COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MATHEMATICAL ENGINEERING AND ARTIFICIAL INTELLIEGENCE

BACHELOR THESIS

Hierarchical Models for Time Series Forecasting

Author: Enrique Campos Alonso

Director: Simón Rodríguez Santana

Madrid

June 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Hierarchical Models for Time Series Forecasting

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/25 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Enrique Campos Alonso Fecha: 11/ 06/ 2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Simón Rodríguez Santana Fecha: 17/06/2024

HIERARCHICAL MODELS FOR TIME SERIES FORECASTING

Author: Campos Alonso, Enrique.
Director: Rodríguez Santana, Simón.
ICAI – Universidad Pontificia Comillas

Abstract

Accurate, reliable and scalable predictions are essential in complex real-world scenarios (*e.g.* finance, energy and climate forecasting) where rapid decisions are critical. This study evaluates multiple models for short and long-term forecasting and studies how these predictions are enhanced by the usage of hierarchical structures in the modelling process. Experiments on synthetic and real-world data reveal that *DLMs* and *ARIMA* achieve the lowest point and probabilistic forecasting errors, but *DLMs* excel in prediction speed and model updates. Reconciliation methods leveraging past residuals improve coherence but degrade distributional accuracy. We propose a deployable framework combining *DLMs* for primary forecasts, *ARIMA* for validation, and *MinT Shrink LW* reconciliation for improved performance.

Keywords: Time-series forecasting, Machine Learning, ARIMA, Dynamic Linear Models, Hierarchical Reconciliation, MinT, S&P500.

1. Introduction

Time-series forecasting underpins decisions in in complex real-world scenarios (*e.g.* finance, energy and climate forecasting) by revealing trends that stabilize markets, optimize resources, and mitigate crises. Traditional point predictions are easy to compute but omit any sort of reliability metric on the forecast.

Uncertainty measures are crucial in this regard since these algorithms are commonly used to aid in decision-making processes. Thus, obtaining calibrate and trustworthy intervals in of utmost importance when dealing with sensitive issues. For instance, rregulators and security teams increasingly demand these calibrated predictive distributions. This study assesses models that natively produce forecast intervals and explores hierarchical reconciliation across related series to boost both accuracy and uncertainty quantification.

2. Thesis Scope and objectives

The scope will be implementing from scratch (in most cases) and making a comprehensive comparison between different forecasting models and finding the most suitable that meets requirements of accuracy, speed and reliability. Additionally, we will investigate the different reconciliation methods to comprehend its potential in situations where there are more relationships between the data. To achieve this, we have set the following objectives:

- **Implement forecasting and reconciliation techniques from scratch,** to provide the analysis with a reproducible pipeline to automatically fit each of the ML models.
- **Identify top-performing forecasting models**, within the requirements we set for our application of fast, reliable and accurate forecasts.

- **Characterize forecast uncertainty,** by using evaluation metrics defined as proper scoring rules.
- **Test Hierarchical Reconciliation Techniques,** evaluate different techniques performance and comprehending relationships between nodes within their scalability limitations.

3. Methodology

The first part of the project has been developing all forecasting models with a customized automatic model fit for some models Deep Learning models have been done with `pytorch`, while *ARIMA* is backed by `statsmodels`. After reviewing (West & Harrison, 1997) we implemented our version of *DLMs*. This makes a total of 7 forecasting models to compare: *DLM, ARIMA, SMA, EMA, MLP, LSTM* and *Conv1D*. These models were chosen since the application required to be lightweight and fast, and implementing big deep learning structures are inefficient in terms of reliability and scalability, even with results not are not always better.

The second part of the thesis has been developing the hierarchical reconciliation methods. For these experiments, we have built our own framework. This was done following the methods described in (Athanasopoulos, Hyndman, Kourentzes, & Panagiotelis, 2023) and (Wickramasuriya, Athanasopoulos, & Hyndman, 2018) for point forecast. Given that we want to provide reliable forecasts, we also implemented two probabilistic methods following the descriptions in (Panagiotelis, Gamakumara, Athanasopoulos, & Hyndman, 2023).

Lastly, we decided to set up two test environments. In the first case, we built a simple hierarchical structure with synthetic series at the lowest levels, that then were added to form the structure. Secondly, we tested our models in actual data from the S&P500 Index, building a hierarchy based on Index, Sector, Industry and Stock that would make a total of ~600 series. More interestingly, the relationship between these nodes based on market-cap weighting, allowing us to see how reconciliation matrices adapted to cases without straight aggregation.

4. Results

In this Section, we will provide the most important results from our experiments:

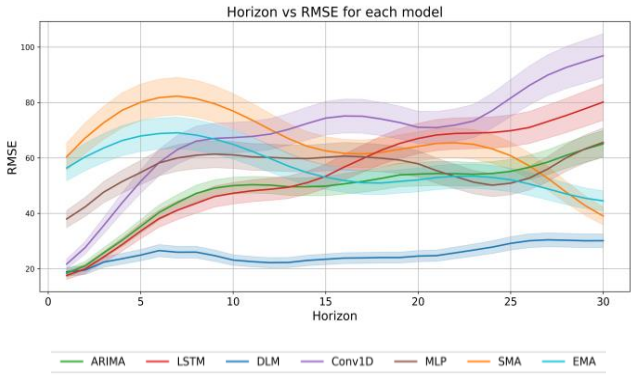

Figure 1 RMSE (+- standard error) against horizon

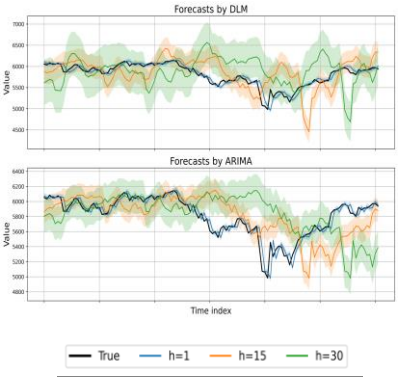

Figure 2 Forecast +- standard deviation for ARIMA and DLM

- The best results were achieved by DLMs and ARIMA models. Due to the DLM's scalability and reliability to a higher extent, we have chosen them as building block given our use case requirements.

Prob. Method	Rec. Method	Index	Sector	Industry	Stock
Base	Base	**178.40 ± 10.48**	**9.92 ± 0.59**	**11.91 ± 0.72**	12.96 ± 0.78
Gaussian	Classic	121791.84 ± 3986.63	668.98 ± 26.37	67.00 ± 3.35	12.96 ± 0.78
	Regression	8953.94 ± 287.90	58.50 ± 2.29	27.77 ± 1.16	12.96 ± 0.78
	MinT OLS	1267.84 ± 45.07	33.62 ± 1.20	27.88 ± 1.15	13.42 ± 0.79
	MinT Sample	488.35 ± 14.03	25.65 ± 0.88	28.54 ± 1.17	22.43 ± 0.88
	MinT Shrink	499.47 ± 13.90	25.80 ± 0.88	28.55 ± 1.16	22.27 ± 0.88
	MinT Shrink LW	451.22 ± 13.28	25.82 ± 0.87	28.50 ± 1.16	21.48 ± 0.85
	WLS_V	671.71 ± 29.59	31.12 ± 1.09	27.60 ± 1.15	13.48 ± 0.79
	WLS_S	5417.19 ± 172.04	59.40 ± 2.31	27.66 ± 1.15	16.74 ± 1.00
Monte Carlo	Classic	166352.57 ± 5257.51	823.19 ± 33.12	71.59 ± 3.66	**12.47 ± 0.68**
	Regression	10580.25 ± 323.58	58.85 ± 2.35	26.98 ± 1.16	**12.47 ± 0.68**
	MinT OLS	1199.39 ± 44.23	31.35 ± 1.14	26.96 ± 1.14	12.80 ± 0.68
	MinT Sample	438.99 ± 12.69	23.22 ± 0.81	27.46 ± 1.15	21.01 ± 0.82
	MinT Shrink	449.04 ± 12.61	23.28 ± 0.81	27.49 ± 1.15	20.74 ± 0.81
	MinT Shrink LW	403.59 ± 11.84	23.28 ± 0.81	27.47 ± 1.14	19.92 ± 0.78
	WLS_V	616.37 ± 27.96	28.97 ± 1.03	26.69 ± 1.13	12.88 ± 0.69
	WLS_S	5544.50 ± 173.08	56.75 ± 2.26	26.71 ± 1.14	15.88 ± 0.90

Figure 2 CRPS for probabilistic reconciliation across levels of hierarchy in S&P500

- Hierarchical reconciliation results have been positive on the synthetic case. In the financial dataset, base forecasts have been fitted more precisely due to the small number of samples. The best reconciliation technique has been *MinT Shrink LW* since it partially solves the issue by estimating the covariance matrix. *Monte Carlo* has been chosen over *Gaussian* given the improved performance and accepting computation cost.

5. Conclusions

In conclusion, we have studied different forecasting models along reconciliation techniques in terms of accuracy, reliability and scalability. In that sense, we have seen the *DLM* as the most appropriate model for the case, followed closely by the ARIMA. They will both as cross-checkers predictors in the case study framework. However, reconciliation techniques have shown its potential in the synthetic example but lacked data for the S&P500 case. We still believe in its use for the case study and recommend *MinT Shrink LW* and *Monte Carlo*, with a robust hierarchical structure.

