Hyperparameter Tuning Optimization for Time Series Financial Analysis

Nikita Gavrilov*, Lynn van Rijt, Thomas Hordijk, Renee Theunissen, Georgiana Manolache*

Fontys University of Applied Sciences, The Netherlands

Email*: {n.gavrilov@student.fontys.nl, g.manolache@fontys.nl}

BDO Accountants & Adviseurs, The Netherlands

Email: {lynn.van.rijt@bdo.nl, thomas.hordijk@bdo.nl, renee.theunissen@bdo.nl}

Abstract

This research investigates hyperparameter optimization (HPO) strategies for metalearning in financial time series analysis, focusing on maximizing prediction accuracy while maintaining computational efficiency. Through a comprehensive evaluation of four HPO approaches - Random Search, Trajectory-Based-Optimization, Optuna, and Adaptive Fidelity Identification (AFI) - we examined their performance across multiple time series datasets within Microsoft Azure Databricks environment. The study utilized multiple evaluation metrics including RMSE, MAE, and R^2, complemented by bootstrap resampling with 10,000 iterations for statistical validation.

Results revealed that sophisticated HPO methods did not provide statistically significant improvements over Random Search, with Trajectory HPO showing comparable performance ($p = 0.4550$) but requiring higher computational resources. AFI HPO demonstrated statistically significant differences ($p = 0.0001$) but with inferior prediction accuracy. The consistent occurrence of negative R-squared values across all methods suggested fundamental challenges in capturing underlying patterns in financial time series data. The research also identified limitations in the current approach to hyperparameter space definition, suggesting that method-specific optimization spaces might produce better results than standardized configurations.

These findings contribute to the understanding of HPO in financial forecasting by challenging assumptions about the necessity of complex optimization strategies and highlighting the importance of balancing complexity with practical utility. The study provides valuable insights for practitioners in selecting appropriate HPO strategies based on their computational constraints and accuracy requirements.

1. Introduction

Financial time series analysis has long been a foundation of economic research and is essential for informed decision-making in sectors such as finance, trading, and economic planning (Chatterjee et al., 2021). Traditional machine learning methods have provided significant advances in time series forecasting. However, the surge in available data and the growing demand for robust predictive performance have led to the adoption of more sophisticated techniques, including metalearning. Metalearning, or "learning to learn," focuses on creating algorithms that adapt dynamically to new tasks by leveraging knowledge from previous learning experiences (Huisman et al., 2021). Despite its potential to enhance forecasting accuracy and flexibility, the effectiveness of metalearning heavily depends on its underlying hyperparameter configurations, which govern the learning dynamics and performance of these models (Hospedales et al., 2021).

The challenge of hyperparameter tuning (HPO) becomes obvious in metalearning due to the interplay between the algorithm's complexity and the noisy nature of financial time series data. Poorly optimized hyperparameters can lead to overfitting, underfitting, or suboptimal learning performance, ultimately hindering the predictive power of metalearning models. Moreover, traditional random search method for HPO often fail to capture the nuanced interactions between hyperparameters in high-dimensional search spaces, especially when computational resources are constrained (Bischl et al., 2023). This limitation becomes even more critical when dealing with time-sensitive financial forecasting tasks, where iterative model modification can be expensively time-consuming. Without an efficient and systematic approach to hyperparameter optimization, the full potential of metalearning in financial forecasting cannot be realized, leaving a significant gap in leveraging advanced learning methods for practical forecasting challenges.

We address these challenges by exploring optimized strategies for hyperparameter tuning in the context of metalearning for financial time series analysis. The question that we intended to answer is:

How can hyperparameter optimization for a meta-learner be achieved to maximize prediction accuracy while minimizing or avoiding any negative impact on computational speed?

The study aims to establish frameworks and methodologies that enhance the efficiency and effectiveness of hyperparameter tuning while minimizing computational overhead. By doing so, we seek to improve the flexibility and performance of metalearning algorithms, enabling them to address the complexities of financial time series data. The outcomes of this research are expected to contribute significantly to both the theoretical understanding and practical applications of hyperparameter optimization in financial forecasting, breaking ground for more accurate and efficient decision-making processes in the financial sector.

First, we will discuss the context and setting of the study as well as outline analysis methods in the Section 2. In the Section 3 of this document, we will report on data collection as well as present key findings and secondary findings with respect to the main research question. We will discuss the study's strengths and limitations, and compare the main results to previous research, in the upcoming Section 4. Finally, the Section 5 will evaluate the research and provide perspectives for future work.

2. Preliminary Research

This chapter presents a comprehensive review of the literature concerning hyperparameter optimization (HPO) in the context of metalearning for time series forecasting. The review examines theoretical foundations, current methodologies, and emerging trends in the field, with particular emphasis on the challenges and opportunities in financial time series analysis.

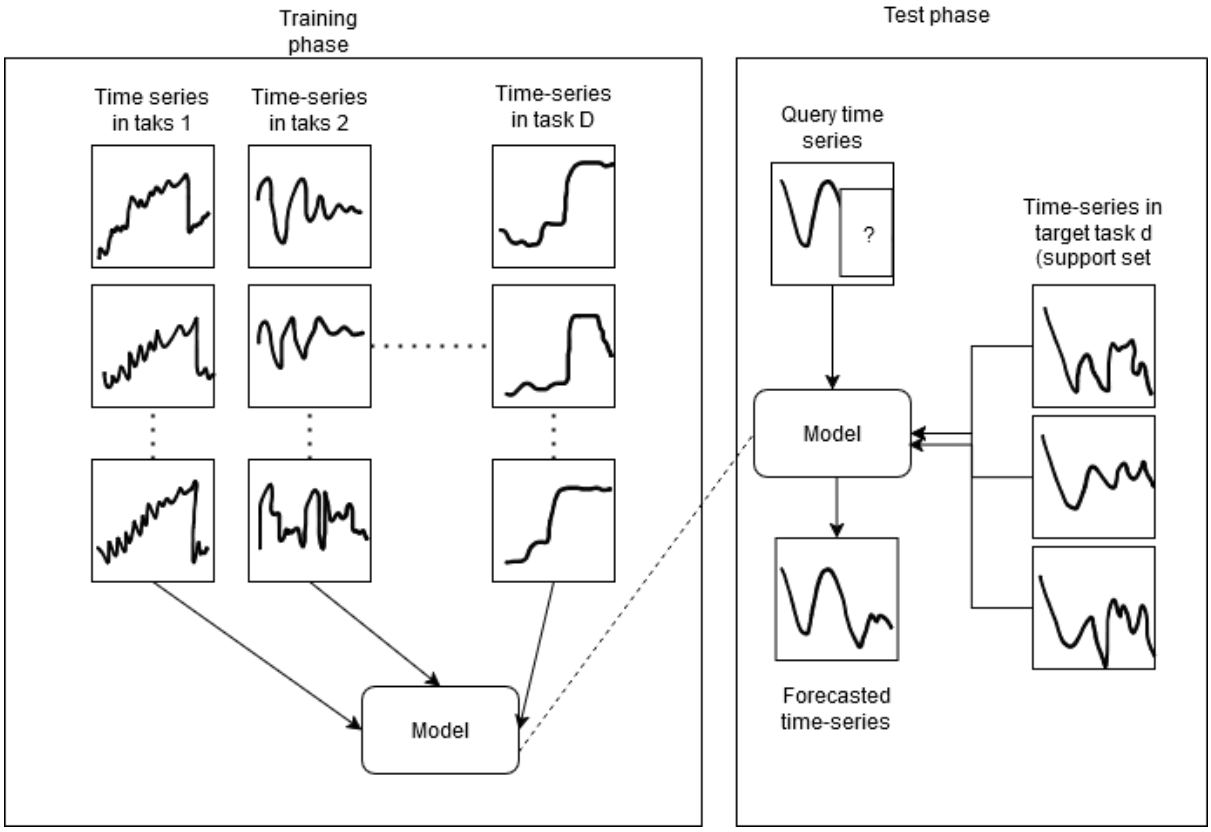

Figure 1: Metalearning workflow

2.1 Foundations of Metalearning

Metalearning, represents a sophisticated approach to machine learning that leverages knowledge from previous learning experiences to improve performance on new tasks. As highlighted by Huisman et al. (2021), metalearning operates on two distinct levels: base-level learning, which focuses on specific problem-solving, and meta-level learning, which aims to enhance the learning process itself. This dual-level approach enables systems to adapt more efficiently to new tasks by capitalizing on prior experiences as it is shown on the Figure 1.

