# Chronos for Multivariate Time Series Forecasting of Financial Data

Giacomo Bastiani
Politecnico di Torino
s343126
*s343126@studenti.polito.it*

Andrea Delli
Politecnico di Torino
s331998
*s331998@studenti.polito.it*

Giorgia Modi
Politecnico di Torino
s330519
*s330519@studenti.polito.it*

***Abstract*—Time-series forecasting is essential in financial markets, guiding decision-making in trading and risk management. This paper examines the application of Chronos, a deep learning framework originally designed for probabilistic univariate forecasting, to financial data. We begin by fine-tuning Chronos on financial time series to evaluate its performance improvements over a zero-shot approach. Next, we explore various strategies for adapting Chronos to multivariate forecasting to better capture key covariates. These include preprocessing techniques that transform multivariate data into a univariate representation and a post-processing approach using a Multi-Layer Perceptron (MLP) to refine Chronos' predictions. Experiments on historical stock market data, specifically Apple Inc. (AAPL), demonstrate that fine-tuning Chronos significantly enhances forecasting accuracy. Among preprocessing methods, Support Vector Regression (SVR) provides the best performance in transforming multivariate inputs. Furthermore, post-processing with an MLP further refines predictions, leading to substantial error reductions. To assess practical applicability, we implement a trading bot that utilizes the predicted stock prices for decision-making. The trading simulations reveal that fine-tuning Chronos on financial data combined with MLP post-processing outperforms both the zero-shot approach and a simple buy-and-hold strategy. These findings underscore the importance of integrating deep learning techniques and multivariate modeling in financial time-series forecasting.**
**The source code of this project is available at https://github.com/JackBstn/Chronos_multivariate.**

## I. Problem Statement

Time-series forecasting is a fundamental problem in many domains, including finance, weather prediction, and supply chain management. It involves predicting future values of a sequence based on its past observations, often using statistical or machine learning models. Traditional approaches often focus on univariate forecasting, where only past values of the target variable are used for prediction. However, in many real-world applications, multiple interdependent variables evolve over time, and leveraging these additional covariates can significantly enhance predictive performance. In this paper, we focus on financial forecasting, a particularly challenging task due to the high volatility and complex dependencies in stock price movements, where the consideration of multiple covariates is essential.

Chronos [1] is a deep learning framework designed for probabilistic univariate time-series forecasting, using pretrained transformer-based architectures, achieving competitive zero-shot forecasting on unseen datasets.

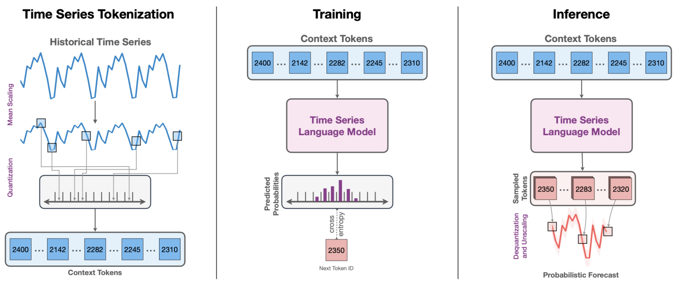

Fig. 1. High-level depiction of Chronos.

In this work, we explore the extension of Chronos to multivariate forecasting by incorporating additional covariates to improve the estimation of the response variable. Given a time-series dataset with multiple dimensions, let $\mathbf{X}_t \in \mathbb{R}^d$ denote the feature vector at time $t$, where one dimension represents the response variable $y_t$, and the remaining $d-1$ dimensions are covariates:

$$\mathbf{X}_t = \{y_t, x_t^1, x_t^2, ..., x_t^{d-1}\}.$$

The forecasting task consists of predicting the future values of $y_t$ given past observations of $\mathbf{X}$, i.e.,

$$y_{C+1:C+H} = f(\mathbf{X}_{1:C}) = f(\mathbf{X}_1, ..., \mathbf{X}_C),$$

where $C$ is the context size and $H$ is the forecasting horizon.

We first explore fine-tuning Chronos solely on the response variable $y_t$ to assess whether this improves performance. Then, to incorporate covariates into Chronos-based forecasting, we explore different strategies. The first method transforms the multivariate time-series into a univariate representation using simple statistical methods before applying Chronos. An alternative strategy applies Chronos independently to each dimension of $\mathbf{X}_t$, and a simple Multi-Layer Perceptron (MLP) is used to combine the results and reconstruct the final prediction.

Finally, we test these methodologies in a financial setting, where the forecasting output is utilized within a simple greedy trading bot. The effectiveness of each approach is evaluated using standard regression metrics and overall trading performance.

