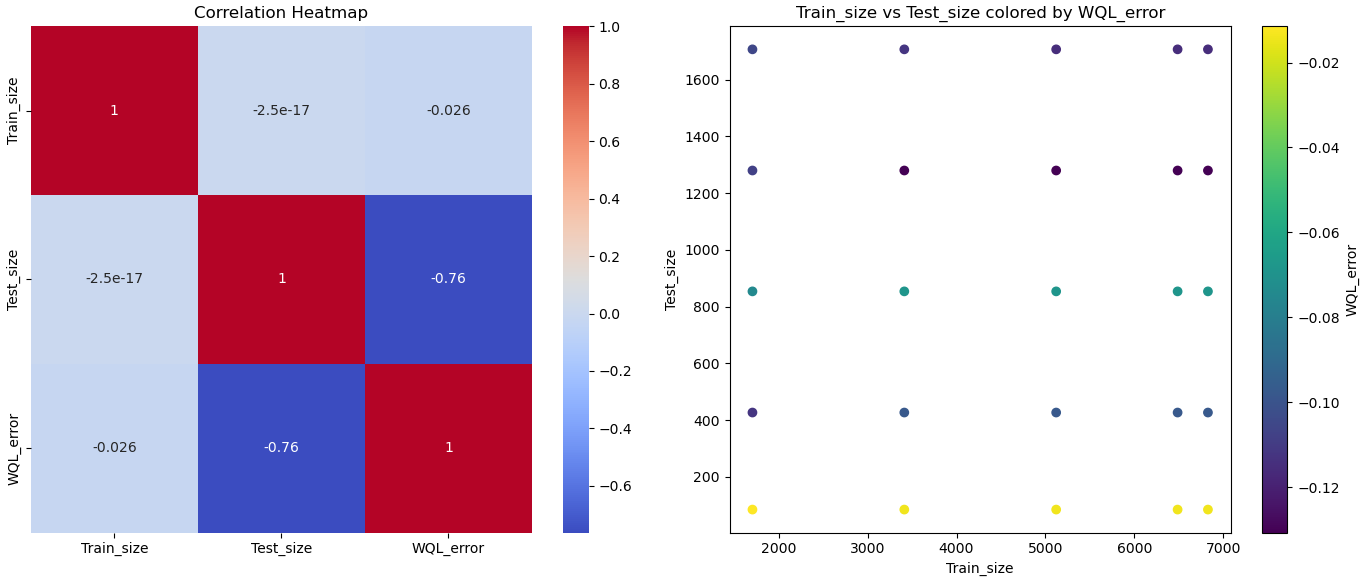

Fig. 13. Correlation between the WQL error and the size of the training and test split