The effectiveness of metalearning systems heavily depends on the interaction between these two levels. Meta-level learning utilizes metaknowledge to guide the selection and configuration of base-level algorithms, identifying optimal solutions for new tasks based on accumulated experience. This process becomes particularly crucial in financial time series analysis, where the ability to adapt to changing market conditions is essential for maintaining predictive accuracy (Brazdil et al., 2022).

2.2 Evolution of Hyperparameter Optimization Techniques

Recent advances in HPO have introduced several sophisticated approaches that address the limitations of traditional methods. (Bischl et al., 2023) categorize these approaches into several key methodologies, each offering unique advantages for financial time series applications.

Adaptive Fidelity Identification (AFI), as discussed by (Jiang et al., 2024a), represents a significant advancement in HPO methodology. This approach dynamically determines appropriate fidelity levels

for evaluating hyperparameter configurations, leading to more efficient resource utilization. The method introduces concepts such as "efficient point" and "saturation point" to optimize the evaluation process. By adaptively selecting the most appropriate fidelity level for evaluation, this method allows for a more nuanced exploration of the hyperparameter space, potentially leading to faster convergence and better overall performance in financial forecasting applications.

Trajectory-Based Multi-Objective Optimization (Trajectory) has emerged as a promising approach for handling the complex trade-offs inherent in financial forecasting models. According to (Z. Wang et al., 2024) this method considers multiple objectives simultaneously while leveraging temporal information from the training process. By tracking model performance across multiple epochs for each hyperparameter setting, it creates a trajectory in the objective space.

Optuna is an advanced hyperparameter optimization framework specifically designed for machine learning applications, featuring an imperative, define-by-run style user API that enables dynamic construction of hyperparameter search spaces (Team, 2019).

The primary strengths of Optuna lie in its flexible architecture and efficient optimization capabilities. It define-by-run API provides high modularity and allows users to dynamically construct search spaces using familiar Python syntax, including conditionals and loops. The framework implements advanced pruning mechanisms to terminate unpromising trials early, significantly reducing computational overhead. Additionally, Optuna supports parallel distributed optimization, enabling simultaneous execution of multiple trials across different nodes, which is particularly valuable for computationally intensive deep learning applications.

3. Related work

The optimization of hyperparameters in time series analysis has garnered significant attention in recent years, with researchers exploring various approaches to enhance model performance while maintaining computational efficiency. This section examines key developments in hyperparameter tuning methodologies, particularly focusing on their application to time series forecasting and metalearning frameworks.

Several researchers have made substantial contributions to advancing hyperparameter optimization techniques. Bischl et al. (2023) provided a comprehensive framework for understanding hyperparameter optimization, establishing foundational principles and best practices that have shaped current approaches. Their work emphasized the importance of balancing exploration and exploitation in hyperparameter search spaces, particularly when dealing with the computational constraints inherent in time series analysis (Bischl et al., 2023).

In the context of metalearning, Huisman et al. (2021) conducted a thorough survey of deep metalearning approaches, highlighting how hyperparameter optimization plays a crucial role in model adaptation and generalization. Their research demonstrated that effective hyperparameter tuning strategies are essential for metalearning systems to successfully transfer knowledge across different time series tasks (Huisman et al., 2021).

Recent advances in adaptive optimization strategies have shown promising results. Jiang et al. (2024) introduced an innovative approach called Adaptive Fidelity Identification, which dynamically determines appropriate fidelity levels for evaluating hyperparameter configurations. This method has proven particularly effective in reducing computational overhead while maintaining model performance, addressing one of the key challenges in time series analysis (Jiang et al., 2024a).

The emergence of federated approaches to hyperparameter optimization has opened new possibilities for handling distributed time series data. Guo et al. (2022) demonstrated the effectiveness of federated hyperparameter optimization in multi-institutional settings, showing how such

approaches can maintain data privacy while achieving optimal model performance. Their work with Auto-FedRL exemplifies how federated learning principles can be successfully applied to hyperparameter optimization tasks (Guo et al., 2022).

Trajectory-based optimization methods have also shown significant promise. As documented by Wang et al. (2022), these approaches consider the temporal evolution of model performance during the optimization process, making them particularly well-suited for time series applications. Their research demonstrated how tracking performance trajectories across training epochs can lead to more robust hyperparameter configurations (Z. Wang et al., 2024).

The challenges of distribution shift in time series data have been specifically addressed by several researchers. Duan et al. (2022) proposed innovative solutions using hypernetworks to combat distribution shifts in time series forecasting, while Jati et al. (2023) introduced hierarchical proxy modeling for improved hyperparameter optimization in time series contexts. These approaches have demonstrated significant improvements in handling the non-stationary nature of time series data (Duan et al., 2022).

In terms of practical implementations, various frameworks have emerged to facilitate hyperparameter optimization. The Optuna framework, as discussed by Bischl et al. (2021), has gained prominence for its flexibility and efficiency in handling high-dimensional hyperparameter spaces. Similarly, the OptFormer approach represents a significant advancement in applying transformer-based architectures to hyperparameter optimization tasks (Bischl et al., 2023).

Recent work has also focused on addressing the computational challenges of hyperparameter optimization in time series analysis. Liu et al. (2021) provided a comprehensive survey of forecasting methods, highlighting the importance of efficient hyperparameter tuning in maintaining model performance while managing computational resources. Their work emphasized the need for scalable solutions that can handle the increasing complexity of modern time series applications (Liu et al., 2021).

The field continues to move toward more adaptive, efficient, and robust approaches that can handle the complexity of real-world time series applications while maintaining computational feasibility.

4. Methodology

4.1 Research Design

The research methodology follows a three-branch approach to evaluate hyperparameter optimization techniques in time series forecasting, as illustrated in Figure 2. This design enables comprehensive assessment through parallel investigation of model trainers, optimization techniques, and statistical validation.

The first branch focuses on model trainers, implementing distinct algorithms. The examples of such algorithms are Decision Tree, Random Forest, and XGBoost, there are 11 algorithms in total. These

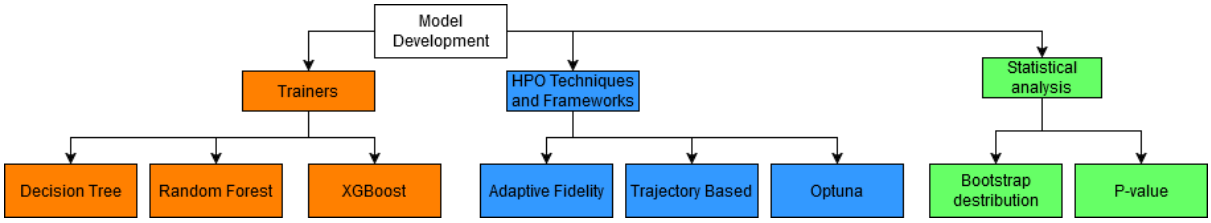

Figure 2: Model development workflow

algorithms were selected because currently BDO use those algorithms in their ClaireVoyance application.

The second branch encompasses the HPO techniques and frameworks under investigation. Three advanced optimization approaches are evaluated: Adaptive Fidelity, which dynamically adjusts evaluation precision; Trajectory Based optimization, which considers the temporal evolution of model performance; and Optuna, which employs a modern hyperparameter optimization framework. This selection represents a spectrum of contemporary optimization strategies, ranging from adaptive to trajectory-based approaches.

The third branch incorporates statistical analysis to ensure rigorous validation of results. Bootstrap distribution analysis is employed to assess the robustness and reliability of the optimization outcomes, while p-value calculations provide statistical significance measures for comparing different

optimization techniques. This statistical framework enables objective evaluation of performance differences between methods.