## II. Methodology

As previously discussed, in our work we employed Chronos model for time-series forecasting. This task involves using historical data from a quantity of interest (typically real-valued) to predict its future values. Formally, given a uniformly spaced time series $\boldsymbol{x}_{1:C} = [x_1, \dots, x_C]$, the goal is to predict the joint distribution of the next $H$ steps, $p(\boldsymbol{x}_{C+1:C+H}|\boldsymbol{x}_{1:C})$. In the case of univariate forecasting, as in the original Chronos model, the observations are scalars, i.e., $x_i \in \mathbb{R}$ for all $i$.

Figure 1 reports a high-level depiction of Chronos. Chronos tokenizes time-series values using scaling and quantization into a fixed vocabulary, enabling it to leverage transformer-based language models like T5 [2]. The model is trained using a cross-entropy loss and benefits from large-scale pretraining on both real and synthetic datasets. During inference, Chronos autoregressively predicts future values by sampling from its learned distribution. This approach allows Chronos to achieve strong performance on in-domain data and competitive zero-shot forecasting on unseen datasets. The output of Chronos consists of multiple probabilistic forecasts, typically capturing an 80% prediction interval, meaning that the true future values are expected to fall within this range with high confidence.

In our experiments, we do not modify the core Chronos pipeline. Instead, we apply fine-tuning for the financial domain, along with preprocessing and post-processing techniques to adapt it for multivariate forecasting. The details of these experimental adaptations are described in the next section.

## III. Experiments

In this section, we aim to explore the application of Chronos to financial data. Our goal is to assess its performance in forecasting stock market trends and investigate possible adaptations to enhance its predictive capabilities. We conduct experiments using historical stock market data from Yahoo Finance, specifically focusing on Apple Inc. (AAPL), which includes key market variables: Open, High, Low, Volume, and Close prices. The Close price serves as the target variable, while the other features act as covariates to enhance predictive performance.

In all our experiments, during validation of performance, Chronos model is provided with context data from January 1, 2024, to March 15, 2024, and is then used to predict stock prices for the following 10 days.

We perform our experiments on Google Colab, utilizing a T4 GPU for accelerated computation. The original implementation of Chronos[1] was used. For the neural network (NN) components, we leverage PyTorch, while Matplotlib is used for visualizations and plotting. Performance metrics are computed using Scikit-learn (sklearn).

We evaluate our models using standard forecasting metrics [3]. Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) measure absolute error magnitudes, with RMSE providing a more interpretable scale. Mean Absolute Percentage Error (MAPE) assesses relative error in percentage terms, making it useful for understanding proportional deviations. Lastly, Mean Absolute Scaled Error (MASE) offers a robust alternative, especially for comparing performance across different datasets. Lower metric values indicate better performance.

[1]https://github.com/amazon-science/chronos-forecasting

### A. Chronos Fine-tuning

In this experiment, we explore fine-tuning the Chronos forecasting model on financial time series data. The goal is to assess whether adapting Chronos to a specific stock (AAPL) can improve its predictive performance and to examine if this fine-tuning generalizes to other stocks (e.g., GOOGL).

The fine-tuning process is performed using historical stock data from Yahoo Finance, spanning 10 years (2013–2023). We focus on the Close prices of AAPL and configure the training with a learning rate of 0.001 while experimenting with two training step configurations: 1000 and 10000 epochs. The training loss during training epochs is reported in Figure 6.

The performance of the fine-tuned model is compared against a zero-shot model using the metrics presented before (MSE, RMSE, MAPE, MASE), both on the AAPL stock, which is used for training, and to the GOOGL stock in order to assess models' generalization in the financial domain.

TABLE I
Performance of Chronos Zero-Shot vs. Fine-Tuned across training sizes and test tickers.

| **Configuration** | MSE ↓ | RMSE ↓ | MAPE ↓ | MASE ↓ |
|---|---|---|---|---|
| **AAPL** | | | | |
| Zero-Shot | 5.5113 | 2.3476 | 1.0250 | 1.0758 |
| Fine-Tuned (1000) | **5.7938** | **2.4070** | **0.8762** | **0.9238** |
| Fine-Tuned (10000) | 14.4451 | 3.8007 | 1.5837 | 1.6626 |
| **GOOGL** | | | | |
| Zero-Shot | 43.4885 | 6.5946 | 4.1831 | 3.3841 |
| Fine-Tuned (1000) | 17.8674 | 4.2270 | 2.7398 | 2.2124 |
| Fine-Tuned (10000) | **7.8661** | **2.8046** | **1.6938** | **1.3618** |

The results reported in Table I show that for AAPL, fine-tuning Chronos for 1000 epochs significantly reduces all error metrics compared to the zero-shot model, indicating that focused training on a single stock can capture its unique dynamics effectively. However, extending the fine-tuning to 10000 epochs leads to a decline in performance relative to the 1000-epoch configuration, which suggests that the model starts to overfit the AAPL training data.