6. References

Athanasopoulos, G., Hyndman, R. J., Kourentzes, N., & Panagiotelis, A. (2023). Forecast reconciliation: A review. Retrieved from https://robjhyndman.com/papers/hf_review.pdf

Panagiotelis, A., Gamakumara, P., Athanasopoulos, G., & Hyndman, R. J. (2023). Probabilistic forecast reconciliation: Properties, evaluation and score optimisation. *European Journal of Operational Research, 306*(2), 693-706. doi:https://doi.org/10.1016/j.ejor.2022.07.040

West, M., & Harrison, J. (1997). *Bayesian Forecasting and Dynamic Models* (2 ed.). New York: Springer. doi:https://doi.org/10.1007/b98971

Wickramasuriya, S. L., Athanasopoulos, G., & Hyndman, R. J. (2018). Optimal Forecast Reconciliation for Hierarchical and Grouped Time Series Through Trace Minimization. *Journal of the American Statistical Association, 114*(526), 804-819. doi:https://doi.org/10.1080/01621459.2018.1448825

MODELOS JERÁRQUICOS PARA PREDICCIÓN DE SERIES TEMPORALES

Autor: Campos Alonso, Enrique.
Director: Rodríguez Santana, Simón.
ICAI – Universidad Pontificia Comillas

Resumen

Predicciones precisas, fiables y escalables son fundamentales en escenarios reales complejos (por ejemplo, en finanzas, energía y predicción climática), donde la toma de decisiones rápida resulta crucial. Este estudio evalúa múltiples modelos para la predicción a corto y largo plazo, y analiza cómo dichas predicciones se ven mejoradas mediante el uso de estructuras jerárquicas en el proceso de modelado. Los experimentos realizados con datos sintéticos y reales demuestran que los modelos *DLM* y ARIMA obtienen los menores errores, tanto puntuales como probabilísticos, si bien los DLM destacan por su rapidez de predicción y capacidad de actualización. Los métodos de reconciliación que aprovechan residuos pasados mejoran la coherencia, pero reducen la precisión de la distribución. Se propone un marco de implementación que combina DLM para las predicciones principales, ARIMA para validación y reconciliación MinT shrink para optimizar el rendimiento.

Palabras clave: Prediccion series temporales, Machine Learning, ARIMA, Modelos Dinamicos Lineales, Concicliación jerárquica, MinT, S&P500.

1. Introducción

La previsión de series temporales sustenta la toma de decisiones en escenarios reales complejos (como las finanzas, la energía y la predicción climática) al revelar tendencias que contribuyen a estabilizar los mercados, optimizar recursos y mitigar crisis. Las predicciones puntuales tradicionales son fáciles de calcular, pero carecen de métricas de fiabilidad asociadas a la previsión.

Las medidas de incertidumbre resultan fundamentales, dado que estos algoritmos se utilizan habitualmente como apoyo en los procesos de toma de decisiones. Por ello, obtener intervalos calibrados y fiables es de suma importancia cuando se abordan cuestiones sensibles. Por ejemplo, los reguladores y los equipos de seguridad exigen cada vez más distribuciones predictivas correctamente calibradas. Este estudio evalúa modelos que generan de forma nativa intervalos de previsión y analiza la conciliación jerárquica entre series relacionadas para mejorar tanto la precisión como la cuantificación de la incertidumbre.

2. Enfoque de tesis y objetivos

El alcance de este trabajo consistirá en implementar desde cero (en la mayoría de los casos) y realizar una comparación exhaustiva entre distintos modelos de previsión, con el fin de identificar el más adecuado en términos de precisión, velocidad y fiabilidad. Adicionalmente, se investigarán los diversos métodos de conciliación para comprender su potencial en contextos donde existan mayores interrelaciones entre los datos. Para ello, se establecen los siguientes objetivos:

- **Implementar técnicas de previsión y conciliación desde cero**, a fin de dotar el análisis de una canalización reproducible que permita ajustar automáticamente cada uno de los modelos de aprendizaje automático.

- **Identificar los mejores modelos de previsión**, de acuerdo con los requisitos establecidos para nuestra aplicación: predicciones rápidas, fiables y precisas.
- **Caracterizar la incertidumbre de las previsiones**, utilizando métricas de evaluación definidas como reglas de puntuación adecuadas (*proper scoring rules*).
- **Probar técnicas de conciliación jerárquica**, evaluar el rendimiento de los distintos métodos y comprender las relaciones entre nodos, considerando sus limitaciones de escalabilidad.

3. Metodología

La primera parte del proyecto ha consistido en desarrollar todos los modelos de previsión, incorporando un ajuste automático personalizado para algunos de ellos. Los modelos de *Deep Learning* se implementaron con PyTorch, mientras que ARIMA se basó en la librería Statsmodels. Tras revisar (West & Harrison, 1997), se desarrolló nuestra propia versión de los DLM. En total, se compararon siete modelos de previsión: DLM, ARIMA, SMA, EMA, MLP, LSTM y Conv1D. Estos modelos se seleccionaron porque la aplicación debía ser ligera y rápida, y la implementación de arquitecturas de *Deep Learning* de gran tamaño resulta ineficiente en términos de fiabilidad y escalabilidad, además de que los resultados no siempre son superiores.

La 2ª de la tesis se centró en el desarrollo de los métodos de conciliación jerárquica. Para estos experimentos se programaron, siguiendo los métodos descritos en (Athanasopoulos, Hyndman, Kourentzes, & Panagiotelis, 2023) y (Wickramasuriya, Athanasopoulos, & Hyndman, 2018) para previsiones puntuales. Como se buscaba proporcionar previsiones fiables, incluimos dos métodos probabilísticos.

Finalmente, se definieron dos entornos de prueba. En primer lugar, se construyó una estructura jerárquica sencilla con series sintéticas en los niveles más bajos, que posteriormente se agregaron para conformar la jerarquía. En segundo lugar, se evaluaron los modelos con datos reales del índice S&P 500, construyendo una jerarquía basada en Índice, Sector, Industria y Acción, que totalizó aproximadamente 600 series. Un aspecto especialmente interesante es que la relación entre estos nodos se basó en la ponderación por capitalización bursátil, lo que permitió analizar cómo se adaptaban las matrices de conciliación en situaciones sin una agregación directa.

4. Resultados

En esta sección mostramos los resultados principales de los experimentos:

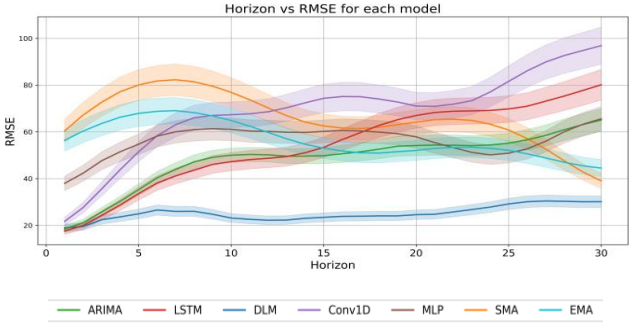

Figura 1 RMSE (+- error estándar) contra horizonte

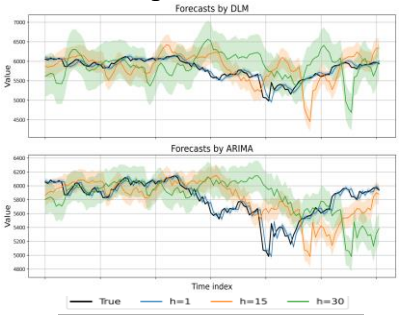

Figura 2 Predicción +- desviación típica para ARIMA y DLM

- Los mejores resultados se obtuvieron con los modelos *DLM* y *ARIMA*. Dado que los *DLM* destacan en escalabilidad y fiabilidad en mayor medida, se han seleccionado como componente principal, de acuerdo con los requisitos de nuestro caso de uso

Prob. Method	Rec. Method	Index	Sector	Industry	Stock
Base	Base	**178.40 ± 10.48**	**9.92 ± 0.59**	**11.91 ± 0.72**	12.96 ± 0.78
Gaussian	Classic	121791.84 ± 3986.63	668.98 ± 26.37	67.00 ± 3.35	12.96 ± 0.78
	Regression	8953.94 ± 287.90	58.50 ± 2.29	27.77 ± 1.16	12.96 ± 0.78
	MinT OLS	1267.84 ± 45.07	33.62 ± 1.20	27.88 ± 1.15	13.42 ± 0.79
	MinT Sample	488.35 ± 14.03	25.65 ± 0.88	28.54 ± 1.17	22.43 ± 0.88
	MinT Shrink	499.47 ± 13.90	25.80 ± 0.88	28.55 ± 1.16	22.27 ± 0.88
	MinT Shrink LW	451.22 ± 13.28	25.82 ± 0.87	28.50 ± 1.16	21.48 ± 0.85
	WLS_V	671.71 ± 29.59	31.12 ± 1.09	27.60 ± 1.15	13.48 ± 0.79
	WLS_S	5417.19 ± 172.04	59.40 ± 2.31	27.66 ± 1.15	16.74 ± 1.00
Monte Carlo	Classic	166352.57 ± 5257.51	823.19 ± 33.12	71.59 ± 3.66	**12.47 ± 0.68**
	Regression	10580.25 ± 323.58	58.85 ± 2.35	26.98 ± 1.16	**12.47 ± 0.68**
	MinT OLS	1199.39 ± 44.23	31.35 ± 1.14	26.96 ± 1.14	12.80 ± 0.68
	MinT Sample	438.99 ± 12.69	23.22 ± 0.81	27.46 ± 1.15	21.01 ± 0.82
	MinT Shrink	449.04 ± 12.61	23.28 ± 0.81	27.49 ± 1.15	20.74 ± 0.81
	MinT Shrink LW	403.59 ± 11.84	23.28 ± 0.81	27.47 ± 1.14	19.92 ± 0.78
	WLS_V	616.37 ± 27.96	28.97 ± 1.03	26.69 ± 1.13	12.88 ± 0.69
	WLS_S	5544.50 ± 173.08	56.75 ± 2.26	26.71 ± 1.14	15.88 ± 0.90

Tabla 2 CRPS para conciliación probabilística reconciliation en la jerarquía del S&P500.