4.2 Tools

The study employs a range of advanced tools and frameworks to ensure rigorous implementation and evaluation of HPO strategies. Python, a widely used programming language in data science and machine learning, serves as the primary development environment. The study uses key libraries such as scikit-learn (Pedregosa et al., 2011), which provides a comprehensive suite of machine learning algorithms, and Optuna (Akiba et al., 2019) which offer powerful tools for systematic hyperparameter tuning. These optimization frameworks facilitate the exploration of high-dimensional parameter spaces and help identify configurations that maximize predictive accuracy and computational efficiency.

The study employs algorithms including decision tree (Kotsiantis, 2013), gradient boosting (Bentéjac et al., 2021), k-nearest neighbours (Grasmit & Akshay, 2021), majority voting, moving average, random forest (Schonlau & Zou, 2020), XGBoost (Chen & Guestrin, 2016), XGBoost pattern, average last year, majority selector, value last year. These models were selected for their versatility and proven effectiveness in handling financial time series data. Tools like Pandas and NumPy are used for data preprocessing, while visualization libraries such as Matplotlib (Hunt & Hunt, 2019) and Seaborn (Waskom, 2021) aid in the graphical representation of trends and results. By integrating these tools, the study ensures a balance between computational efficiency and analytical rigor, enabling the robust evaluation of hyperparameter optimization techniques.

The selection of hyperparameter optimization techniques and frameworks in this research was guided by a systematic evaluation of their applicability to meta-learning in time-series forecasting, with particular emphasis on computational efficiency, scalability, and alignment with current state-of-the-art practices. Adaptive Fidelity Identification (Jiang et al., 2024b) was chosen for its dynamic resource allocation capabilities, essential for managing the computational demands of meta-learning across multiple tasks, while Trajectory-Based Multi-Objective Optimization (W. Wang et al., 2024) was

selected for its ability to capture temporal dependencies in model performance, a crucial consideration for time-series applications. The implementation of these techniques is supported by complementary frameworks: Optuna for its flexible architecture and advanced pruning mechanisms. This comprehensive selection of methods and frameworks enables efficient exploration of hyperparameter spaces.

4.3 Data Collection Methods

The datasets used in this research was provided by the project organizers and is structured as a time series dataset with three primary columns: category, value, and date. The category column identifies the classification or grouping of each data point, the value column represents the observed numerical measurement, and the date column provides the time-based context. This structure enables the dataset to capture trends, patterns, and seasonal fluctuations essential in financial time series data (Table 1).

During the preprocessing phase the datasets were enriched with 29 additional columns to fit the trainers. The full list of added columns can be found in the Appendix B section.

Table 1: Data description

Dataset Name	Records	Features	Description
Day Data	1456	32	Daily financial data capturing detailed transactional trends.
Month Data	3137	32	Summarizing revenue, expenses, and key performance indicators.
Week Data	3028	32	Representing intermediate time-series granularity for trends and weekly performance.

Before analysis, the data underwent a preprocessing phase to ensure its suitability for machine learning applications. The dataset was enriched with based on ‘date’ column. The ‘date’ column was converted to a datetime format and set as the index, this will allow for easier time-based operations and analysis. Then the data was checked for missing values. Additionally, the dataset was dynamically splitted per category. These preprocessing steps are critical for maintaining data integrity and ensuring the validity of the results derived from following analyses.

4.4 Data Analysis Methods

The study employs a multifaceted approach to data analysis, incorporating both quantitative and qualitative methods to provide a comprehensive evaluation of hyperparameter optimization techniques.

Quantitative data analysis is primarily focused on assessing the performance of different hyperparameter optimization techniques through statistical and predictive metrics. Key metrics include Root mean square (RMSE), Mean absolute error (MAE) (De Myttenaere et al., 2016), and Coefficient of determination (R-squared) .

RMSE was chosen as a primary metric due to its sensitivity to large prediction errors, making it particularly relevant for financial forecasting where significant deviations can have substantial practical implications. RMSE's quadratic scoring rule penalizes larger errors more heavily than smaller ones, aligning with real-world financial risk assessment practices. MAE was included to provide a linear measure of error that is less sensitive to outliers than RMSE, offering a more robust assessment of model performance across different scales of prediction errors. The R-squared was selected to evaluate the models' ability to capture variance in the data, providing insight into how well the predictions follow the actual temporal patterns. While RMSE and MAE offer direct measures of prediction accuracy, R^2 complements these by quantifying the proportion of variance explained by the model, making it particularly valuable for assessing the quality of fit in time series data where trend and seasonality patterns are important. Together, these metrics provide a balanced evaluation framework that considers both the magnitude of prediction errors and the models' ability to capture underlying patterns in the data.

Qualitative data analysis complements the quantitative evaluation by exploring the optimization process's nuances. Thematic analysis is used to identify recurring patterns and themes in the optimization outcomes, such as computational bottlenecks or convergence challenges. This involves coding data from logs and parameter configurations to uncover insights into the behaviour of the optimization algorithms. By combining qualitative observations with quantitative findings, the study provides a comprehensive understanding of hyperparameter tuning in metalearning.

To integrate the results from quantitative and qualitative analyses, a mixed-methods approach is used. This involves contextualizing quantitative metrics with qualitative insights, such as interpreting anomalous results or understanding optimization trends. The integration ensures that the findings are not only statistically robust but also practically meaningful, offering actionable insights for improving hyperparameter optimization strategies.

4.5 Evaluation

The evaluation of hyperparameter optimization techniques is conducted using three key metrics: predictive accuracy, computational efficiency, and model robustness. A comprehensive assessment framework is employed to quantify the performance across these dimensions, enabling objective comparison between different HPO approaches.

Predictive Accuracy.

The accuracy of each HPO technique is assessed by calculating the RMSE and MAE of the resulting models. These metrics provide complementary views of prediction accuracy. RMSE is calculated as:

$$RMSE = \sqrt{\frac{1}{n}\sum(yi - \hat{y}i)^2}$$

Where y_i represents the actual value and $\hat{y}_i$represents the predicted value. RMSE penalizes larger errors more heavily due to the squaring operation, making it particularly suitable for financial forecasting where large prediction errors can be costly. MAE provides a linear measurement of error and is calculated as:

$$MAE = \frac{1}{n}\sum|yi - \hat{y}i|$$

Computational Efficiency. Computational efficiency is evaluated through two components: convergence time and resource utilization. The Time Score Component normalizes the computational time required by each HPO technique relative to a baseline grid search approach:

$$Time\ Score\ =\ 1\ -\frac{Time\ Taken}{Max\ Time\ Taken}$$

Model Robustness. Model robustness is evaluated using R-squared (R^2) score. R-squared is calculated as:

$$R^2 =\ 1\ -\frac{[\sum(yi\ -\ \hat{y}i)^2]}{[\sum(yi\ -\ \bar{y})^2]}$$

4.6 Overview of Experimental Setup

The experimental implementation was conducted within the Microsoft Azure Databricks environment, leveraging distributed computing capabilities for efficient data processing and model training. The computational infrastructure consisted of a cluster configuration with two worker nodes: one Standard_DS5_v2 node equipped with 16 cores and 56GB of memory, and another with 32 cores and 112GB of memory. The runtime environment utilized Databricks Runtime 16.0, incorporating Apache Spark 3.5.0 and Scala 2.12, providing a robust framework for large-scale data processing.

4.7 Hyperparameter space

The effectiveness of hyperparameter optimization techniques heavily depends on the quality and scope of the defined hyperparameter spaces. In this study, we conducted a comprehensive analysis and adjustment of hyperparameter spaces across different model trainers to ensure meaningful optimization opportunities while maintaining computational feasibility.

The trainers were categorized into three groups based on their hyperparameter space definitions. The first group comprised trainers with well-defined hyperparameter spaces, including Gradient Boosting, Random Forest, K-Neighbors Regressor, and XGBoost. These trainers featured carefully calibrated parameter ranges that provided sufficient optimization potential. For instance, XGBoost's configuration included meaningful ranges for critical parameters such as maximum depth [15, 30, 50] and learning rate [0.001, 0.01, 0.1], enabling effective model tuning.