For GOOGL, fine-tuning on AAPL significantly improved performance, with error metrics dropping further as training extends to 10000 epochs. Despite potential overfitting on AAPL, the model appears to have captured latent market patterns that generalize well to GOOGL, enhancing its predictive accuracy.

The forecasting plots in Appendix A confirm that fine-tuning enhances prediction accuracy compared to the zero-shot approach. For AAPL, fine-tuning with 1000 epochs yields forecasts that closely align with actual prices, while for GOOGL, the best results are achieved with 10000 epochs.

However, for both stocks, fine-tuning with 10000 epochs reduces model uncertainty, as reflected in the narrowing 80% prediction interval.

## *B. Preprocessing Methods for Multivariate-to-Univariate Transformation*

To adapt Chronos for multivariate forecasting, we explore preprocessing techniques that transform multivariate input into a univariate representation before applying Chronos. These methods fall into two categories: simple statistical transformations and ML-based approaches.

*a) Simple Statistical Methods:* We consider basic transformations that reduce the multivariate time-series to a single-dimensional representation:

- **Arithmetic Mean:** Computes the simple average of all covariates at each time step.
- **Principal Component Analysis (PCA):** Projects the multivariate series into its first principal component.
- **Weighted Sum:** Applies fixed weights to each covariate and sums them to obtain a univariate series. Different weight combinations were tested and the best one was selected.

*b) ML-Based Methods:* We also investigate data-driven transformations using machine learning:

- **Linear Regression:** Learns a linear combination of covariates to predict the target variable.
- **Random Forest:** Uses an ensemble of decision trees to model relationships in the multivariate data.
- **Support Vector Regression (SVR):** Employs a kernel-based regression approach to capture non-linear dependencies.

After applying these transformations, we applied Chronos on the resulting univariate series and evaluate forecasting performance.

Tables II and Table III present the results for simple statistical methods and ML-based transformations respectively. In both tables, “Zero-Shot” refers to predictions made by Chronos without any modifications applied, using only the target variable “Close”.

TABLE II
COMPARISON OF SIMPLE PREPROCESSING METHODS FOR CHRONOS FORECASTING.

| Method | MSE ↓ | RMSE ↓ | MAPE ↓ | MASE ↓ |
|---|---|---|---|---|
| Zero-Shot | 21.6738 | 4.6555 | 1.7280 | 1.6963 |
| Weighted Sum | 9.4147 | 3.0683 | 2.3355 | 2.5822 |
| Arithmetic Mean | **5.0647** | **2.2505** | **1.2343** | **1.3786** |
| PCA | 11.5545 | 3.3992 | 1.9314 | 3.2845 |

The results demonstrate that all preprocessing techniques significantly reduced forecasting errors across all metrics compared to the baseline “Zero-Shot”. Arithmetic Mean achieved the best performance among simple methods. On the other hand, for ML-based approaches, Support Vector Regression (SVR) outperformed other models, achieving also the overall best results.

TABLE III
COMPARISON OF ML-BASED PREPROCESSING METHODS FOR CHRONOS FORECASTING.

| Method | MSE ↓ | RMSE ↓ | MAPE ↓ | MASE ↓ |
|---|---|---|---|---|
| Zero-Shot | 21.6738 | 4.6555 | 1.7280 | 1.6963 |
| Linear Regression | 9.8389 | 3.1367 | 2.3248 | 2.2640 |
| Random Forest | 6.1217 | 2.4742 | 1.3226 | **1.4692** |
| SVR | **4.9271** | **2.2197** | **1.1278** | 1.5331 |

The forecasting plots illustrating the results of different preprocessing methods are reported in Appendix B. As observed, the Chronos 80% prediction interval shrinks significantly across all preprocessing methods compared to the baseline “Zero-shot”, with the exception of Weighted Sum (Figure 7) and Linear Regression (Figure 10). This suggests that preprocessing techniques generally help Chronos generate more confident and stable forecasts by effectively incorporating multivariate information.

These findings suggest that integrating additional information from covariates provides crucial insights that cannot be extracted using only the target variable. Therefore, an adaptation of Chronos to multivariate forecasting becomes necessary. This experiment shows that transforming a multivariate problem into a univariate representation through preprocessing is a viable and effective strategy, allowing Chronos to adapt successfully to multivariate forecasting and achieve significantly improved results.

## *C. Post-processing using MLP for Multivariate-to-Univariate Transformation*

To further enhance Chronos’ forecasting capabilities, we explore a post-processing approach that leverages a Multi-Layer Perceptron (MLP) to refine predictions. Unlike preprocessing methods that transform input data before applying Chronos, this approach focuses on improving Chronos’ predictions by aggregating them into a single final forecast for the target variable—the next day’s closing price.