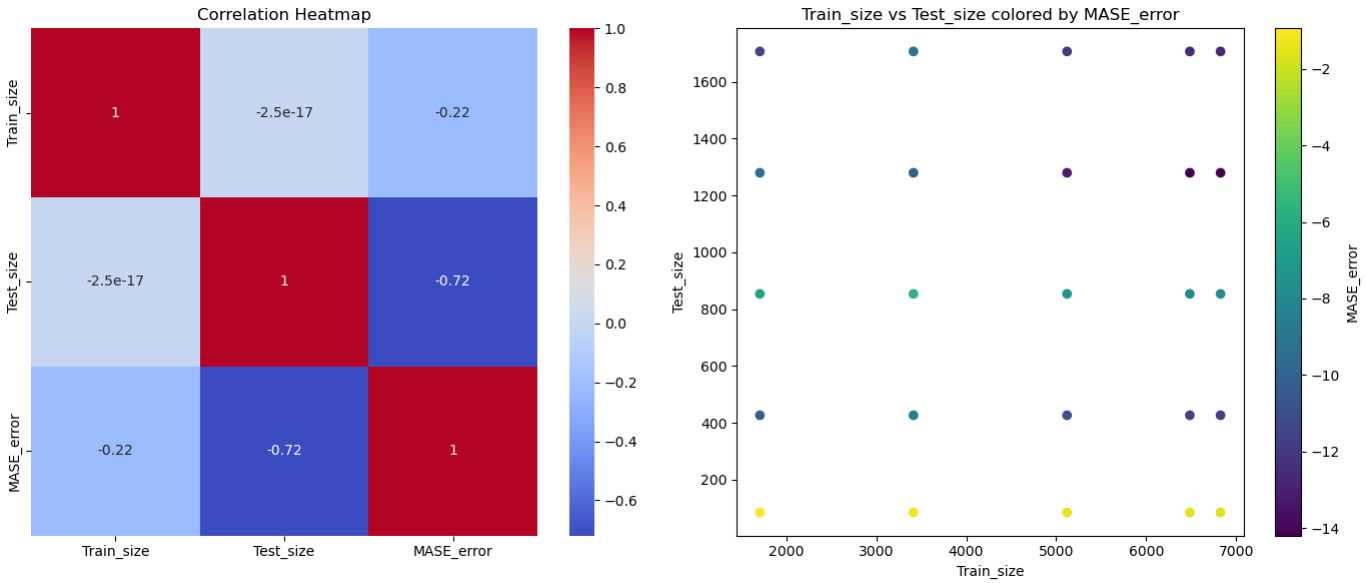

Fig. 14. Correlation between the MASE error and the size of the training and test split