The second group consisted of trainers with limited hyperparameter spaces, such as Moving Average and Value Last Years trainers. These trainers initially offered restricted optimization possibilities due to constrained parameter ranges or fixed options. For example, the Moving Average trainer was limited to basic parameters with fixed options for averaging methods.

The third group included trainers with suboptimal configurations, which required modification. These cases included redundant parameter settings and insufficient value ranges that could potentially limit the effectiveness of optimization efforts.

To address these variations and ensure consistent evaluation of HPO techniques, we implemented adjustments to the hyperparameter spaces. These modifications focused on expanding parameter ranges where appropriate to improve optimization efficiency, standardizing parameter definitions across similar model types, and ensuring all parameters had valid ranges for optimization.

The complete specifications of the adjusted hyperparameter spaces for each trainer are provided in the Appendix C section.

4.8 Statistical analysis

4.8.1 Bootstrap

Bootstrap (Léger et al., 1992) distribution analysis was implemented as a robust statistical method for evaluating the reliability and stability of our HPO techniques. This resampling approach involves repeatedly sampling the original dataset with replacement to create multiple synthetic datasets, enabling the estimation of statistical properties without assuming a specific underlying distribution. In this study, we employed 10,000 bootstrap iterations to generate reliable distributions of performance metrics (RMSE, MAE, and R^2) for each HPO method.

Bootstrap analysis provides reliable confidence intervals and statistical estimates without requiring assumptions about the underlying data distribution. Finally, this method is particularly valuable when working with limited data samples, as it enables robust statistical assumption.

4.8.2 P-value

P-value analysis was employed to quantitatively assess the statistical significance of performance differences between various HPO methods and the baseline Random Search approach. This metric provides a probabilistic measure of whether observed differences in performance could have occurred by chance, with values below 0.05 traditionally indicating statistical significance. This approach helps identify whether sophisticated HPO methods provide statistically significant improvements over simpler approaches, justifying their additional computational complexity. The combination of p-value analysis with bootstrap distributions provides a comprehensive statistical framework.

5. Results

5.1 Initial Result Analysis with Outliers

Table 2: Results with Outliers

Model Name	Mean RMSE ± std	Mean MAE ± std	Mean R2 ± std	Mean Total Time ± std
AFI	5964.74 ± 9016.76	4326.57 ± 7955.56	-21.76 ± 88.83	1.12 ± 1.27
Optuna	4941.53 ± 7576.24	3455.08 ± 5742.34	-15.88 ± 69.22	5.47 ± 5.82
Random search	4703.31 ± 7212.52	3274.86 ± 5400.67	-13.73 ± 59.7	4.9 ± 5.25
Trajectory	4967.69±7583.49	3467.05 ± 5783	-16.74 ± 73.51	5.31 ± 5.58

The preliminary analysis of model performance with outliers revealed substantial instability across all hyperparameter optimization methods, highlighting significant challenges in achieving reliable predictions. Table 2 presents the comprehensive results for each method, demonstrating the impact of outliers on various performance metrics.

The most striking observation is the extreme variability in prediction accuracy metrics. The Trajectory method exhibited particularly high standard deviations in both RMSE (±27583.49) and MAE (±5783) values, indicating severe instability in prediction performance. Similar patterns of high variability were observed across other methods, with AFI showing RMSE standard deviations of ±9016.76 and Optuna displaying variations of ±7576.24.

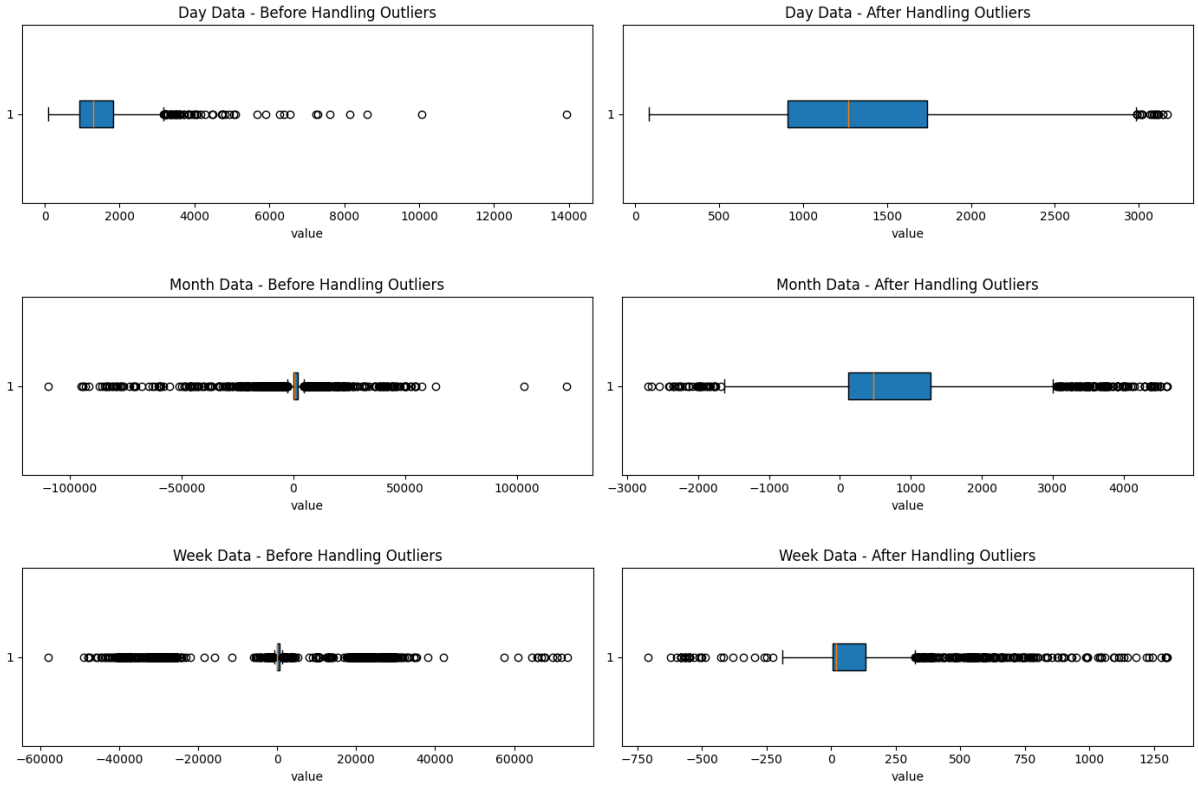

Figure 3: Handling data outliers

Model fit quality, as measured by R-squared, demonstrated consistently poor performance across all methods. The R^2 values ranged from -21.76 (AFI) to -13.73 (Random Search), with substantial standard deviations. These strongly negative values indicate that the models' predictions were less accurate

than a simple horizontal line representing the mean of the observed values, suggesting fundamental issues in capturing the underlying patterns in the presence of outliers.

Comparative analysis of performance metrics across methods revealed notable variations in prediction accuracy. AFI demonstrated the highest error rates with RMSE of 5964.74 and MAE of 4326.57, indicating the poorest prediction accuracy among all methods. Optuna showed moderate performance with RMSE of 4941.55 and MAE of 3455.08, while Random Search emerged as the most effective approach, achieving the lowest RMSE of 4703.31 and maintaining moderate MAE of 3574.15. The Trajectory method, despite its sophisticated approach, produced high error rates with RMSE of 4967.69 and MAE of 3467.05.

The computational efficiency metrics remained relatively stable across methods, suggesting that outliers primarily affected prediction accuracy rather than computational performance. However, the extreme variability in performance metrics made it impossible to draw meaningful conclusions about the relative effectiveness of different HPO approaches, necessitating proper outlier treatment for more reliable comparative analysis.

These initial results underscored the critical importance of outlier handling in time series analysis, particularly when evaluating sophisticated optimization techniques. The presence of outliers not only degraded model performance but also obscured the potential advantages of different HPO approaches.

5.2 Handling outliers

The analysis of data distributions across different temporal aggregations revealed significant outlier presence that required systematic treatment. Box plots were utilized to visualize and assess the distribution characteristics before and after outlier handling procedures.