The approach consists of applying Chronos independently to each dimension of the multivariate time series to generate separate predictions. These predictions are then used as input features for an MLP, which learns to merge them into a single refined forecast of the next day’s closing price.

The training dataset is generated by applying a sliding window across a specified data interval. For each window, Chronos is used to predict the next day’s value for each covariate, and the true value of the response variable at that time is used as the label. These labeled predictions for each covariate then serve as inputs to the MLP.

The MLP is designed with three fully connected layers, where the input layer accepts Chronos’ multivariate predictions as features, then there are two hidden layers that utilize ReLU activation functions and dropout regularization, and at the end there’s the output layer, that produces a single scalar value representing the refined closing price forecast.

The network is trained with a learning rate of $0.001$ and a weight decay of $4 \times 10^{-5}$. The learning rate follows a step decay schedule with step sizes of $\{150, 300, 600\}$ epochs and a decay factor of $0.1$. The network consists of two hidden layers with 128 and 64 units, respectively, and is trained using Mean Squared Error (MSE) loss and an Adam optimizer. A learning rate scheduler with step-wise decay and early stopping with a patience of 100 epochs are applied to prevent overfitting. The maximum number of training epochs is set to 1000. The training and validation loss during training is reported in Figure 14.

We evaluate the performance of Zero-Shot Chronos versus Chronos combined with MLP. The results are presented in Table IV, and the plot is reported in Figure 13.

TABLE IV
COMPARISON OF CHRONOS WITH AND WITHOUT MLP POST-PROCESSING.

| Method | MSE ↓ | RMSE ↓ | MAPE ↓ | MASE ↓ |
|---|---|---|---|---|
| Zero-Shot | 109.9269 | 10.4846 | 5.1278 | 5.3112 |
| Chronos + MLP | **5.8794** | **2.4247** | **0.9345** | **0.9848** |

The results demonstrate that incorporating an MLP as a post-processing step significantly enhances forecasting accuracy. The Chronos+MLP combination achieves a considerable reduction in all the metrics compared to Chronos Zero-Shot, highlighting the advantage of leveraging deep learning to refine multivariate predictions into a more accurate univariate forecast.

## D. Stock trading simulation

To practically verify the capabilities of the various models (Zero-Shot, Fine-Tuned, Zero-Shot + MLP, and Fine-Tuned + MLP) a stock trading simulation was implemented. The Zero-Shot model refers to the original Chronos model, while the Fine-Tuned model corresponds to Chronos fine-tuned as described in Section III-A. The Zero-Shot with MLP applies the MLP as outlined in Section III-C, and the Fine-Tuned with MLP combines the fine-tuned model with the neural network. The objective is to assess how effectively the aforementioned systems can predict the next day's closing price and make informed trading decisions, ultimately comparing its performance against a traditional Buy & Hold strategy.

The stock that has been used is AAPL. A sliding window of 90 days is used as context for each prediction, starting from November 1st, 2023.

The trading bot operates on a daily basis following a straightforward decision-making process:

- BUY: If the predicted closing price is at least $0.1\%$ higher than the current price and sufficient capital is available, as many shares as possible are purchased.
- SELL: If the predicted closing price is at least $0.1\%$ lower than the current price and shares are held, all held shares are sold.
- HOLD: If the predicted change does not exceed the ±$0.1\%$ threshold, or the predicted change does exceed the fixed threshold but there are no shares held or insufficient budget, the current position is maintained.

Each day, the portfolio value is updated by combining the remaining budget with the current value of any held shares.

At the end of the simulation period, the final portfolio value is compared with a Buy & Hold benchmark, where the entire initial capital is invested at the beginning of the forecasting period. The results of the simulations are shown in Table V, while the performance of the different trading strategies over time is presented in Figure 15. Overall, these results suggest that fine-tuning the Chronos model is the most critical factor in enhancing trading performance, as it leads to higher portfolio values. The addition of the neural network offers a slight further improvement, particularly when paired with the fine-tuned model. In fact, Fine-Tuned + MLP achieved the best results.

TABLE V
COMPARISON OF THE TRADING RESULTS USING DIFFERENT METHODS.

| Method | Initial budget [€] | Final portfolio value [€] |
|---|---|---|
| Zero-Shot | 10000.00 | 9981.79 |
| Fine-Tuned | 10000.00 | 10247.77 |
| Zero-Shot + MLP | 10000.00 | 10064.53 |
| Fine-Tuned + MLP | 10000.00 | **10289.94** |

## IV. CONCLUSION

This study explored the extension of Chronos, a deep learning framework for univariate time-series forecasting, to a multivariate financial setting. The remarkable outcomes of our research highlight that fine-tuning Chronos on financial data significantly enhances its forecasting accuracy over the zero-shot approach. Additionally, preprocessing techniques, particularly Support Vector Regression (SVR), proved to be the most effective in transforming multivariate inputs into a form suitable for Chronos, while post-processing with a Multi-Layer Perceptron (MLP) further refined predictions and reduced errors. In a simulated trading environment, the fine-tuned Chronos model with MLP post-processing outperformed both the baseline zero-shot approach and a buy-and-hold strategy.