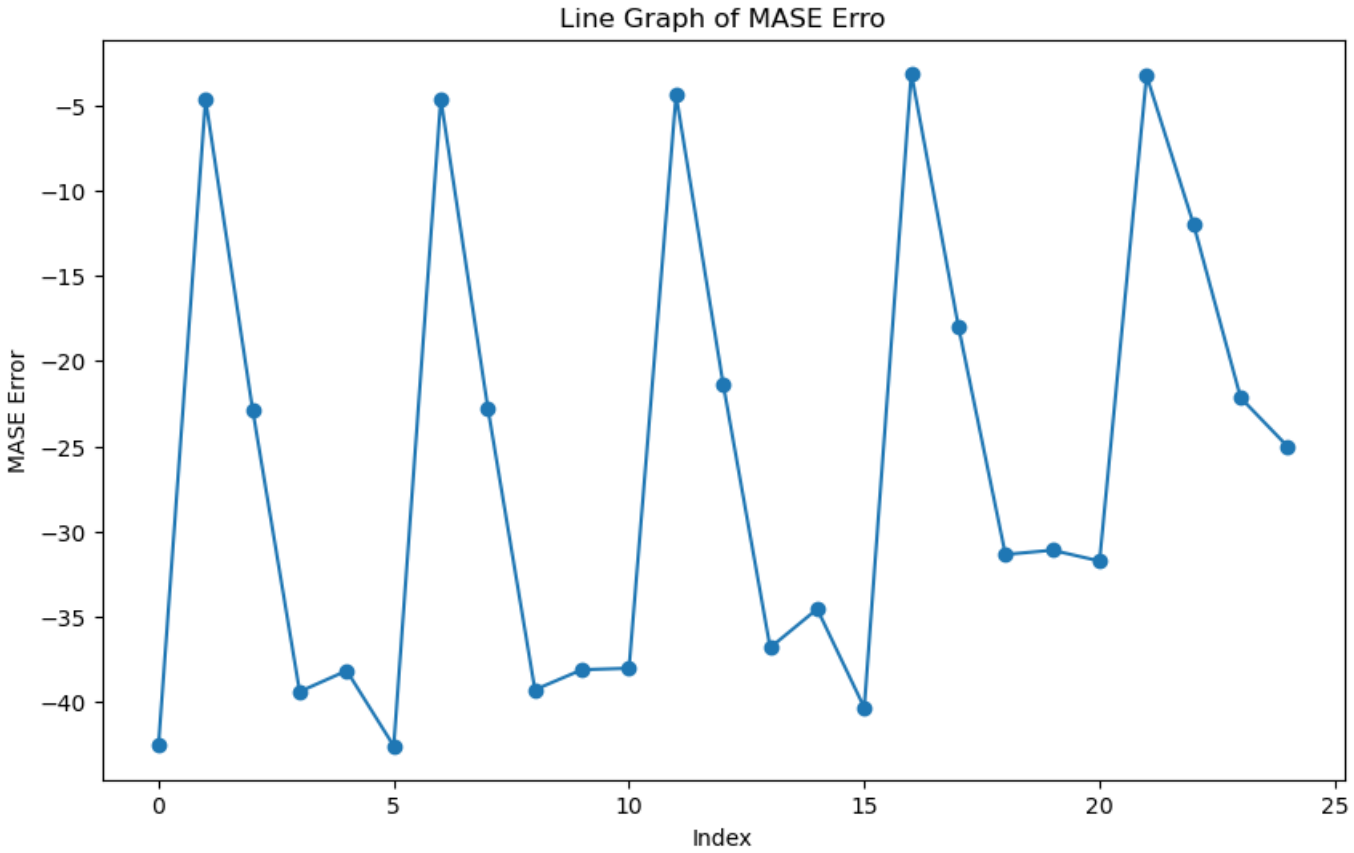

Fig. 15. behaviour of the MASE in a daily dataset of the Demography Category

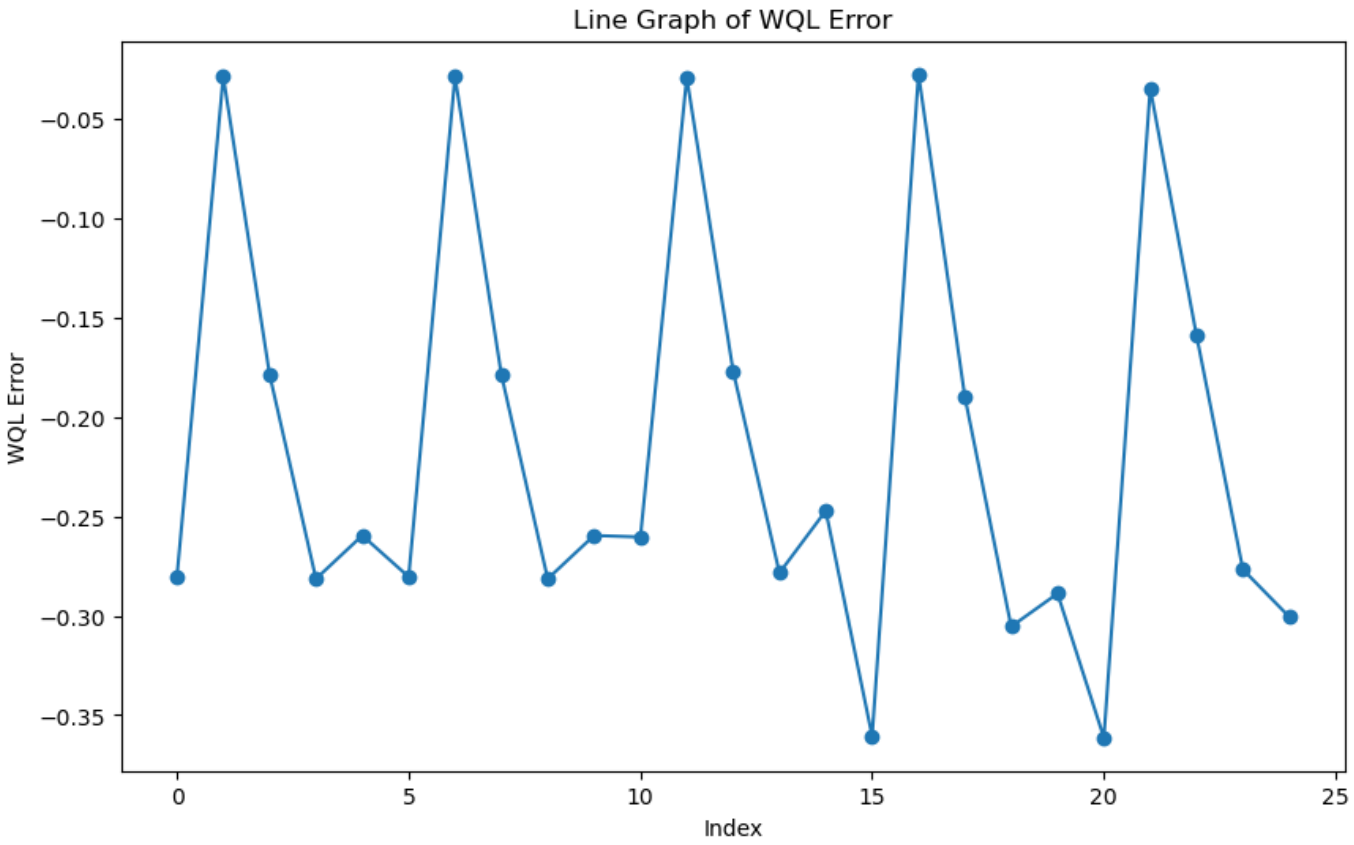