- Los resultados de la conciliación jerárquica han sido positivos en el caso sintético. En el conjunto de datos financieros, las previsiones base se han ajustado con mayor precisión debido al reducido número de muestras. La técnica de conciliación que ha mostrado mejor rendimiento ha sido *MinT Shrink LW*, ya que solventa parcialmente el problema mediante la estimación de la matriz de covarianzas. Se ha optado por el método *Monte Carlo* en lugar del gaussiano, dado su mejor desempeño, asumiendo el coste computacional asociado.

5. Conclusiones

En conclusión, se han estudiado distintos modelos de previsión junto con técnicas de conciliación, evaluándolos en términos de precisión, fiabilidad y escalabilidad. En este sentido, se ha identificado el *DLM* como el modelo más adecuado para el caso de estudio, seguido de cerca por el *ARIMA*. Ambos se utilizarán como predictores complementarios en el marco de trabajo propuesto. No obstante, las técnicas de conciliación han demostrado su potencial en el ejemplo sintético, pero se ha visto limitada su aplicación en el caso del S&P 500 debido a la escasez de datos. Aun así, se mantiene la confianza en su utilidad para el estudio y se recomienda emplear *MinT Shrink LW* y *Monte Carlo*, siempre que se cuente con una estructura jerárquica robusta.

6. Referencias

Athanasopoulos, G., Hyndman, R. J., Kourentzes, N., & Panagiotelis, A. (2023). Forecast reconciliation: A review. Retrieved from https://robjhyndman.com/papers/hf_review.pdf

Panagiotelis, A., Gamakumara, P., Athanasopoulos, G., & Hyndman, R. J. (2023). Probabilistic forecast reconciliation: Properties, evaluation and score optimisation. *European Journal of Operational Research, 306*(2), 693-706. doi:https://doi.org/10.1016/j.ejor.2022.07.040

West, M., & Harrison, J. (1997). *Bayesian Forecasting and Dynamic Models* (2 ed.). New York: Springer. doi:https://doi.org/10.1007/b98971

Wickramasuriya, S. L., Athanasopoulos, G., & Hyndman, R. J. (2018). Optimal Forecast Reconciliation for Hierarchical and Grouped Time Series Through Trace Minimization. *Journal of the American Statistical Association, 114*(526), 804-819. doi:https://doi.org/10.1080/01621459.2018.1448825

Report Index

Figure Index

Tables Index

1. INTRODUCTION

Time series forecasting is a classic machine learning domain that plays a critical role in identifying trends and patterns from temporal data. It underpins critical decision-making where these trends drive social and economic outcomes. In climate and energy forecasting, for example, accurate demand forecasts help balance supply, stabilize prices or reduce carbon emissions. In Spain, Red Electrica reports national forecasts over short-, medium- and long-term horizons to adjust market prices and anticipate needs for different scenarios (Red Electrica, 2025). This proves that having a high-performance model gives a competitive advantage and can provide fair electricity to clients.

Although point estimates give a good initial approach, there is no real measure of how trustworthy that prediction is. There is a demand for understanding forecast risk. For instance, financial regulators require well-calibrated tail risk measures to protect against crises (Basel Committe on Banking Supervision, 2017). Or cybersecurity teams must weigh the probability of a rare but costly breach when protecting a system (Naveiro, Rodríguez, & Ríos Insúa, 2018). However, predictive distributions are more costly and complicated to predict that just point forecasts, so researchers must define a trade-off for how much value they can gain from them. In most applications, the highest value in these distributions is that they enable risk-aware choices, allowing us to compare scenario probabilities or set dynamic safety margins.

In Section 3.1 and 3.2 we will explore some forecasting models that, by nature, allow us to predict these intervals and compare if we lose any forecasting ability in the process. And finally, in Section 3.3 and 3.4, we will explore hierarchical forecasting techniques that integrate information across distinct levels, such as individual assets in aggregated portfolios or regional and national energy grids. These models should allow us to capture dependencies between series and improve point forecasts and uncertainty calibration.

1.1 CONTEXT AND MOTIVATION

In financial institutions, centralized data servers and streamlined ETL pipelines are key for feeding risk-forecasting models with some market information needed to monitor and manage exposure across portfolios (AWS; JPMorgan Chase, 2022). Complex dependencies among instruments—whether equities, derivatives, or fixed-income products—must be modelled not only to enhance point-forecast accuracy but, to produce calibrated risk measures to support decision-making.

At the same time, rising data volumes introduce a trade-off between model complexity and scalability. As the dataset grows, training and inference times can lengthen to the point where forecasts arrive too late. Managing computational resources efficiently to provide accurate and risk-aware predictions is a central challenge. It demands scalable architectures, with incremental learning techniques and optimal information extraction from features. Therefore, building hierarchical structures could be an efficient way of improving predictions.

The market for these technologies is huge, For instance, High Frequency Trading (HFT) accounts for 30% - 40% of total daily equity trades in European and American markets (Grand View Research, 2024). Forecasting has also become one of the most profitable segments of Machine Learning. As per (Fortune Business Insights, 2025), the Big Data Analytics market was valued at $350bn in 2024, with high growth expectations. These numbers prove the global interest in finding methods that allow for fast, accurate and reliable predictions to ultimately increase profits.

1.2 OBJECTIVES

To answer to these challenges, we have listed a set of main objectives to gain the most understanding of the topic, and some secondary goals to give full context of implementation in a professional setting.

- **Implement forecasting and reconciliation techniques from scratch,** to provide the

analysis with a reproducible pipeline to automatically fit each of the ML models.

- **Identify top-performing forecasting models**, within the requirements we set for our application of fast, reliable and accurate forecasts.
- **Characterize forecast uncertainty,** by using evaluation metrics defined as proper scoring rules.
- **Test Hierarchical Reconciliation Techniques,** evaluate different techniques performance and comprehending relationships between nodes within their scalability limitations.

And one secondary objectives:

- **Design a practical Implementation infrastructure,** by choosing the best combination of models and techniques, to provide a case study where hierarchical data is available.

All of this will be analysed from the perspective of accuracy, reliability and fast forecasting.

1.3 Project Structure

The project has been structured to fulfil these objectives. Initially, Section 2 will give an overview of several state-of-the-art methods used in time series forecasting. Following this, we will dive into the most recent research done in hierarchical reconciliation.

The next Section, 3, will be a thorough explanation of all the methods that will be implemented for the experiments. They will be divided into classic machine learning and deep learning forecasting models, and another subsection for the reconciliation methods, including the probabilistic techniques.

In Section 4 we will assess all the different methods documented in Section 3. The first experiment will be done on synthetic data, with the objective of understanding how the different forecasting models work. Then, we will choose the best model for an application focused on speed, accuracy and scalability, and comprehend the reconciliation techniques. The second experiment will follow the same idea, but in financial data from the S&P500 Index. We will evaluate the results of the forecasting and reconciliation methods too.

Lastly, Section 5 will serve as a posterior analysis of the project in the business world. There, we will devise how it could be implemented in a real-world application, where the methods used have come from the main conclusions from the research.

1.4 Planification and Economic Feasibility

The process followed for the project started by overviewing the state-of-the-art research on the topic. Once we had a clear scope, we implemented the initial models and reconciliation techniques in Python. This was particularly challenging, as our results needed to be in line with results from specialised frameworks. Also, finding the correct datasets was quite challenging since free financial data is scarce and other applications did not fulfil our requirements. Lastly, we proceeded with the analysis of the results.

Economically, this project has not required many resources besides the technological hardware and the open-source data. At an industrial level, it would be appropriate to invest in top tier hardware since computations take long and there is a lot of data to process. Another big issue is finding sources with free financial data, which are scares and limited. Specially in the case we wanted to use real-time data, we would probably need to have Bloomberg Terminal (annual fee of $28k) or, more economical, FactSet (annual fee of $12k) (Wall Street Prep, 2025).