Despite these successes, we encountered several challenges. Fine-tuning Chronos posed challenges with overfitting, particularly when training for extended epochs, and adapting Chronos to multivariate data introduced complexity in feature selection and transformation. While SVR improved accuracy, it added computational overhead, making real-time applications more challenging.

Key insights of our work include the necessity of fine-tuning transformer-based models on domain-specific data, the importance of leveraging multivariate information for capturing financial dependencies, and the practical benefits of integrating deep learning with structured forecasting pipelines. Future work could explore alternative feature selection methods, more advanced post-processing architectures, and reinforcement learning for adaptive trading strategies.

TABLE VI
TRAINING HYPERPARAMETERS AND NETWORK ARCHITECTURE

| **Parameter** | **Value** |
|---|---|
| Learning Rate | 0.001 |
| Weight Decay | $4 \times 10^{-5}$ |
| Learning Rate Schedule | Step decay |
| Step Sizes | $\{150, 300, 600\}$ epochs |
| Decay Factor | 0.1 |
| Hidden Layers | 2 (128, 64 units) |
| Loss Function | Mean Squared Error (MSE) |
| Optimizer | Adam |
| Early Stopping | Patience of 100 epochs |

## REFERENCES

[1] A. F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. P. Arango, S. Kapoor, J. Zschiegner, D. C. Maddix, H. Wang, M. W. Mahoney, K. Torkkola, A. G. Wilson, M. Bohlke-Schneider, and Y. Wang, "Chronos: Learning the language of time series," 2024. [Online]. Available: https://arxiv.org/abs/2403.07815

[2] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *CoRR*, vol. abs/1910.10683, 2019. [Online]. Available: http://arxiv.org/abs/1910.10683

[3] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207006000239

## Appendix A
## Chronos Fine-Tuning

In this appendix, we present all the figures obtained during the experiment presented in section III-A.

Figure 2, 3, 4, 5 display the results for the Chronos fine-tuning methods, applied on different stock prices, with the training losses reported in Figure 6.