Fig. 16. behaviour of the WQL in a daily dataset of the Demography Category

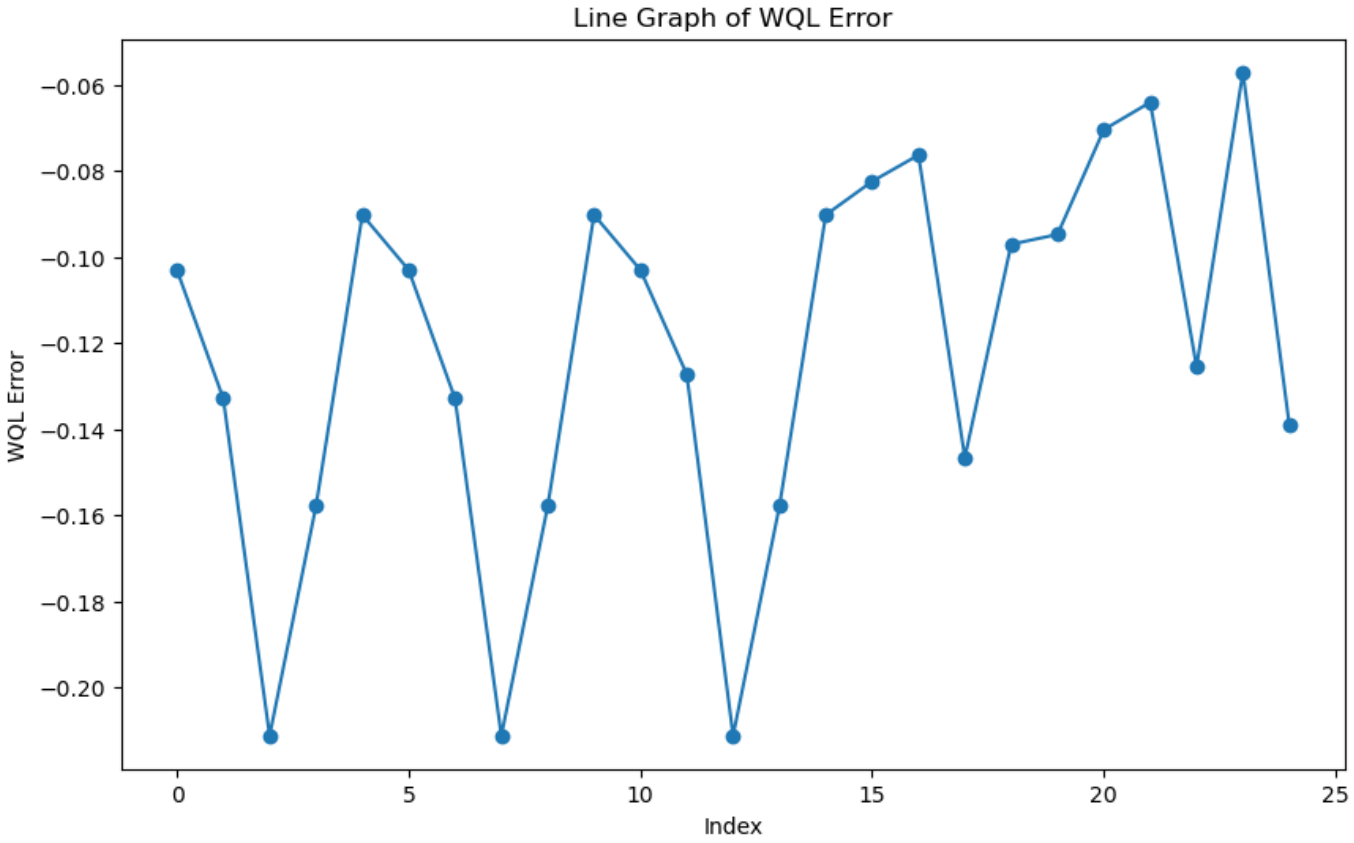