At the daily aggregation level (Figure 3), the initial distribution exhibited extreme values extending up to 14,000, with numerous outliers scattered beyond the upper quartile. Post-processing, the distribution was effectively contained within a more reasonable range of 0-3,000, resulting in a more balanced interquartile range and improved data consistency.

The weekly data (Figure 3) demonstrated the most extreme outlier behaviour initially, with values spanning from -60,000 to +60,000. Following outlier treatment, the distribution was substantially normalized to a range of -750 to +1,250, representing a significant reduction in data volatility while preserving essential temporal patterns.

Monthly aggregated (Figure 3) data similarly displayed extensive outlier presence, initially ranging from -100,000 to +100,000. The outlier handling process successfully compressed this range to -3,000 to +4,000, maintaining the underlying temporal structure while eliminating extreme anomalies that could potentially distort subsequent analyses.

This systematic outlier treatment across all temporal granularities was crucial for ensuring robust model performance and reliable hyperparameter optimization. The processed distributions provided a more stable foundation for the comparative analysis of different HPO techniques, reducing the risk of optimization being unduly influenced by extreme values.

5.3 Result Analysis After Handling Outliers

Table 3: Results after handling outliers

Model Name	Mean RMSE ± std	Mean MAE ± std	Mean R2 ± std	Mean Total Time ± std
AFI	777. 77 ± 625.05	582.15 ± 517.24	-1.14 ± 1.82	1.1 ± 1.41
Optuna	582.12 ± 431.62	432.88 ± 368.09	-0.59 ± 1.69	4.99 ± 5.59
Random Search	579.24 ± 447.21	423.16 ± 366.46	-0.52 ± 2.43	3.28 ± 1.67
Trajectory	587.23 ± 433.59	432.9 ± 367.05	-0.58 ± 2.29	4.91 ± 5.8

Following the implementation of outlier handling procedures, the analysis revealed markedly improved stability and reliability across all hyperparameter optimization methods, as presented in Table 3. The results demonstrate substantial enhancements in both prediction accuracy and metric consistency, enabling more meaningful comparisons between different HPO approaches.

The Root Mean Square Error (RMSE) analysis showed significant improvement in both absolute values and stability. Random Search achieved the lowest RMSE of 579.24 (±447.21), followed closely by Trajectory HPO at 587.23 (±433.59) and Optuna at 583.12 (±431.62). AFI demonstrated relatively higher error rates with an RMSE of 777.77 (±625.05). Notably, the standard deviations decreased by an order of magnitude compared to the pre-treatment results, indicating substantially more consistent prediction performance across all methods.

Mean Absolute Error (MAE) measurements revealed similar patterns of improvement. Random Search maintained strong performance with the lowest MAE of 423.18 (±366.46), while Trajectory and Optuna showed comparable results with MAE values of 432.90 (±367.05) and 432.88 (±368.09) respectively. AFI exhibited higher error rates with MAE of 582.15 (±517.24). The reduced standard deviations in MAE values across all methods indicate more stable and reliable predictions.

The R-squared values, while remaining negative, showed marked improvement in consistency. Optuna achieved the least negative R^2 value of -0.59 (±1.69), followed closely by Trajectory at -0.58 (±2.29) and Random Search at -0.62 (±2.43). AFI demonstrated the poorest fit with an R-squared of -1.14 (±1.82). These values suggest persistent challenges in capturing all underlying patterns in the time series data, despite the improved stability in predictions.

Computational efficiency analysis revealed interesting trade-offs between performance and resource utilization. Random Search demonstrated the most efficient processing with a mean total time of 3.28 (±1.67) units, while Optuna and Trajectory required substantially more computational resources at 4.89 (±5.59) and 4.91 (±5.8) units respectively. AFI showed the most efficient runtime at 1.11 (±1.41) units, though this came at the cost of reduced prediction accuracy.

5.4 Statistical Analysis of HPO methods

A statistical analysis was conducted to evaluate the significance of performance (Table 4) differences between the proposed HPO methods and the baseline Random Search approach. The analysis employed a bootstrap resampling methodology with 10,000 iterations to ensure robust statistical inference. The null hypothesis posited no significant difference in performance metrics between methods, while the alternative hypothesis suggested superior performance of the proposed methods.

Table 4: Statistical Results

Metric	Method	P-Value
best_rmse_mean	Trajectory HPO	0.4550
best_rmse_mean	Optuna HPO	0.4946
best_rmse_mean	AFI HPO	0.0001
mae_mean	Trajectory HPO	0.4278
mae_mean	Optuna HPO	0.4192
mae_mean	AFI HPO	0.0019
r2_mean	Trajectory HPO	0.5696
r2_mean	Optuna HPO	0.5641
r2_mean	AFI HPO	0.9499

5.5.1 RMSE Analysis

The best RMSE metric analysis revealed no statistically significant differences for most methods. Trajectory HPO and Optuna HPO produced p-values of 0.4550 and 0.4946 respectively, indicating no significant improvement over the baseline Random Search. However, AFI HPO showed a statistically significant difference with a p-value of 0.0001, though this difference corresponded to worse performance as evidenced by higher error metrics.

5.5.2 MAE Performance

The Mean Absolute Error analysis showed similar patterns. Both Trajectory HPO and Optuna HPO demonstrated no significant differences from Random Search, with p-values of 0.4278 and 0.4192 respectively. AFI HPO again showed statistical significance ($p = 0.0019$), but this corresponded to inferior performance rather than improvement.

5.5.3 R-squared Metric Evaluation

The analysis of R-squared values revealed no statistically significant differences across all methods. Trajectory HPO (p = 0.5696), Optuna HPO (p = 0.5641), and AFI HPO (p = 0.9499) all showed p-values well above the conventional significance level of 0.05, indicating that none of the methods achieved significant improvements in model fit quality compared to Random Search.

5.5.4 Overview of statistical results

The statistical analysis reveals that sophisticated HPO methods did not provide statistically significant improvements over the baseline Random Search method. In fact, where statistical significance was observed (in the case of AFI HPO), it corresponded to degraded performance. These findings suggest that the additional complexity of advanced HPO methods may not justify their implementation in similar time series forecasting applications, particularly when considering their higher computational requirements. The consistency of these results across different performance metrics strengthens the conclusion that Random Search remains a competitive approach for hyperparameter optimization in this context.

6. Discussion

This research aimed to identify optimal strategies for hyperparameter optimization in time series forecasting while balancing prediction accuracy with computational efficiency. The experimental results challenged conventional assumptions about sophisticated HPO techniques, revealing that simpler approaches can be equally or more effective than complex methods.

The comparative analysis of HPO techniques produced several critical insights. Trajectory HPO achieved comparable performance to Random Search in terms of error metrics, with no statistically significant differences (p = 0.4550 for RMSE), but required substantially higher computational resources - approximately 4.5 time units compared to Random Search's 3.3 units. This trade-off aligns with findings from Bischl et al. (2023) regarding the computational complexity of advanced HPO methods.

AFI HPO, while showing statistically significant differences (p = 0.0001 for RMSE), demonstrated inferior prediction accuracy and higher error rates. Similarly, Optuna HPO showed no significant improvement over Random Search (p = 0.4946 for RMSE), despite its sophisticated optimization approach. These findings suggest that increased method complexity does not necessarily translate to better performance in time series forecasting applications.

The consistent occurrence of negative R-squared values across all methods suggests fundamental challenges in capturing the underlying patterns in financial time series data. This observation aligns with research by Duan et al. (2022) on distribution shifts in time series forecasting and indicates that the challenges may extend beyond hyperparameter optimization.

6.2 Methodological Implications

The preprocessing phase's effectiveness in handling outliers played a crucial role in establishing a stable foundation for HPO comparison. However, the persistent negative R^2 values suggest that even with robust preprocessing, capturing complex temporal dependencies in financial data remains challenging. This finding supports Guo et al.'s (2022) observations regarding the importance of data quality in financial forecasting.

The selection of evaluation metrics proved crucial in understanding the nuanced performance differences between methods. The comprehensive statistical analysis revealed that sophisticated HPO methods provide no significant advantages over Random Search, challenging assumptions about the necessity of complex optimization strategies.