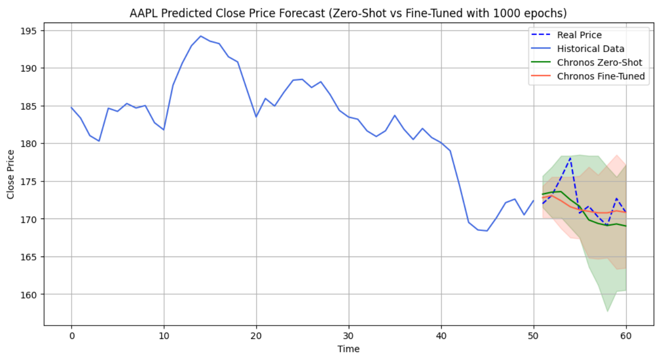

Fig. 2. Fine-Tuning with 1000 epochs - Results on AAPL

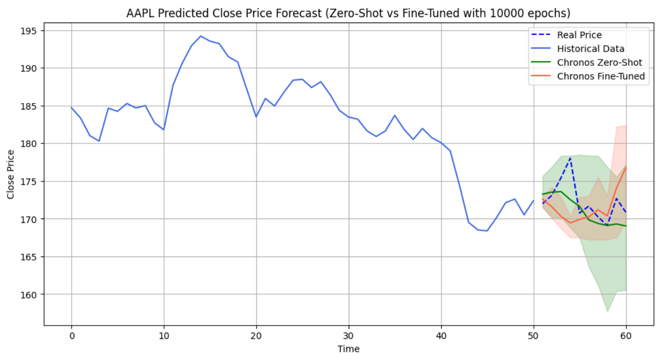

Fig. 3. Fine-Tuning with 10000 epochs - Results on AAPL

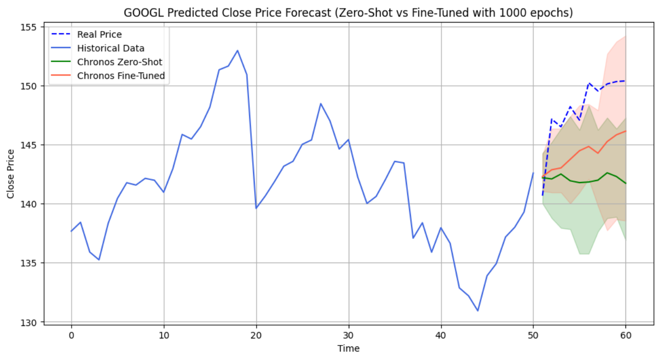

Fig. 4. Fine-Tuning with 1000 epochs - Results on GOOGL

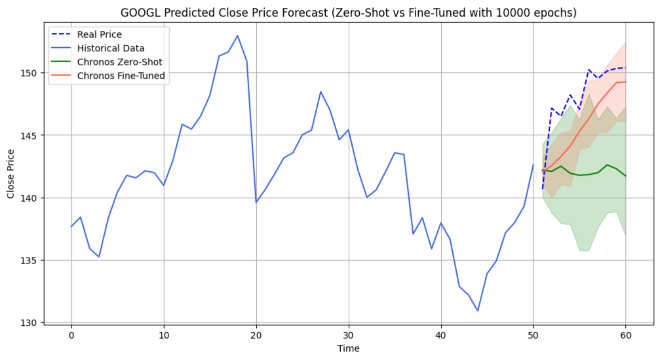

Fig. 5. Fine-Tuning with 10000 epochs - Results on GOOGL

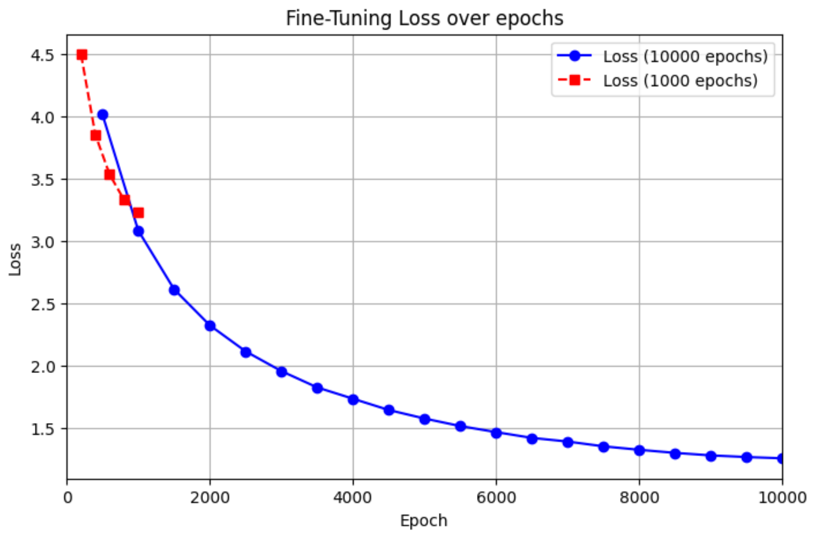

Fig. 6. Training loss during Chronos Fine-Tuning

## Appendix B
## Preprocessing Methods for Multivariate-to-Univariate Transformation

In this appendix, we present all the figures obtained during the experiment presented in section III-B.
Figures 7, 8 and 9 display the results for the simple statistical preprocessing methods.
Figures 10, 11 and 12 present the results for the ML-based preprocessing methods.