Fig. 17. behaviour of the WQL in a monthly dataset of the Demography Category

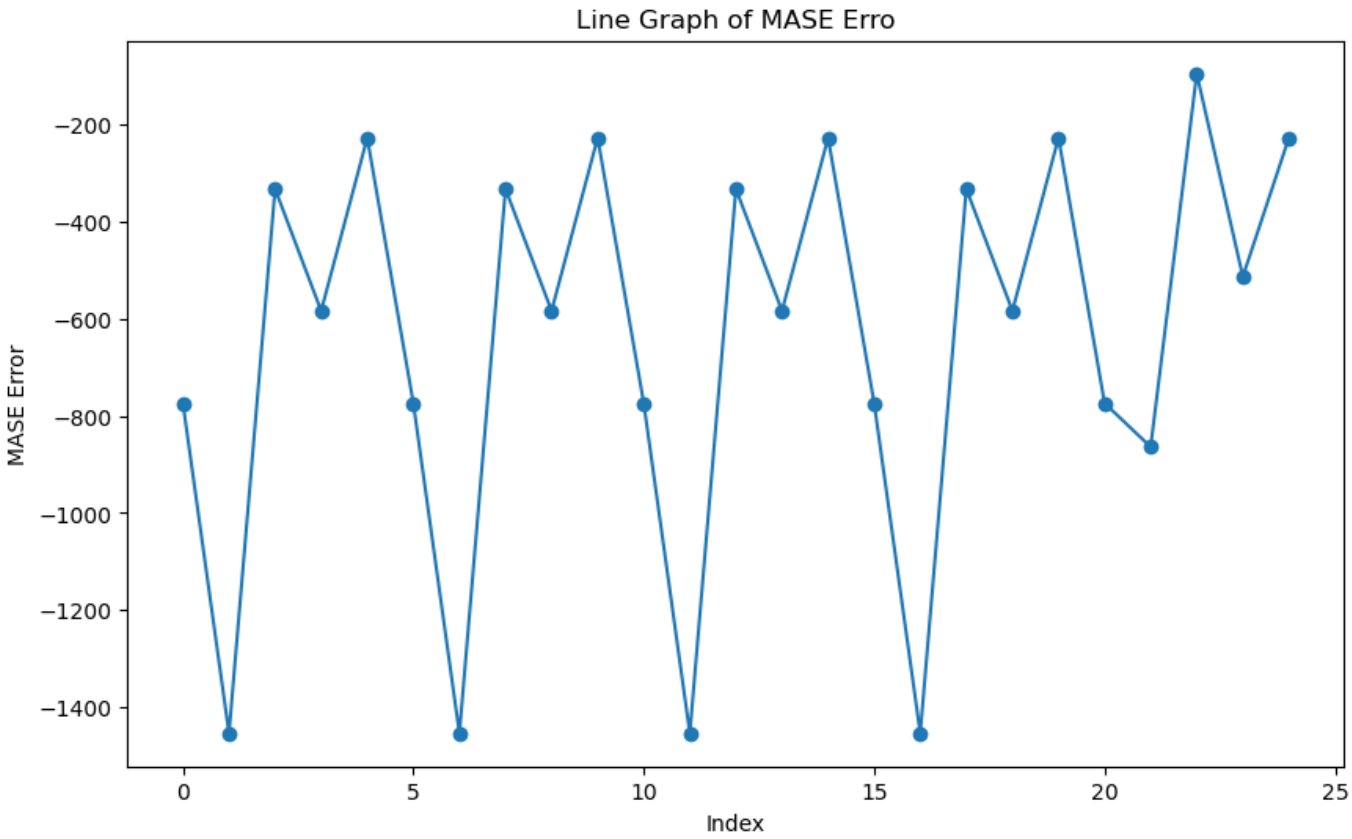

Fig. 18. behaviour of the MASE