2. State of the Art

Historically, time series forecasting was predominantly performed with moving averages

and linear regression. Soon, those methods were quickly replaced by exponentials by the 1950s. It was not until the 1960s and 70s that some of the most well-known statistical models were developed (Wong, 2024). In that decade, (Box & Jenkins, 1976) formalized the ARIMA (p, d, q) model and extended it to SARIMA. Moreover, (West & Harrison, 1997) textbooks leaded the development of Bayesian Dynamic Models (DLMs). And more recently, Prophet (Taylor & Letham, 2017) rose as a popular alternative in the family of DLMs, given its ability to handle seasonality and holiday days.

The biggest revolution post ARIMA arrived with the new Artificial Intelligence era. In the early 2000s MLPs were already gaining popularity as hardware components became more efficient (Wong, 2024). This continued with the Deep Learning Era with the arrival of LSTMs (Hochreiter & Schmidhuber, 1997), that gathered a lot of attention in the upcoming decade. The rise of use of CNNs for image processing created a new application of 1D convolutional networks for time series forecasting, with posterior research into Temporal Convolutional Networks (Lea, Vidal, Reiter, & Hager, 2016). Attention mechanisms were evaluated in (Vaswani, et al., 2023) being the latest model architectures and effective in other sequence-to-sequence tasks like NLP. Now, it seems that research is being done on probabilistic AI, multivariate time series forecasting and hierarchical reconciliation (Wong, 2024), which will be overviewed later.

Since time-series forecasting has been a popular machine learning task, there is some research done on comparing different models. Although these types of analysis are quite dependable on the type of data being used, one of the most complete comparisons we found is in (Tchoketch-Kebir & Madouri, 2024) for economic forecasting. Results have shown that modern deep learning models tend to outperform classic ARIMA processes, making a noticeable error increase in long-term forecast. Specifically, a simple MLP showed to be the best in short-term forecasts, and the LSTM in longer-term. However, we missed an analysis and comparison in training and inference times between the models.

As mentioned earlier, another important focus of this investigation is hierarchical structures and their subsequent reconciliation methods. One of the most relevant events these methods where used was the M5 (Makridakis) competition, with a hierarchical dataset provided by Walmart (Makridakis, Spiliotis, & Assimakopoulos, 2022). As per their conclusions on the competition, a combination of bottom-up and top-down worked better than forecasting series separately.

Figure 1 Example of hierarchical structure (Palande & Recasens, 2019)

This field of time-series forecasting assumes there is a dependency tree between several series and aims to improve predictions by using those relationships in the data. In Figure 1 we can see an example, where bottom level sales series could more reliable than top level, and reconciliation is used to identify selling trends in "Buckhead" and "Evanston" to improve forecasts. Classically, there were three main methods: Bottom-Up (BU), Middle-Out (MO) and Top-Down (TD) reconciliation (Hyndman & Athanasopoulos, 2021). However, they leave a lot of questions unanswered as they assume a lot about the nature behind the data. For example, BU directly adds each time series as equal to form the "parent", although this does not need to be strictly true (*e.g.*, ETFs, where individual stocks form the ETF with different weights). TD has a similar issue concerning what weight to assign to each "child" series. A review of the methods is found in

(Athanasopoulos, Hyndman, Kourentzes, & Panagiotelis, 2023)

However, reconciliation techniques delved deeper into how to data related to each other. Minimum Trace (MinT) optimal reconciliation (Wickramasuriya, Athanasopoulos, & Hyndman, 2018) was created by minimizing the total variance of the reconciliation errors from the historical series. It assumes that the base forecasts are unbiased and that the covariance matrix is known, although in cases where it is not available, the paper provides different methods to estimate it.

However given our interest in predicting a probability distribution, we must include several methods to expand point forecast to a complete distribution. Typically, this could be done by projecting the base probabilistic forecasts with the summing matrix, but there are other methods. Under a Gaussian assumption (Quinn, Corliss, & Povinelli, 2024), this projection admits a closed-form solution that minimizes the trace of the reconciled covariance matrix, just as in MinT, but generalizing to full predictive densities. Other methods that do not follow Gaussian behaviour rely on drawing base-forecast samples and then applying constrained importance sampling (Panagiotelis, Gamakumara, Athanasopoulos, & Hyndman, 2023), this ensures that every simulated path lies in the coherent manifold, giving marginal and joint predictive intervals that are both calibrated and coherent.

2.1 CONTRIBUTION

The value proposal in this project is giving a holistic comparison in time-series forecasting where the environment requires a fast, accurate and reliable method. Also, it will help understand the advantages of grouping series and therefore extracting the benefits of hierarchical reconciliation to improve base forecasts. This will expand the current research in terms of adding a time variable to explore, both in forecasting and reconciliation, while developing the custom architecture.

3. METHODOLOGY

This Section will give a thorough explanation into all the models and reconciliation techniques implemented in Python by us, along some frameworks like `statsmodels` or `pytorch`. We aim to make fitting process as automatic as possible. All code is provided in the following GitHub Repository:
https://github.com/HenryECA/DLM_Model

3.1 STATISTICAL METHODS

3.1.1 SIMPLE MOVING AVERAGE (SMA)

SMA is a simple model that computes each forecast as the unweighted mean of the most recent k observations. For the window k:

$$f_t = \frac{1}{k}\sum_{i=0}^{k-1} y_{t-1}$$

The hyperparameter k controls the window size. The higher it is, the smoother the predictions will be. In practice, this parameter could be found with Grid Search or visually assessing the predictions' smoothness. We have used 15 as the window size, half the horizon.

The main benefit of this model is its simplicity, and selecting k is easy. However, it has many pitfalls. When producing k-step ahead forecast, the following step will always follow the same trend. Also, it has a slow reaction to changes in the series.

3.1.2 EXPONENTIAL SMOOTHING (EMA)

Exponential smoothing (or exponential moving average) is defined by the following equation:

$$S_t = \alpha y_t + (1-\alpha)S_{t-1}$$

Where S_t is exponential smoothing factor and $\alpha \in (0,1)$ is the smoothing coefficient that controls the assigned weight to recent observations. As it can be seen, it only has one hyperparameter, and one running parameter in the model. It can be selected

with Grid Search or manually depending on if the user wants to prevail previous data or new ones.

The problem with this model is that it only works for one step forecasts, as S_t will stay constant, thus providing the same forecasts. Ultimately this means that these models are limited to short-term forecasting and with a slow reaction to changes in the regime.

3.1.3 ARIMA

The *ARIMA(p,d,q)* model proposed in (Box & Jenkins, 1976) is a univariate time-series model that combines three components:

- Autoregressive (AR – p): the series $y_t^{(d)}$ is modeleled following an AR process of order p, making it linearly dependent on its own past p values.
- Integrated (I – d): the original series is differentiated to achieve stationarity:

$$y_t^{(d)} = (1-L)^d y_t$$

- Moving Average (MA – q): the series $y_t^{(d)}$ is allowed to depend on past white-noise shocks of $\varepsilon_{t-j} \sim \mathcal{N}(0, \sigma^2)$ up to lag q.

The best way to identify a fitted ARIMA model is observing the ACF and PACF functions. The first one is used to identify the correlation peaks of a time series with itself at different lags, and the PACF is used for the same task, but after removing the effects of the previous lags. The implementation uses the `statsmodels` class of ARIMA (statsmodels, 2024) but is capable of automatically identifying the best fit within a set of parameters by brute force. The algorithm searches the combinations of p,q up to 4 and d up to 2, which are in reasonable limits. It uses AIC (Akaike's Information Criteria) as metric to evaluate the best combination of parameters. If k is the number of estimated parameters and $\hat{L}$ is the maximum value of the likelihood function, then:

$$AIC = 2k - 2 * ln(\hat{L})$$

This metric shows the accuracy and complexity of a model in a single statistic. We have preferred this over BIC (Bayesian Information Criterion) since the dataset we are working is not very large, and BIC is stricter on bigger datasets (Jani Data Diaries, 2024).

After the best model has been identified, $\{\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q, \sigma^2\}$ are estimated and we get the residuals r_t. Multi-step forecasts are later obtained recursively. For a horizon h:

$$f_{t+h}^{(d)} = \sum_{i=1}^{p} \phi_i f_{t+h-i}^{(d)} + \sum_{i=1}^{q} \theta_i r_{t+h-i}^{(d)}$$

ARIMA models are very popular as they offer a simple to fit, reliable and light-weight model. Additionally, it has had many developments since it was created, which helps to adjust to specific settings for improved performance. For instance, SARIMA, which captures seasonal patterns, or ARIMAX (Wong, 2024), that incorporates external predictor variables. However, even these other methods have shown improved performance, they also require a more exhaustive computational fitting process, which would be incompatible with our scalability purposes.

3.1.4 DYNAMIC LINEAR MODELS (DLM)

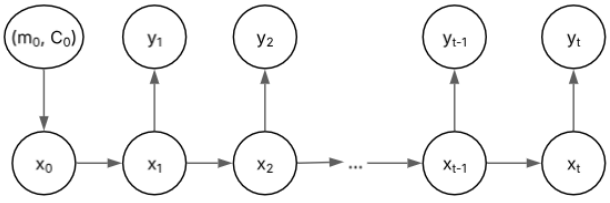

Figure 2 Dynamic Linear Model recursive diagram (Locruz, Lasala, & Lekuona, 2000)

Dynamic Linear Models (DLMs), formalized by (West & Harrison, 1997), provide a Bayesian framework for tie series analysis. An unobserved state vector evolves linearly over time generating noisy observations. In the text, they also show how the models are scalable to express as different modules with trend, seasonal or autoregressive

components. As seen in Figure 2, by recursively updating with each new data point, it is also able to track uncertainty through full posterior distributions. Because of this modularity, lightweight and probabilistically interpretable results, they have become important in modern forecasting and online learning.