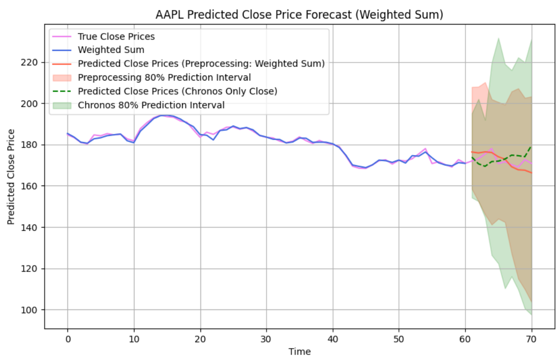

Fig. 7. Weighted Sum Results

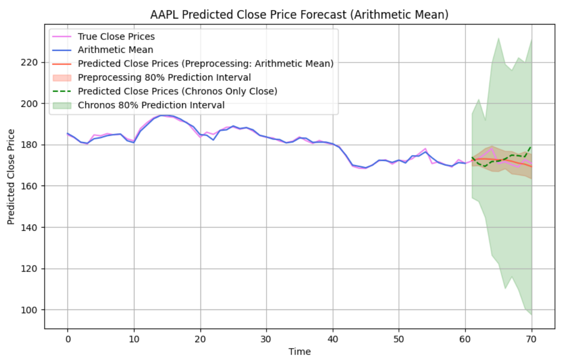

Fig. 8. Arithmetic Mean Results

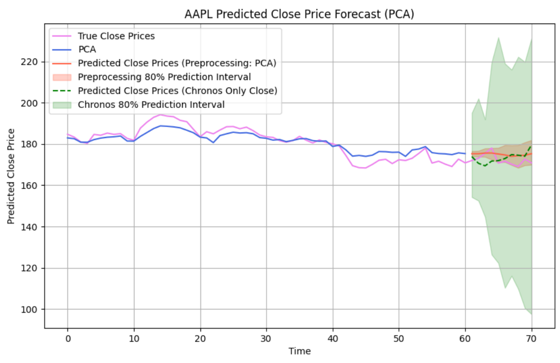

Fig. 9. PCA Results

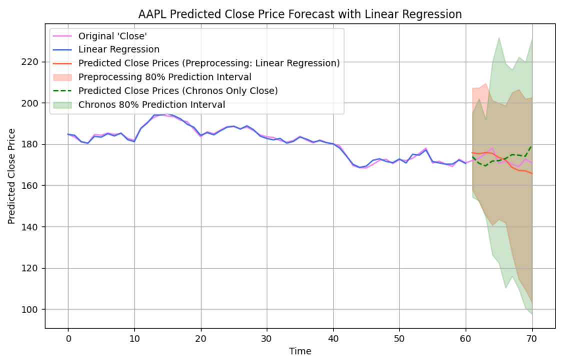

Fig. 10. Linear Regression Results

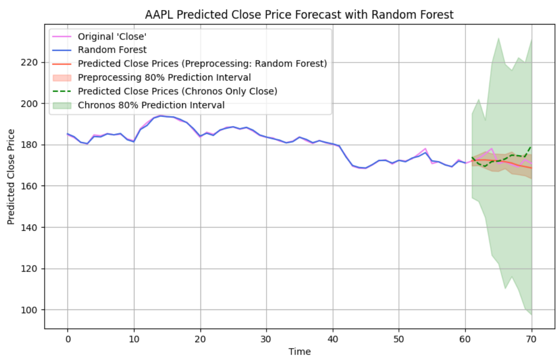

Fig. 11. Random Forest Results

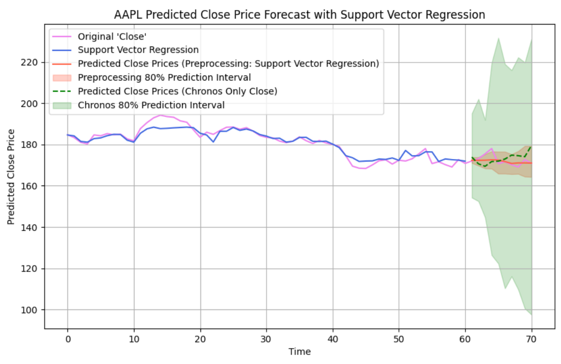

Fig. 12. Support Vector Regression Results

## Appendix C
## Post-processing using MLP for Multivariate-to-Univariate Transformation

In this appendix, we present all the figures obtained during the experiment presented in section III-C.
Figure 13 presents the results for the Chronos+MLP method, with the Neural Network loss reported in Figure 14.