in a monthly dataset of the Demography Category

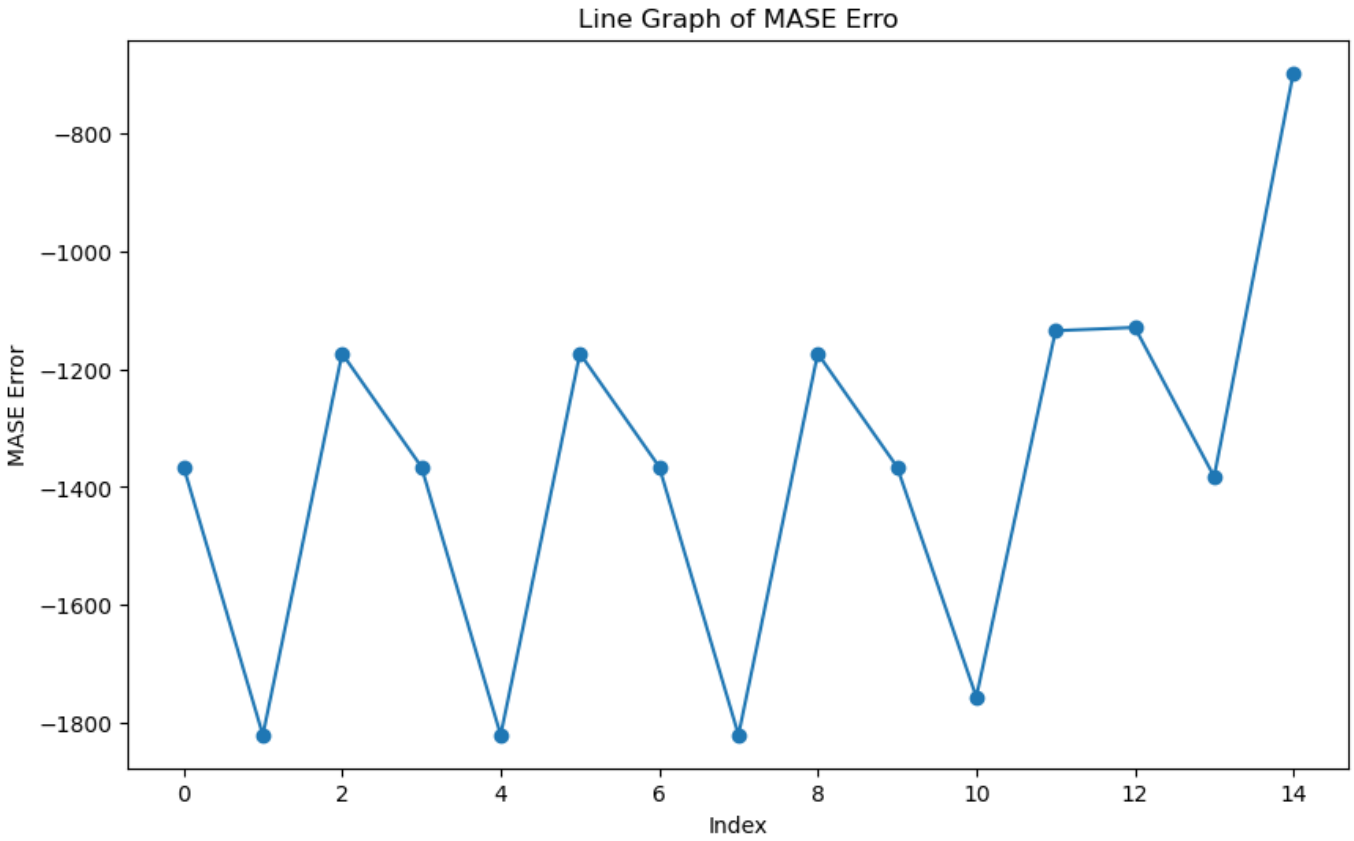

Fig. 19. behaviour of the MASE in a yearly dataset of