In DLMs, we assume that at each time t there is a latent state vector $x_t \in \mathbb{R}^p$ summarizing the unobserved structure of our time series. This includes all our components: level, linear, seasonality or autoregressive. As mentioned, this state evolves linearly with noise and produces scalar observation $y_t \in \mathbb{R}$ with a linear projection.

The first step in implementing a DLM is defining $m_0 \in \mathbb{R}^p$ and $C_0 \in \mathbb{R}^{pxp}$, which are the initial guess for the latent components and the uncertainty we have with them. From our initial guesses, we will define a Gaussian prior.

$$x_0 \sim \mathcal{N}(m_0, C_0)$$

Typically, this is set with $m_0 = 0$ or an empirical guess, being $C_0 = kI$ with k large.

As mentioned, this state vector will drift linearly at every step following:

$$x_t = Fx_{t-1} + w_t, \qquad w_t \sim \mathcal{N}(0, V)$$

In which $F \in \mathbb{R}^{pxp}$ is the state transition matrix and $V \in \mathbb{R}^{pxp}$, that controls the variance of w_t. This is known as the state evolution phase.

Once we have calculated the latent state x_t, we will observe only a scalar, our prediction:

$$y_t = Gx_t + v_t, \qquad v_t \sim \mathcal{N}(O, W)$$

Where $G \in \mathbb{R}^{1xp}$ observation (measurement) matrix that maps the state to the observation space and $W \in \mathbb{R}$ observation noise variance.

However, one of the main benefits of Dynamic Linear Models is its update algorithm. To fuse prior knowledge with new data we use a Kalman Filter.

After assuming that $y_{1:t-1}$ is our posterior for x_{t-1} and Gaussian with mean m_{t-1} and covariance C_{t-1}, we can perform the one-step ahead prediction with

$$a_t = Fm_{t-1}, \qquad R_t = FC_{t-1}F^t + V$$

So that before seeing the true observation y_t, we believe that:

$$p(x_t|y_{1:t-1}) = \mathcal{N}(a_t, R_t)$$

From this distribution we can derive the forecast f_t and its uncertainty Q_t:

$$f_t = Ga_t, \qquad Q_t = GR_tG^t + W$$

Here we can observe the scalar error $e_t = y_t - f_t$ and the Kalman gain,

$$A_t = R_tG^tQ_t^{-1}$$

These values tell us how much to correct our state prediction after knowing the new data:

$$m_t = a_t + A_te_t, \quad C_t = R_t - A_tQ_tA_t^T$$

This way, $x_t|y_{1:t} \sim \mathcal{N}(m_t, C_t)$.

In case we want to forecast more than one step ahead (horizon h) without further observation, we just need to iterate the prediction equations.

$$a_{T+k} = Fm_{T+k-1} \qquad R_{T+k} = FC_{T+k-1}F^t + V$$

For $k = 1, \dots, h$. With this we can get our forecasts as:

$$f_{T+h} = Ga_{T+h} \qquad Q_{T+h} = GR_{\{T+h\}}G^t + W$$

But now, since we do not have true observation y_t to compare, we continue assuming $m_{T+k} = a_{T+k}$ and $C_{T+k} = R_{T+k}$ for each of the steps.

Additionally, DLMs allow us to have a distribution for each forecast over our prediction by doing:

$$\sigma_{T+h} = \sqrt{Q_{T+h}}$$

We have implemented this model in a custom class in python. However, it is true that we find hard to define the correct components and initial matrices to have a stable and accurate DLM. Therefore, we have developed an automatic function that fits the DLM with a given training data. The user can insert the components they want to add, including:

- **Level,** random walk with noise with dimension $p = 1$. It assumes the series can drift up and down, moving by a small Gaussian shock. The algorithm defines:

$$F_{lvl} = [1] \quad G_{lvl} = [1] \quad V_{lvl} = \sigma^2_{\Delta y}$$

- **Trend block**, implements a local linear trend with level l and slope b. It assumes that b_t can also move with small or fixed variance. It defines:

$$F_{tr} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad G_{tr} = [1 \quad 0] \quad V_{tr} = diag\left(\sigma^2_{\Delta^2 y}\right)$$

- **Seasonal blocks,** for each of the supplied periods. If the user does not supply any, the algorithm detects the highest appearing frequencies with FFT (Fast Fourier Transform) (Musbah, El-Hawary, & Aly, 2019). For each period p, the added components are:

$$F_{seas} = \begin{pmatrix} cos(2\pi/p) & sin(2\pi/p) \\ -sin(2\pi/p) & cos(2\pi/p) \end{pmatrix}$$

$$G_{seas} = [1 \quad 0] \quad V_{seas} = diag\left(\sigma^2_y\right)$$

- **Autoregressive block**, the users set an AR order k capturing the k last lags of the series. This creates a k-dimensional state. The algorithm fills F and G with zeroes and V with the diagonal of the variance.

Lastly, all V matrices are multiplied by a factor the user sets up manually. It is usually in the $[1e^{-2}, 1e^{-3}]$ range.

Then, the algorithm combines these matrices creating the different modules required for forecasting. The matrices are the block diagonal of the larger matrices F, G, V, and W.

Also, we can infer the size of these matrices, and number of parameters. If we assume that p is the number of seasonal components and k is the autoregressive terms:

$$s = 1 + 2 + 2 * p + k$$

$$Params = F + V + G + W = s^2 + s^2 + s + 1$$

We have chosen this model for several reasons. First, it enables online updating in a very lightweight manner. As has been seen, when a new datapoint arrives, we need to run one inverse matrix and several multiplications. This makes it extremely efficient for long series. Additionally, the model is modularized, so it is easy to add or discard components. And lastly, but most importantly, without overhead calculations you also get full Bayesian intervals.

3.2 DEEP LEARNING METHODS

3.2.1 MLP

An MLP is thought as a stack of fully connected layers forming a neural network, where each layer applies a linear transformation, a ReLU activation and optional dropout:

$$o^{(i)} = ReLU\left(W^{(i)} x^{(i-1)} + b^{(i)}\right)$$

The model includes a last linear layer (`fc_out`) that maps the last h features to a single scalar output.

The model accepts a fixed-size vector of length `input_size` that is fed into the network. The output is a scalar value as the forecast that should continue the initial vector. In case we wanted to perform multi-step predictions, we would update the initial vector with the last forecast. Ultimately, the main inconvenient of doing this with an MLP is that errors quickly accumulate, since the model has not been prepared to manage temporary dependencies in the input data. All values in the vector are i.i.d.

In addition to `input_size`, the model requires `hidden_sizes`, a list including the number of neurons per layer and the dropout probability. Lastly, the model requires the typical deep learning training pipeline with its optimizer, loss function, training and testing pipelines etc.

3.2.2 CONV1D

A 1D convolutional model relies on performing the convolution operation over a set of inputs. This is sliding a "1D kernel" of small dimensions across the input sequence. The output $j-th$ output feature at position t will be defined by:

$$\left(o^{(i)}\right)_t = ReLU\left(\sum_{r=1}^{k} w_r x_{t+r-1} + b\right)$$

Lastly, the output vector is flattened and passed through a linear layer to get a final forecast. Similarly to the MLP, the forecast updates the fixed size input vector to get the multi-step forecast.

The operation could include pooling or striding to down-sample each feature map to reduce length or enforce invariance. Apart from this decision, we also need to define the number of Conv1D layers to use, the input, output and kernel sizes, with the dropout probability.

3.2.3 LSTM

Long-Short Term Memory idea was introduced in (Hochreiter & Schmidhuber, 1997), although the model has widely been adapted during the 2010s into the current implementations. It is an adapted recurrent neural network (RNN) designed to capture short-term and long-term dependencies with a cell state. As it can be seen in Figure 3, the cell state is updated by three gates that input different information: forget (what information to discard from the previous), input (what new information to add) and the output gate (controls how much of the new cell state influences the next hidden state).

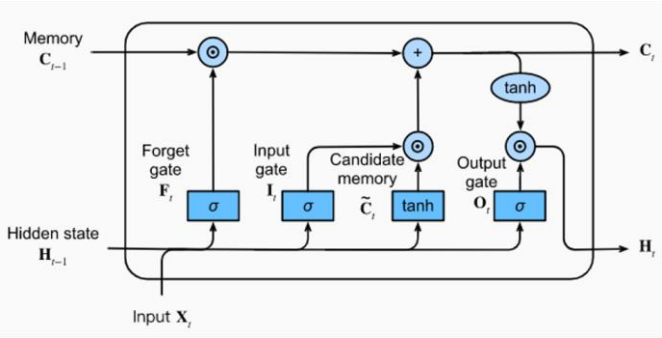

Figure 3 LSTM Architecture (Dive Into Deep Learning, 2022)

In the implementation we have made using `pytorch`, the user will need to define the input size from the original series (x_t), the hidden size of the memory cell c_t and the number of layers.