6.3 Practical Implications

For practitioners in financial forecasting, these findings have several important implications. The competitive performance of Random Search, combined with its lower computational requirements, suggests it as a preferred choice for many applications, particularly in resource-constrained environments or real-time systems. The lack of statistically significant improvements from more complex methods indicates that organizations might not benefit from implementing sophisticated HPO strategies.

6.4 Limitations and Future Directions

A significant limitation of this study stems from the substantial variation in dataset sizes across different categories following the partitioning strategy. While some categories contained abundant data points enabling robust model training and evaluation, others were constrained by limited samples, potentially affecting the reliability of optimization outcomes. This uneven distribution of data points may have impacted the comparative assessment of HPO methods, particularly for sophisticated approaches that typically benefit from larger datasets to demonstrate their full capabilities.

The standardization of hyperparameter spaces across all optimization techniques, while ensuring fair comparison, may have accidentally limited the potential of individual HPO methods. Each optimization approach possesses unique characteristics that could be better leveraged through specifically custom-made hyperparameter spaces. The current implementation's standardised approach to hyperparameter space definition might have constrained the ability of more sophisticated methods to fully utilize their advanced optimization strategies.

These limitations point to several promising directions for future research. A key area of investigation would be the identification of optimal hyperparameter spaces specifically designed for each HPO technique, potentially revealing performance advantages not apparent with standardized configurations. Additionally, expanding the scope beyond regression tasks to include classification problems would provide broader insights about the data.

7. Conclusion

This study provides important insights into the effectiveness of various hyperparameter optimization approaches in financial time series forecasting. The comprehensive evaluation of Random Search, Trajectory HPO, Optuna HPO, and AFI HPO revealed that sophisticated optimization techniques do not necessarily outperform simpler approaches, particularly when considering both prediction accuracy and computational efficiency.

Our statistical analysis demonstrated no significant performance improvements from advanced HPO methods compared to Random Search, with Trajectory HPO and Optuna HPO showing comparable error metrics ($p > 0.05$) but requiring substantially higher computational resources. Where statistical significance was observed, as with AFI HPO ($p = 0.0001$), it corresponded to degraded rather than improved performance. These findings challenge the conventional wisdom that more complex optimization strategies inherently lead to better results.

The research identified several critical limitations, including variations in dataset sizes across categories and the potential constraints imposed by standardized hyperparameter spaces. These limitations suggest promising directions for future research, particularly in developing method-specific hyperparameter spaces and expanding the investigation to include classification tasks alongside regression problems.

The practical implications of this research are particularly relevant for organizations implementing time series forecasting systems. The competitive performance of Random Search, combined with its lower computational overhead, suggests it as a viable choice for many applications, especially in resource-constrained environments.

References

Vanschoren, J. (2019). Chapter 2 Meta-Learning.

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631. KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/3292500.3330701

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, *54*(3), 1937–1967. https://doi.org/10.1007/s10462-020-09896-5

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A., Deng, D., & Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowledge Discovery*, *13*(2), e1484. https://doi.org/10.1002/widm.1484

Brazdil, P., Van Rijn, J. N., Soares, C., & Vanschoren, J. (2022). Metalearning for Hyperparameter Optimization. In P. Brazdil, J. N. Van Rijn, C. Soares, & J. Vanschoren, *Metalearning* (pp. 103–122). Springer International Publishing. https://doi.org/10.1007/978-3-030-67024-5_6

Chatterjee, A., Bhowmick, H., & Sen, J. (2021). Stock Price Prediction Using Time Series, Econometric, Machine Learning, and Deep Learning Models. *2021 IEEE Mysore Sub Section International Conference (MysuruCon)*, 289–296. https://doi.org/10.1109/MysuruCon52639.2021.9641610

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/2939672.2939785

De Myttenaere, A., Golden, B., Le Grand, B., & Rossi, F. (2016). Mean Absolute Percentage Error for regression models. *Neurocomputing*, *192*, 38–48. https://doi.org/10.1016/j.neucom.2015.12.114

Duan, W., He, X., Zhou, L., Thiele, L., & Rao, H. (2022). *Combating Distribution Shift for Accurate Time Series Forecasting via Hypernetworks* (No. arXiv:2202.10808). arXiv. http://arxiv.org/abs/2202.10808

Grasmit, K., & Akshay, M. (2021). *K-Nearest Neighbour Algorithm*. https://www.semanticscholar.org/paper/K-Nearest-Neighbour-Algorithm-Grasmit-Akshay/03869fd1e17fd632be725fdf2d256184c7e84454

Guo, P., Yang, D., Hatamizadeh, A., Xu, A., Xu, Z., Li, W., Zhao, C., Xu, D., Harmon, S., Turkbey, E., Turkbey, B., Wood, B., Patella, F., Stellato, E., Carrafiello, G., Patel, V. M., & Roth, H. R. (2022). Auto-FedRL: Federated Hyperparameter Optimization for Multi-institutional Medical Image Segmentation. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, & T. Hassner (Eds.), *Computer Vision – ECCV 2022* (Vol. 13681, pp. 437–455). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-19803-8_26

Hospedales, T. M., Antoniou, A., Micaelli, P., & Storkey, A. J. (2021). Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. https://doi.org/10.1109/TPAMI.2021.3079209

Huisman, M., Van Rijn, J. N., & Plaat, A. (2021). A survey of deep meta-learning. *Artificial Intelligence Review*, *54*(6), 4483–4541. https://doi.org/10.1007/s10462-021-10004-4

Hunt, J., & Hunt, J. (2019). *Introduction to Matplotlib*. 35–42. https://doi.org/10.1007/978-3-030-25943-3_5

Jiang, J., Wen, Z., Mansoor, A., & Mian, A. (2024a). Efficient Hyperparameter Optimization with Adaptive Fidelity Identification. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 26181–26190. https://doi.org/10.1109/CVPR52733.2024.02474

Jiang, J., Wen, Z., Mansoor, A., & Mian, A. (2024b). Efficient Hyperparameter Optimization with Adaptive Fidelity Identification. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 26181–26190. 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/CVPR52733.2024.02474

Kotsiantis, S. B. (2013). Decision trees: A recent overview. *Artificial Intelligence Review*, *39*(4), 261–283. https://doi.org/10.1007/s10462-011-9272-4

Léger, C., Politis, D. N., & Romano, O. P. (1992). Bootstrap Technology and Applications. *Technometrics*, *34*(4), 378–398. https://doi.org/10.1080/00401706.1992.10484950

Liu, Z., Zhu, Z., Gao, J., & Xu, C. (2021). Forecast Methods for Time Series Data: A Survey. *IEEE Access*, *9*, 91896–91912. https://doi.org/10.1109/ACCESS.2021.3091162

[PDF] Coefficient of Determination | Semantic Scholar. (n.d.). Retrieved January 19, 2025, from https://www.semanticscholar.org/paper/Coefficient-of-Determination-Pyrczak-Oh/5b1a0e6dd248e67f33d85f076513ab8a193daa06

[PDF] Root-mean-square error (RMSE) or mean absolute error (MAE): When to use them or not | Semantic Scholar. (n.d.). Retrieved January 19, 2025, from https://www.semanticscholar.org/paper/Root-mean-square-error-(RMSE)-or-mean-absolute-when-Hodson/b1cf13d4ae3181fdf6dcce70caa4dea8965c68b8

[PDF] The p-value Function and Statistical Inference | Semantic Scholar. (n.d.). Retrieved January 19, 2025, from https://www.semanticscholar.org/paper/The-p-value-Function-and-Statistical-Inference-Fraser/89a93fd671bb2ba8aa69564afc5b54df43527fc8

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Louppe, G., Prettenhofer, P., Weiss, R., Weiss, R. J., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. https://www.semanticscholar.org/paper/Scikit-learn%3A-Machine-Learning-in-Python-Pedregosa-Varoquaux/168f28ac3c8c7ea63bf7ed25f2288e8b67e2fe74

Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal: Promoting Communications on Statistics and Stata*, *20*(1), 3–29. https://doi.org/10.1177/1536867X20909688