the Demography Category

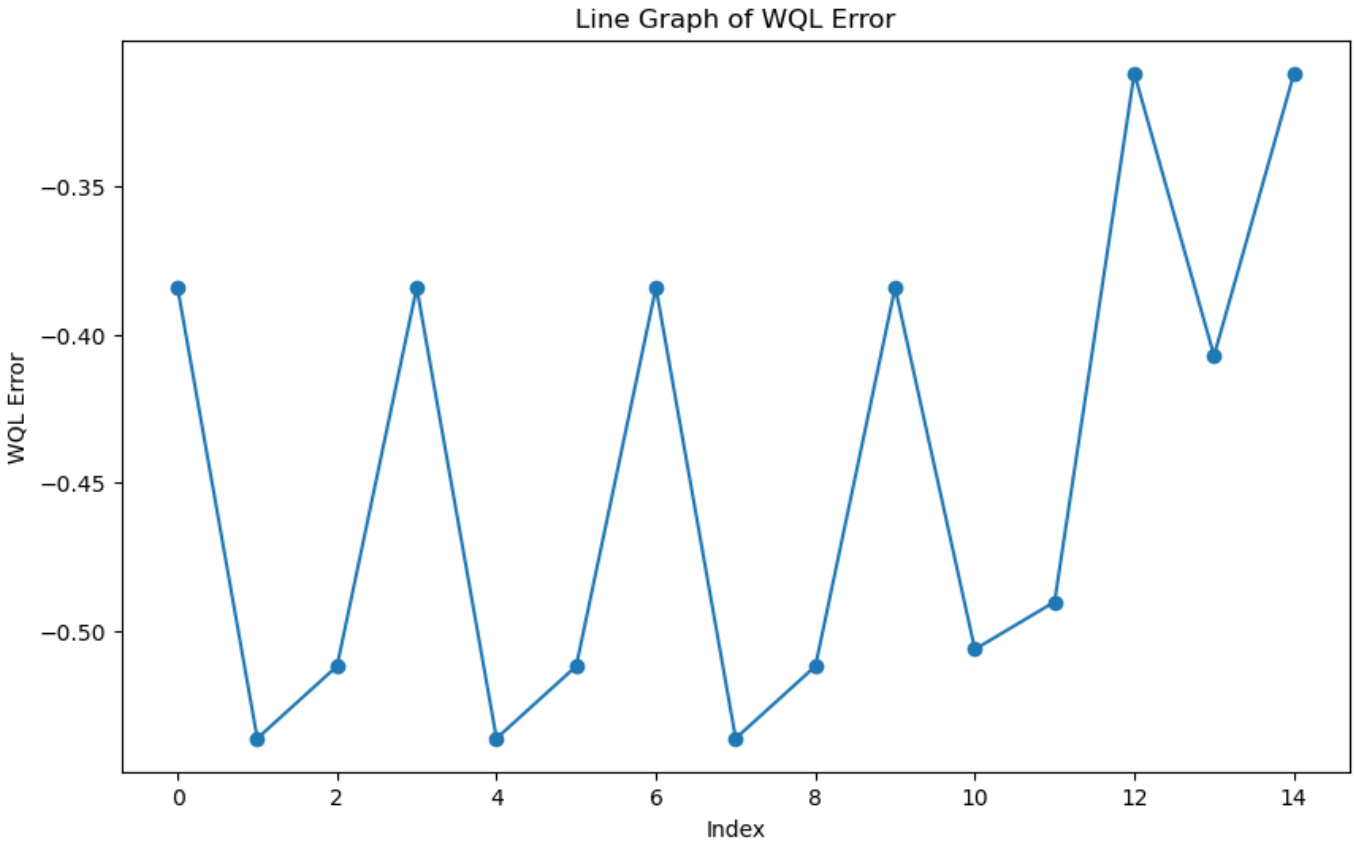

Fig. 20. behaviour of the WQL in a yearly dataset of the Demography Category