What is most attractive of this model is its ability to dynamically choose what information to keep at each step during long sequences, making it very efficient for time-series forecasting. The main downside is its complicated learning process with these dependencies between weights.

3.3 HIERARCHICAL RECONCILIATION TECHNIQUES

The second part of this research project is understanding the different reconciliation techniques and how they update the forecasts to make them more accurate. We begin with a set of base forecasts, $\hat{y}_t \in \mathbb{R}^m$, which we will try to adjust so that the reconciled forecasts are coherent (*i.e.* they respect the aggregation constraints) and they minimize overall forecast error across all levels.

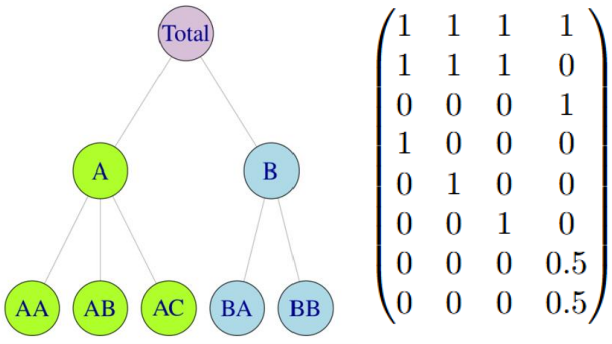

Figure 4 Example of summing matrix with Classic.

If we define m as the total number of series (nodes) and n the number of forecasts at the lowest level. The summing matrix $S \in \mathbb{R}^{mxn}$ encodes the aggregation structure. So, if we have a vector $b \in \mathbb{R}^n$, then we can get a vector of all series:

$$y_t = Sb_t$$

We will then define a matrix $G \in \mathbb{R}^{mxm}$ that transforms the full vector of base forecasts $\hat{y}_t$ into reconciled forecasts:

$$\tilde{y}_t = SP\hat{y}_t \quad G = SP$$

The second equation ensures coherence in the reconciliation. $P \in \mathbb{R}^{nxm}$ is a matrix that projects forecasts from all levels back to bottom nodes.

3.3.2 *CLASSIC RECONCILIATION*

Among hierarchical approaches, the two most widespread and simple techniques are Bottom-Up (BU) and Top-Down (TD), being Middle-Out a mix between them. While implementing them, we realised that it could be summarized in one function that automatically identifies which nodes needs to do bottom-up and which ones are top-down.

Our implementation requires the user to input which nodes are the base forecasts, so that it can reconcile up from that. This way, the bottom-up process will add all nodes with the same weight, and the top-down will divide equally so it adds up to 1. For instance, in figure 4, nodes "BA" and "BB" are inferred as top-down since "B" is the last (column) base forecast.

It must be noted there are other methods to estimate the weights for top-down, but in this case, we wanted to build a baseline model that does not require any other information and only the structure. Therefore, in cases where the aggregation is not a direct addition, we expect it to work badly.

3.3.3 *REGRESSION*

As mentioned, *Classic* method does not oversee well cases where the aggregation is not an equal distribution. We found some other documentation that followed historical averages and their proportions, or using the historic average proportion. However, these two methods continued to be unrealistic.

A solution we thought could be worth testing was fitting regression weights. Given that we have the historical series for all nodes, we could select the related ones and perform a fitted regression. This can be done when assigning weights from parents to children (Top-Down) or from children to parents (Bottom-Up). This way we capture more realistic relationships between nodes than just assuming equality. In this case we do require to have data from all series to fit the regression weights, unlike *Classic* reconciliation.

Also, a reinterpretation of what S matrix means (*i.e.* how base forecasts aggregate), would allow us to assess with this method as S, given it is the most accurate representation of how nodes relate to each other. We will evaluate this in Section 4.2.

3.3.4 *MINT RECONCILIATION*

Minimum Trace (MinT) optimal reconciliation was initially introduced in (Wickramasuriya, Athanasopoulos, & Hyndman, 2018). The objective is to reconcile the base forecasts of all nodes in the hierarchy by finding a linear adjustment that ensures coherence with the aggregation constraints and minimizes the total variance of the forecast's errors.

The general MinT solution requires to compute the residuals of the base forecasts, yielding an m-sized vector such that:

$$e_{t+h} = y_{t+h} - \hat{y}_{t+h}$$

If we define $W_h = Var(e_{t+h})$ the covariance matrix of the residuals from all nodes, then our optimal reconciliation matrix $G = \in \mathbb{R}^{mxm}$ must solve:

$$\min_G trace(Var(e_{t+h})) \quad s.t\ GS = S$$

In the paper (Wickramasuriya, Athanasopoulos, & Hyndman, 2018), it is demonstrated that if we enforce unbiasedness between the nodes and the coherent constraint, we can define matrix G as:

$$G_{MinT} = (S^t W^{-1} S)^{-1} S^t W^{-1}$$

And that would give us the full reconciliation process to end with:

$$\tilde{y}_{t+h} = S G_{MinT} \hat{y}_{t+h}$$

However, in practice, matrix W_h is unknown and must be estimated from historical residuals. However, we must have in mind that this matrix can be big and ill-conditioned. This means that it will be very sensible to its solutions and will make hard to get its inverse given its closeness to singularity.

The paper (Wickramasuriya, Athanasopoulos, & Hyndman, 2018) also proposes several methods to estimate it.

MinT OLS

Ordinary Least Squares (*MinT OLS*) method has no prior information about the residuals of the series. It assumes that all series have an equal, uncorrelated, residual variance:

$$\widehat{W}_h^{OLS} = k_h I_m$$

Here, k_h is a scalar multiplier (*e.g.* the average in-sample forecast error variance at horizon h) that we have assumed for all experiments as 1. I_m is the m-dimensional identity matrix.

The main benefit of this method is that it is always well-conditioned. Therefore, it is very easy to compute and will not trigger inverse errors.

WLS$_S$

Structural Diagonal (WLS_S) method does not require prior information on the residuals; only how base nodes aggregate. It assumes that the bottom-level series share a common variance, but the aggregated series variance scales with the number of bottom components:

$$\widehat{W}_h^{WLS_S} = k_h \Lambda \quad with \quad \Lambda = diag(S1_n)$$

In this equation, 1_n is a vector n-dimensional vector of bottom level forecasts., so that Δ counts how many bottom series feeds into each node. Meanwhile it is still diagonal and invertible.

WLS$_V$

Weighted Least Squares Variances is the simplest method to include historical residuals. It assumes that only the diagonal variances matter, by allowing each node to have its own variance.

$$W_h^{WLS_V} = k_h diag(\widehat{W}_h)$$

$$\widehat{W}_h = \frac{1}{T} \sum_{i=1}^{T} \hat{e}_{t+h} \hat{e}_{t+h}^t$$

One of the main benefits of this method is that instead of estimating a full covariance matrix $W_h \in \mathbb{R}^{mxm}$, we would only need its diagonal, m variances. This method continues to be diagonal and invertible, and in most cases, still well-conditioned.

MinT Sample

This method follows MinT reconciliation without assuming anything, as residuals are informative enough to estimate W_h:

$$\widehat{W}_h^{Sample} = \frac{1}{T} \sum_{i=1}^{T} (\hat{e}_{t+h} - \bar{e}_h)(\hat{e}_{t+h} - \bar{e}_h)^t$$

The main benefit of this method is that is can be as informative as possible with the information we have from all series. However, it requires to compute an mxm matrix that easily tends to be ill-conditioned or singular when m is large relative to T (residuals length).

Therefore, we have implemented a small algorithm to solve this issue, as we encountered it on several

occasion. Once the $W_h(0)$ has been estimated, calculated its condition metric, and if it is larger than $1e5$, we have performed Tikhonov regularization. It searches within a Grid Search for the most appropriate α value so that:

$$W_h(\alpha) = W_h(0) + \alpha \frac{trace(W_h(0))}{m}$$

This is now better conditioned that the original one.

3.3.5 *MINT SHRINK*

This method is a variant of *MinT Sample,* which aims to stabilize the sample-covariance estimator by shrinking it towards a simpler, well-conditioned target.

$$\widehat{W}_h^{Shrink} = \lambda \widehat{D} + (1-\lambda)\widehat{W}_h^{sample}$$

In this case, $\widehat{D} = diag(\widehat{W}_h^{sample})$, is the structured matrix containing the variances of the diagonal. The parameter λ is the shrinkage intensity which minimizes the mean-squared error of the estimator.

Compared to *MinT Sample*, this method improves the condition of the W_h matrix noticeably, although it highly depends on the parameter lambda. If it is too close to 1, then W_h^{shrink} will be closer to $W_h^{WLS_v}$ than to W_h^{sample}. Therefore, we have implemented two methods to estimate lambda.

First, we have approximated the variances of the correlations of each residual by:

$$var_{r_{ij}} = \frac{\left(1 - r_{ij}^2\right)^2}{T-1}$$

Then, the best λ has been estimated by summing these variances and comparing it to the total correlations:

$$\lambda = \frac{\sum var_{r_{ij}}}{\sum r_{ij}}$$

This is clipped between [0,1]. Also, we have performed the same algorithm as in *MinT* Sample, to improve the matrix's condition using Tikhonov's regularization.