Team, O. (2019, September 25). Optuna: An Automatic Hyperparameter Optimization Framework. *Open Data Science Conference*. https://odsc.com/blog/optuna-an-automatic-hyperparameter-optimization-framework/

Wang, W., Fan, Z., & Ng, S. H. (2024). *Trajectory-Based Multi-Objective Hyperparameter Optimization for Model Retraining*. https://doi.org/10.48550/ARXIV.2405.15303

Wang, Z., Dahl, G. E., Swersky, K., Lee, C., Nado, Z., Gilmer, J., Snoek, J., & Ghahramani, Z. (2024). *Pre-trained Gaussian Processes for Bayesian Optimization* (No. arXiv:2109.08215). arXiv. http://arxiv.org/abs/2109.08215

Waskom, M. (2021). seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631. KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/3292500.3330701

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, *54*(3), 1937–1967. https://doi.org/10.1007/s10462-020-09896-5

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A., Deng, D., & Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowledge Discovery*, *13*(2), e1484. https://doi.org/10.1002/widm.1484

Brazdil, P., Van Rijn, J. N., Soares, C., & Vanschoren, J. (2022). Metalearning for Hyperparameter Optimization. In P. Brazdil, J. N. Van Rijn, C. Soares, & J. Vanschoren, *Metalearning* (pp. 103–122). Springer International Publishing. https://doi.org/10.1007/978-3-030-67024-5_6

Chatterjee, A., Bhowmick, H., & Sen, J. (2021). Stock Price Prediction Using Time Series, Econometric, Machine Learning, and Deep Learning Models. *2021 IEEE Mysore Sub Section International Conference (MysuruCon)*, 289–296. https://doi.org/10.1109/MysuruCon52639.2021.9641610

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/2939672.2939785

De Myttenaere, A., Golden, B., Le Grand, B., & Rossi, F. (2016). Mean Absolute Percentage Error for regression models. *Neurocomputing*, *192*, 38–48. https://doi.org/10.1016/j.neucom.2015.12.114

Duan, W., He, X., Zhou, L., Thiele, L., & Rao, H. (2022). *Combating Distribution Shift for Accurate Time Series Forecasting via Hypernetworks* (No. arXiv:2202.10808). arXiv. http://arxiv.org/abs/2202.10808

Grasmit, K., & Akshay, M. (2021). *K-Nearest Neighbour Algorithm*. https://www.semanticscholar.org/paper/K-Nearest-Neighbour-Algorithm-Grasmit-Akshay/03869fd1e17fd632be725fdf2d256184c7e84454

Guo, P., Yang, D., Hatamizadeh, A., Xu, A., Xu, Z., Li, W., Zhao, C., Xu, D., Harmon, S., Turkbey, E., Turkbey, B., Wood, B., Patella, F., Stellato, E., Carrafiello, G., Patel, V. M., & Roth, H. R. (2022). Auto-FedRL: Federated Hyperparameter Optimization for Multi-institutional Medical Image Segmentation. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, & T. Hassner (Eds.), *Computer Vision – ECCV 2022* (Vol. 13681, pp. 437–455). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-19803-8_26

Hospedales, T. M., Antoniou, A., Micaelli, P., & Storkey, A. J. (2021). Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. https://doi.org/10.1109/TPAMI.2021.3079209

Huisman, M., Van Rijn, J. N., & Plaat, A. (2021). A survey of deep meta-learning. *Artificial Intelligence Review*, *54*(6), 4483–4541. https://doi.org/10.1007/s10462-021-10004-4

Hunt, J., & Hunt, J. (2019). *Introduction to Matplotlib*. 35–42. https://doi.org/10.1007/978-3-030-25943-3_5

Jiang, J., Wen, Z., Mansoor, A., & Mian, A. (2024a). Efficient Hyperparameter Optimization with Adaptive Fidelity Identification. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 26181–26190. https://doi.org/10.1109/CVPR52733.2024.02474

Jiang, J., Wen, Z., Mansoor, A., & Mian, A. (2024b). Efficient Hyperparameter Optimization with Adaptive Fidelity Identification. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 26181–26190. 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/CVPR52733.2024.02474

Kotsiantis, S. B. (2013). Decision trees: A recent overview. *Artificial Intelligence Review*, *39*(4), 261–283. https://doi.org/10.1007/s10462-011-9272-4

Léger, C., Politis, D. N., & Romano, O. P. (1992). Bootstrap Technology and Applications. *Technometrics*, *34*(4), 378–398. https://doi.org/10.1080/00401706.1992.10484950

Liu, Z., Zhu, Z., Gao, J., & Xu, C. (2021). Forecast Methods for Time Series Data: A Survey. *IEEE Access*, *9*, 91896–91912. https://doi.org/10.1109/ACCESS.2021.3091162

[PDF] Coefficient of Determination | Semantic Scholar. (n.d.). Retrieved January 19, 2025, from https://www.semanticscholar.org/paper/Coefficient-of-Determination-Pyrczak-Oh/5b1a0e6dd248e67f33d85f076513ab8a193daa06

[PDF] Root-mean-square error (RMSE) or mean absolute error (MAE): When to use them or not | Semantic Scholar. (n.d.). Retrieved January 19, 2025, from

https://www.semanticscholar.org/paper/Root-mean-square-error-(RMSE)-or-mean-absolute-when-Hodson/b1cf13d4ae3181fdf6dcce70caa4dea8965c68b8

[PDF] The p-value Function and Statistical Inference | Semantic Scholar. (n.d.). Retrieved January 19, 2025, from https://www.semanticscholar.org/paper/The-p-value-Function-and-Statistical-Inference-Fraser/89a93fd671bb2ba8aa69564afc5b54df43527fc8

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Louppe, G., Prettenhofer, P., Weiss, R., Weiss, R. J., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. https://www.semanticscholar.org/paper/Scikit-learn%3A-Machine-Learning-in-Python-Pedregosa-Varoquaux/168f28ac3c8c7ea63bf7ed25f2288e8b67e2fe74

Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal: Promoting Communications on Statistics and Stata*, *20*(1), 3–29. https://doi.org/10.1177/1536867X20909688

Team, O. (2019, September 25). Optuna: An Automatic Hyperparameter Optimization Framework. *Open Data Science Conference*. https://odsc.com/blog/optuna-an-automatic-hyperparameter-optimization-framework/

Wang, W., Fan, Z., & Ng, S. H. (2024). *Trajectory-Based Multi-Objective Hyperparameter Optimization for Model Retraining*. https://doi.org/10.48550/ARXIV.2405.15303

Wang, Z., Dahl, G. E., Swersky, K., Lee, C., Nado, Z., Gilmer, J., Snoek, J., & Ghahramani, Z. (2024). *Pre-trained Gaussian Processes for Bayesian Optimization* (No. arXiv:2109.08215). arXiv. http://arxiv.org/abs/2109.08215

Waskom, M. (2021). seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Appendix A

Glossary

Term	Definition
Root Mean Square Error (RMSE)	A statistical measure that quantifies the standard deviation of the differences between predicted and observed values in a model. It is calculated by taking the square root of the mean squared differences, providing a scale-dependent measure of prediction accuracy.
Mean Absolute Error (MAE)	A metric that measures the average magnitude of errors between predicted and actual values, calculated by taking the average of the absolute differences. Unlike RMSE, it treats all errors with equal weight and maintains the same unit as the original data.
R-squared	A statistical measure representing the proportion of variance in a dependent variable that is predictable from independent variable(s). Values range from 0 to 1, where 1 indicates perfect prediction and 0 indicates the model explains none of the variability.
k-fold cross-validation	A resampling procedure used to evaluate machine learning models where the dataset is divided into k subsets. The model is trained on k-1 folds and validated on the remaining fold, with this process repeated k times to assess model performance across different data partitions.
Federate HPO	A distributed approach to hyperparameter optimization where multiple clients collaborate to find optimal model parameters while keeping their data localized, ensuring privacy and reducing communication overhead.
Adaptive Fidelity Identification	A method that dynamically adjusts the fidelity or resolution of system identification based on the required accuracy and computational resources, enabling efficient model development through adaptive sampling and evaluation.
Trajectory-Based Multi-Objective Optimization	An optimization approach that simultaneously considers multiple objectives by tracking the evolution of solution trajectories through the objective space, enabling the identification of Pareto-optimal solutions while maintaining search diversity.
Optuna	An open-source hyperparameter optimization framework that implements various sampling algorithms and pruning strategies to efficiently search high-dimensional parameter spaces, featuring automatic hyperparameter optimization through directed acyclic graphs.
Optformer frameworks	A transformer-based architecture designed for optimization tasks that learns to predict