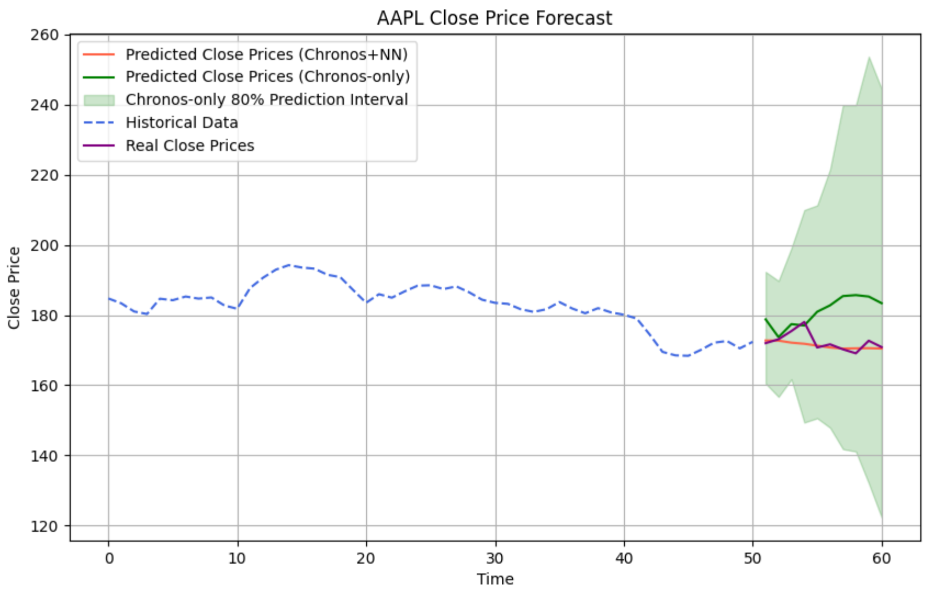

Fig. 13. Chronos + MLP Results

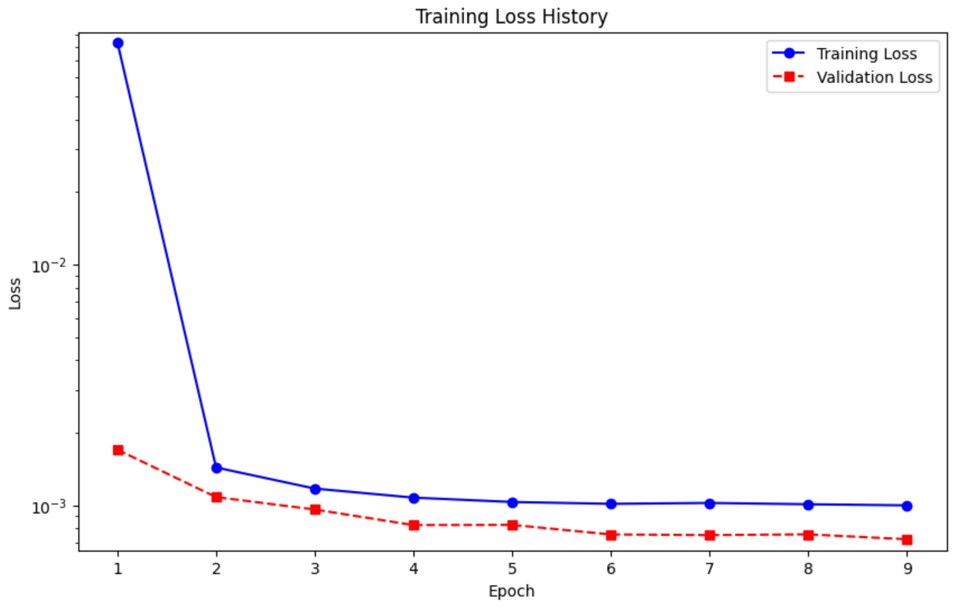

Fig. 14. Training and validation losses during MLP training

## Appendix D
## Trading bot

In this appendix, we present all the figures obtained during the experiment presented in section III-D. Figure 15 shows the different results obtained using the discussed trading methods.

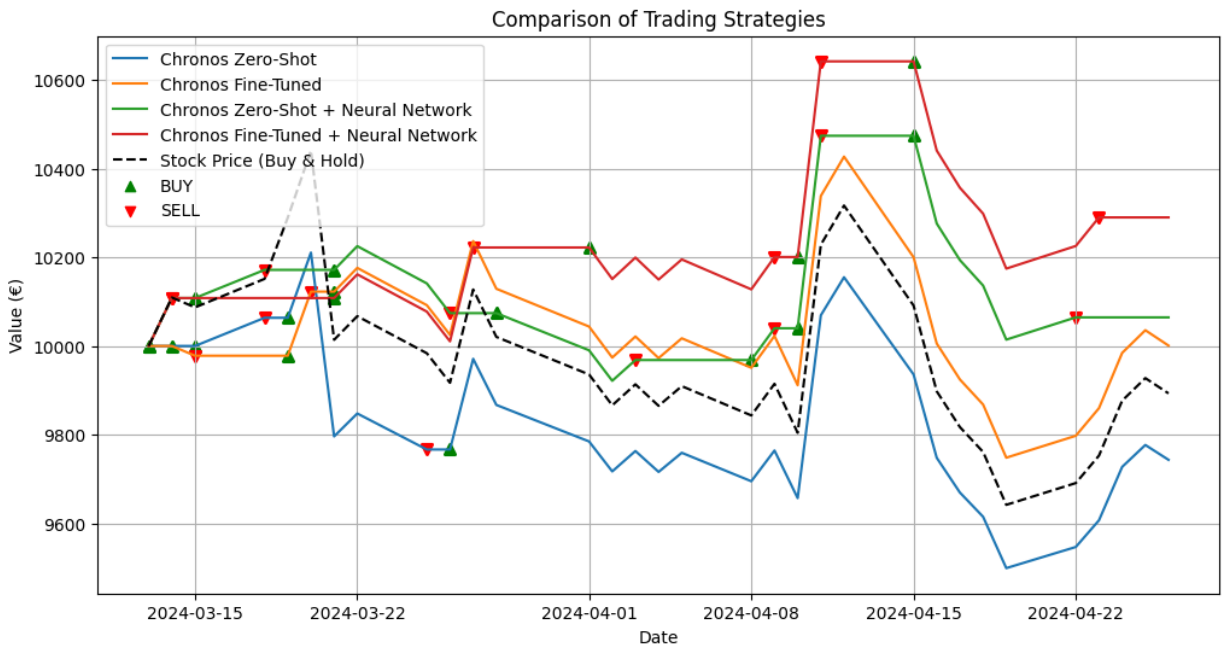

Fig. 15. Comparison of different trading strategies