The second method will estimate λ with the LeoditWolf estimator (Leodit & Wolf, 2004). This method follows the formulas in *MinT Shrink* but estimates λ automatically in the `sklearn.covariance.LeoditWolf`. We have named *MinT Shrink LW* for experiments and analysis. We also implement the Tikhonov regularization.

MinT Shrink achieved some of the best results in (Wickramasuriya, Athanasopoulos, & Hyndman, 2018), so we expect a superior performance.

3.4 *PROBABILISTIC RECONCILIATION*

However, these methods only provide "point" reconciliation, and we require additional steps to provide a distribution over them. We have selected two simple methods, each of them following the main probabilistic "schools". The first one, "closed form" and "variational" will be based on a Gaussian assumption, and the second one is based on Monte Carlo sampling.

The first attempts to provide with full distributions were introduced in (Taieb, Taylor, & Hyndman, 2017), but later works from the previously mentioned (Panagiotelis, Gamakumara, Athanasopoulos, & Hyndman, 2023) build up on that approach. They provide alternative methods to estimate W or choose shrinkage targets that work better for probabilistic forecasts.

3.4.2 *GAUSSIAN*

In this method, it is assumed that the unreconciled base forecasts follow a multivariate normal distribution:

$$\hat{y}\_t \sim \mathcal{N}(\mu\_t, diag(\sigma_{1,t}^2, \dots . \sigma_{m,t}^2)$$

If we take G, the MinT reconciliation matrix estimated from any of the previous methods, and W

the assumed error-covariance, following the Gaussian assumption.

$$\mathbb{E}[\tilde{y}_t] = G\mu_t, \qquad Var(\tilde{y}_t) = SGW(SG)^t$$

In practice, we will extract marginal standard deviations (the squared root diagonal of the covariance matrix) as coherent uncertainty measures for each set of predictions:

$$\sigma(\tilde{y}_{i,t}) = \sqrt{[SGW(SG)^t]_{ii}}$$

This method is very efficient computationally since it does not require to calculate further matrices, and uses already defined W and G. However, a Gaussian assumption on the base forecasts is strong and might not always hold.

3.4.3 MONTE CARLO

This second alternative is based on Monte Carlo sampling, where we will draw samples based on each forecast's distribution and then apply reconciliation transform to each draw.

The first step is to draw $l = 1 \dots K$ samples from:

$$\hat{y}_t^{(l)} \sim \mathcal{N}\left(\mu_t, diag(\sigma_{1,t}^2, \dots., \sigma_{m,t}^2)\right)$$

For each of the unreconciled samples, we will perform:

$$\tilde{y}_t^{(l)} = SG\hat{y}_t^{(l)}$$

Lastly, we compute the sample mean and covariance:

$$\widehat{\mu_t^{(MC)}} = \frac{1}{K}\sum_{l=1}^{K} G\ \widehat{y_t^{(l)}}$$

$$\widehat{\Sigma_t^{(MC)}} = \frac{1}{K-1}\sum_{l=1}^{K}\left(G\,\widehat{y_t^{(l)}} - \widehat{\mu_t^{(MC)}}\right)\left(G\,\widehat{y_t^{(l)}} - \widehat{\mu_t^{(MC)}}\right)^t$$

Then we will take the diagonal entries from $\widehat{\Sigma_t^{(MC)}}$ as the coherent variance estimates per series and do the square root to get standard deviations.

The main drawback for this approach is its computational cost. Its dependence on the number of samples takes, K, means we need to adjust it conveniently to correctly manage our resources. Unlike the analytical or fixed-formed Gaussian method, Monte Carlo reconciles from multiple draws from the assumed distribution, yielding more realistic forecasts and uncertainty.

3.5 EVALUATION METHODS

In this Section, a brief description of the evaluation methods used for the experiments. We will name y_i the true targets, $\hat{y}_i$ the model's point prediction and $\hat{\sigma}_i$ the standard deviation for $\hat{y}_i$. N is the total number of samples.

3.5.2 RMSE

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

RMSE is an error metric that penalizes larger errors quadratically and then rescales the values to the original units. It is widely used when large deviations are especially undesirable.

3.5.3 MAE

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

This error metric averages the deviation between true and predicted values. It is preferred in cases where a linear error penalty is more appropriate.

3.5.4 EXPECTED CALIBRATION ERROR

The Expected Calibration Error (ECE) is a metric that shows how well a probabilistic model's predicted confidences match its empirical accuracy.

The first step is to take M equally spread coverage levels from the [0,1] interval. The, we will compute

the empirical coverage for each of those levels following:

$$cov(\alpha_j) = \frac{1}{N}\sum_{i=1}^{N} 1\{y_i \in [\hat{y}_i \pm z_{1-\alpha_j}\hat{\sigma}_i]\}$$

The, we will calculate the ECE by:

$$ECE = \frac{1}{M}\sum_{i=1}^{M}\left|cov(\alpha_j) - (1-\alpha_j)\right|$$

In cases where the $ECE = 0$, then the model's predicted probability matches its actual correctness rate, so it is perfectly calibrated. A larger value of ECE tells us that on average the model 's confidence is misaligned with reality. The main inconvenience with this method is defining the correct M, number of α levels. We have used 10.

3.5.5 NLL

Negative log-likelihood (NLL) measures how well the model assigns high probability to the observed labels. Therefore, we are required to express our mean and variance predictions into a probabilistic distribution. We will do that by assuming it is Gaussian:

$$p_\theta(y_i \mid x_i) = \mathcal{N}(y_i; \hat{y}_i, \sigma_i^2)$$

Then, the NLL becomes:

$$NLL = -\frac{1}{N}\sum_{i=1}^{N} log[\mathcal{N}(y_i; \hat{y}_i, \sigma_i^2)]$$

And if we substitute the Gaussian distribution formula inside the NLL, then it becomes:

$$\text{NLL} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{2}\log(2\pi\,\sigma_i^2) + \frac{(y_i - \hat{y}_i)^2}{2\,\sigma_i^2}\right]$$

Therefore, the lower NLL value indicates a better characterization of the distribution compared to the actual data. It is one of the proper scoring rules mentioned in (Gneiting & Raftery, 2007), as it effectively measures accuracy of the mean forecast and the quality of the uncertainty estimates.

3.5.6 CRPS

The last error metric we wanted to compute is the Continuous Ranked Probability Score, as it is included in one of the proper scoring rules in (Gneiting & Raftery, 2007). In that case, CRPS of a predictive CDF (Cumulative Distribution Function) F and observed value y_i is:

$$\text{CRPS}(F, y_i) = \int_{-\infty}^{\infty}(F(x) - 1\{x \geq y_i\})^2\, dx$$

The intuition behind the metric is that it measures how much mass F places around true y and the squared difference will be large in some regions. So, a lower CRPS value indicates a better probabilistic forecast. Also, it is important to note that CRPS has the same unit as variable y.

However, our models output a mean and standard deviation, as an adapted form assuming Gaussian in terms of them is:

$$\text{CRPS}(\mathcal{N}(\mu, \sigma^2), y) = \sigma\left[t(2\Phi(t) - 1) + 2\phi(t) - \frac{1}{\sqrt{\pi}}\right]$$

Where: $t = \frac{y-\mu}{\sigma}$, a normal distribution with mean 0 and variance 1. The cdf (Cumulative Distribution Function, $\phi(t)$) is giving the probability that standard normal variable is $\leq t$ and the pdf (Probability Density Function, $\Phi(t)$) is giving the relative likelihood at t.

4 EXPERIMENTS AND RESULTS

As mentioned, we will test all these methods with two different datasets: first, in Section 4.1 we will use synthetic data where the distributions of the random generators and functions are known. This way we will be able to understand the potential and pitfalls of each technique in a controlled environment. Secondly, in Section 4.2 we will use the S&P500 Index, having as base forecasts the

stocks and market-capitalization weighting. This will help up see how reconciliation techniques perform with real data.

For each dataset, we will give a brief description of the structure, along with the analysis of the experiments with the forecasting models and the reconciliation methods.

4.1 SYNTHETIC SERIES

The first step to understanding how the different models work is assessing them in a controlled environment where randomness and trends are known. This will allow us to learn how the model behaves under different conditions, providing a better characterization of their limitations. Initially, we will briefly describe the dataset, Then, we will continue with the forecasting experiment and, based on it, select a model to perform the hierarchical reconciliation analysis.

Dataset Structure

To evaluate the performance of the model, both for forecasting and reconciliation, we have constructed a hierarchical structure as shown in Figure 5:

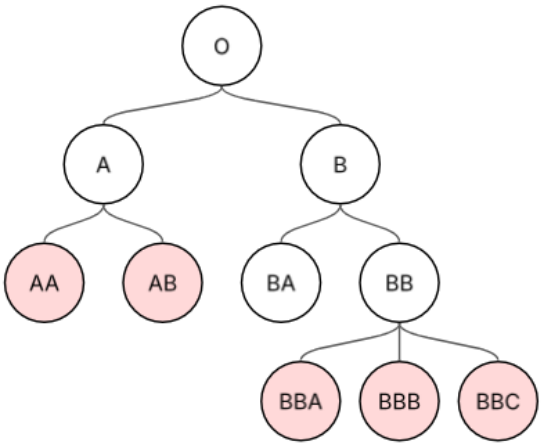

Figure 5 Synthetic Series Hierarchical Structure

The selected functions plus Gaussian random noise $(\mu = 0, \sigma = 5)$ for each of the leaf nodes are:

- $AA = 5 * log(t)$
- $AB = 50 * np.sin(t/5)$
- $BA = 0.01 * t - 50$
- $BBA = -0.005 * t + 33$
- $BBB = 50 * cos(t/3)$
- $BBC = 50 * e^{(-t/500)}$

To create the upper nodes, we have added them in the structure following the relationships. We have chosen this architecture given its variability in the type of functions but also the simplicity of each part separately.