	optimal hyperparameters by modelling the relationship between hyperparameter configurations and their corresponding performance metrics through attention mechanisms.
Data normalization	A preprocessing technique that rescales numeric features to a standard range, typically [0,1] or [-1,1], to ensure all features contribute proportionally to the model and improve convergence during training.
One-Hot Encoding	A data preprocessing method that converts categorical variables into binary vectors where each category becomes a new binary column, with a value of 1 indicating the presence of that category and 0 indicating its absence.
Data Preprocessing	The systematic transformation of raw data into a clean, structured format suitable for machine learning models, encompassing tasks such as handling missing values, scaling, encoding categorical variables, and feature engineering.
Decision Tree Model	A hierarchical model that makes predictions by recursively partitioning the feature space into regions based on feature thresholds, creating a tree-like structure of decision rules that can be easily interpreted.
Gradient Boosting	An ensemble learning technique that builds a sequence of weak learners (typically decision trees) where each subsequent model aims to correct the errors of its predecessors by optimizing along the gradient of the loss function.
K-Nearest Neighbours Model	A non-parametric algorithm that classifies or predicts based on the majority class or average value of the k closest training examples in the feature space, using a distance metric to determine similarity.
Majority Voting Model	An ensemble method that combines predictions from multiple models by selecting the most frequently predicted class (for classification) or averaging predictions (for regression) to make final decisions.
Moving Average Model	A time series analysis technique that predicts future values based on the weighted average of past observations, smoothing out short-term fluctuations to identify longer-term trends.
Random Forest Model	An ensemble learning method that constructs multiple decision trees using random subsets of features and bootstrap samples of the training data, combining their predictions to reduce overfitting and improve generalization.

XGBoost Model	An optimized implementation of gradient boosting that employs advanced techniques such as regularization, tree pruning, and parallel processing to achieve state-of-the-art performance in many machine learning tasks while maintaining computational efficiency.
P-value	A statistical measure that quantifies the probability of obtaining test results at least as extreme as the observed results, assuming that the null hypothesis is true. Values below a predetermined significance level (typically 0.05) suggest strong evidence against the null hypothesis.
T-test	A statistical hypothesis test used to determine whether there is a significant difference between the means of two groups or populations. It is particularly useful when working with normally distributed data and small sample sizes, providing a framework to assess whether observed differences are statistically meaningful.
Cross-fold Validation	A resampling procedure used to evaluate machine learning models where the dataset is divided into k subsets (folds). The model is iteratively trained on k-1 folds and validated on the remaining fold, with this process repeated k times to assess model performance across different data partitions, providing a robust estimate of model generalization capability.
Optuna	Optuna Framework
AFI	Adaptive Fidelity Identification
Trajectory HPO	Trajectory-Based Multi-Objective Optimization

Appendix B

List of added columns:

Year, quarter, month_number, week_of_year, day_of_year, day_of_week_number, is_month_start, is_month_end, is_quarter_start, is_quarter_end, is_year_start, is_year_end, is_holiday, is_working_day, day_of_week_number, first_workday_of_month, january_start, february_start, march_start, april_start, may_start, june_start, july_start, august_start, september_start, october_start, november_start, december_start

Appendix C

1. Trainer average last year

```
# allowed hyperparameters
    self.space_hyperparameters = {
```

```
    "avg_or_med": ["avg", "med"],
    "time_period": ["week", "month", "year"],
    "pattern": [True, False]
}
```

2. Trainer decision tree

```
# Allowed hyperparameters
    self.space_hyperparameters = {
    "max_depth": [5, 10, 20, 50, 100],
    "min_samples_split": [2, 5, 10],
    "prediction_mode": ["Zero", "AverageTrend"],
    "outlier_removal": [True]
}
```

3. Trainer gradient boosing

```
# Allowed hyperparameters
    self.space_hyperparameters = {
    "learning_rate": [0.001, 0.01, 0.05, 0.1, 0.2], # Expanded range for learning rates
    "n_estimators": [50, 100, 200, 300], # Added more options for the number of boosting rounds
    "subsample": [0.5, 0.75, 1], # Subsampling for stochastic boosting
    "min_samples_split": [2, 5, 10], # Broader range for splitting# Broader range for splitting
    "max_depth": [3, 6, 10, 15, 30], # Added more options for tree depth
    "prediction_mode": ["Zero", "AverageTrend"]
}
```

4. Trainer knn

```
# Allowed hyperparameters
    self.space_hyperparameters = {
    "n_neighbors": [1, 3, 5, 7, 10],  # Added more options for the number of neighbors
    "weights": ["uniform", "distance"],  # Included both weighting methods
    "algorithm": ["auto", "ball_tree", "kd_tree", "brute"],  # Added more algorithms
```

```
    "metric": ["euclidean", "manhattan", "minkowski"],  # Added more distance metrics
    "leaf_size": [10, 20, 30, 50],  # Expanded range for leaf size
    "p": [1, 2],  # L1 (Manhattan) and L2 (Euclidean) norms
    'outlier_removal': [True, False]
}
```

5. Trainer majority selector

```
# Allowed hyperparameters
    self.space_hyperparameters: dict[str, list[Any]] = {"": [""]}
```

6. Trainer moving average

```
# allowed hyperparameters
    self.space_hyperparameters = {
    "avg_or_med": ["avg", "med"],
    "time": ["week", "month", "year"],
    "pattern": [True, False],
    "outlier_removal": [True, False]
}
```

7. Trainer random forest

```
# Allowed hyperparameters
    self.space_hyperparameters = {
    "max_depth": [50, 100, 200, 300, 500], # More options for the number of trees
    "n_estimators": [5, 10, 20, 50, 100], # Added smaller and larger depth options
    "min_samples_split": [2, 5, 10], # Extended range
    "bootstrap": [True, False],
    "prediction_mode": ["Zero", "AverageTrend"],
    "outlier_removal": [True, False]
}
```

8. Trainer random forest pattern

```
# Allowed hyperparameters
        self.space_hyperparameters = {
            "n_estimators": [50, 100, 200, 300],
            "max_depth": [10, 20, 50, 100],
            "min_samples_split": [2, 5, 10],
            "bootstrap": [True, False],
            "prediction_mode": ["Zero", "AverageTrend"],
            "outlier_removal": [True, False]
        }
```

9. Trainer value last year

```
# Allowed hyperparameters
        self.space_hyperparameters: dict[str, list[Any]] = {
            "prediction_mode": ["Zero", "AverageTrend"],
            "outlier_removal": [True, False],
        }
```

10. Trainer XGBoost

```
# Allowed hyperparameters
        self.space_hyperparameters = {
            "objective": ["reg:squarederror", "reg:linear"],
            "n_estimators": [50, 100, 200, 300, 500],
            "max_depth": [3, 6, 10, 15, 30],
            "learning_rate": [0.001, 0.01, 0.05, 0.1, 0.2],
            "subsample": [0.5, 0.75, 1],  # Subsampling for stochastic gradient boosting
            "prediction_mode": ["Zero", "AverageTrend"],
            "outlier_removal": [True, False]
        }
```

11. Trainer XGBoost pattern

```
        # Allowed hyperparameters
            self.space_hyperparameters = {
```

```
    "objective": ["reg:squarederror", "reg:linear"],
    "n_estimators": [50, 100, 200, 300],
    "max_depth": [5, 10, 15, 30],
    "learning_rate": [0.01, 0.05, 0.1, 0.2],
    "subsample": [0.5, 0.75, 1],
    "prediction_mode": ["Zero", "AverageTrend"],
    "outlier_removal": [True, False]
}
```