Additionally, we find that there could be some interest in assigning different weights while creating the parent nodes. However, since we want to structure the project as an ablation experiment with rising complexity, we have decided to leave that experiment towards the S&P500 forecasting, in Section 4.2.

To perform this experiment, we generated 10k samples from these functions and then propagated them upwards in the hierarchy. Experiments have been conducted by preserving the last 20% of the data for testing. We scaled the data for the Deep Learning models (Section 3.3), *DLMs* and *ARIMA* after seeing better performance.

Forecasting Experiment

In this case we will compare the performance on the different forecasting models on the top series 'O' from the hierarchy. Since it is formed by all the child's series, it will be composed with a wide variety of functions and their additive Gaussian random noise.

The experiments have been in done in file `model_runner.py`. The user needs to input a numpy array with the time series data, along with the training percentage, the largest horizon to forecast and the window size for *SMA, MLP, LSTM* and *Conv1D*. Then, the data is scaled using numpy's `StandardScaler` and processed through each model. Given the architecture of Deep Learning methods, we need to create the sequences of `window_size` arrays that will output our forecast. In our case we

have used a `window_size` of 15 and `max_horizon` of 30.

In every model, we have conducted one time-step forecasts and used that value to update it. In some cases (*EMA, ARIMA, DLM*) to change parameters and in others (*LSTM, MLP, Conv1D, SMA*) to append at the end of the input array. We save the model state before predicting the full horizon and update the initially fitted state of the model.

Lastly, the model computes the proper scoring metrics described in Section 3.5 and saves the forecasts. The plots shown here are directly provided by the code in `plotter.py`, available in the code repository.

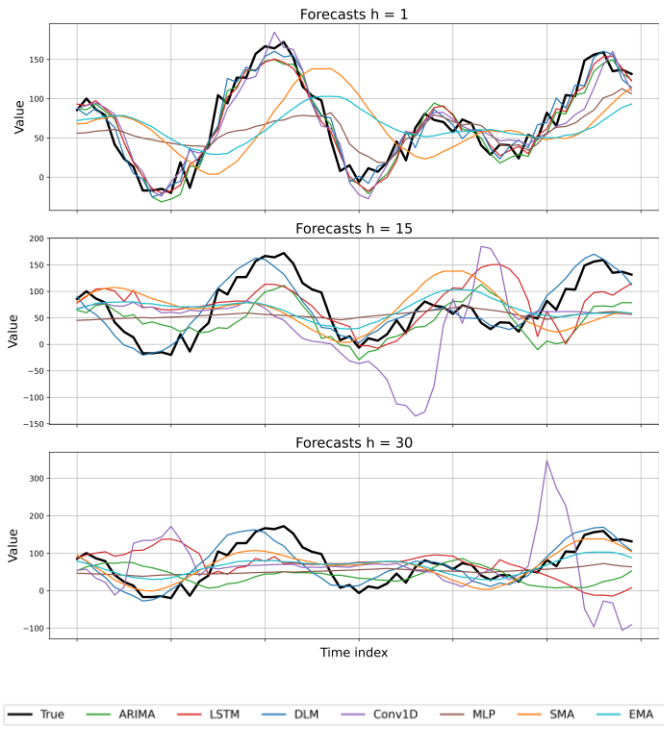

Figure 6 Point Forecast for h = [1, 15, 30] on "O"

Figure 6 shows a sample of the forecasts. We have plotted the h (1,15,30) forecast for those specific indexes. The black lines represent the real data, which is kept the same for all three figures although it is represented in different scales (horizontal scale). This is because forecasts get worse as horizon increases, as we will see in posterior plots.

The first plot, with $h = 1$, can distinguish three methods: *EMA, SMA* and *MLP*, that show significantly worse performance when compared to the other methods. They tend to stay centred in the series and not adapting to the needed level of flexibility shown by the data. If we go down to higher horizons, it seems like they displace their peaks in time. In fact, *EMA* does just that and *SMA* does it with the trend of the first forecast point. The behaviour of the *MLP* looks simpler as it tends to shrink the values towards the mean of the series when updating the input vector with the forecasts. This is because it does not have temporal awareness as the input vector values are considered identically and independently distributed (i.i.d.).

Secondly, we have *Conv1D* and *LSTM*, where they initially tend to follow the series very well. However, as the horizon gets bigger, *Conv1D* suffers from an overfitting issue. As we can see in $h = 15$ in the middle and $h = 30$, towards the end, the models over prioritize the latest information over the old one. This is shown by the high peaks and valleys. However, this is fixed in the cell structure in the *LSTM*, where the model rarely deviates in short-term horizons. When it does, it adjusts back fast and then continues normally (end of $h = 15$), with the curve in red.

Lastly, from this visual comparison, we can see that the best fitting models have been *ARIMA* and *DLMs*. *ARIMA* fits the series every well, although as the horizon increases, it loses track of the series. Like *LSTM*, the error it makes usually stays close to the true series, and when it misses, it gets fixed easily. The best performing model (Figure 7) for this series is the *DLM* since the seasonal and linear trends of the series are easily replicated by the model. The small displacement in the higher horizons comes from model predicting earlier the peaks in the base series, probably due to a small shift in the seasonality of the series.

Further analysis could be performed if we analyse the RMSE of each model compared to the horizon. We have shown RMSE since we want to analyse point forecasts first, and then we will see how the

predictive intervals behave. Therefore, probabilistic evaluation metrics will be analysed after.

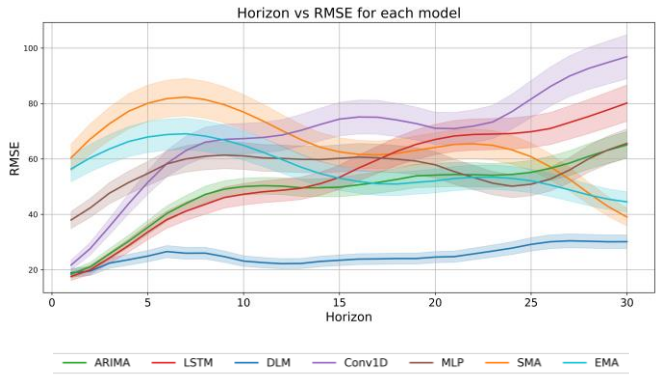

Figure 7 RMSE (+- standard error) against horizon

The first thing we can see from Figure 7 is how well the *DLM* adjusts to seasonality, and we can see the effect of the mentioned displacement towards higher horizons. Another very good proof of its performance is how thin and constant standard error (interval around each value) is. This means that we do not have many outliers in terms of residuals and are all quite similar.

Another good comparison can be seen between *ARIMA* and *LSTM*. The standard error is much thinner in the *ARIMA*, and more constant, meaning that when there is a change in regime, predictions are affected less, and the model recovers faster. Also, they both tend to work good in the short term, although the *LSTM* is not as used due to scalability concerns. The contrary can be seen with SMA and EMA, that as we mentioned in Section 3.1, they are very resilient to change as new data come in.

The fact that their RMSE is going down with horizons shows the models doing underfitting, like *SMA* and *EMA*, are benefited over those doing overfitting. These models are essentially Deep Learning ones and would require less training, more data or a bigger structure to be fixed. A clear example is the behaviour of *Conv1D,* which has increasing error with increasing standard error too when doing long-term forecasts.

Metric	Model	Horizon 1–6	Horizon 13–18	Horizon 25–30
CRPS	ARIMA	16.37 ± 0.40	29.74 ± 0.64	34.27 ± 0.90
	DLM	$\mathbf{13.55 \pm 0.35}$	$\mathbf{13.35 \pm 0.28}$	$\mathbf{17.37 \pm 0.37}$
NLL	ARIMA	$\mathbf{4.75 \pm 0.03}$	5.39 ± 0.03	5.57 ± 0.04
	DLM	4.99 ± 0.07	$\mathbf{4.58 \pm 0.02}$	$\mathbf{4.83 \pm 0.02}$
ECE	ARIMA	$\mathbf{0.05 \pm 0.00}$	0.10 ± 0.01	0.06 ± 0.01
	DLM	0.14 ± 0.01	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.05 \pm 0.01}$

Table 1 Probabilistic metrics for short, medium and long-term forecasting on "O"

In this Table we can see the probabilistic metrics computed for *DLM* and *ARIMA*, averaged around short, medium and long-term horizons. Bold values represent the best score for each metric and horizon. It also applies to following tables. Other models are not shown because they do not predict a distribution.

Results show that DLM tends to work better on longer term horizons, while ARIMA on shorter. That could align with what we saw on Figure 7. The fact that ARIMA is better in terms of NLL and ECE in the short horizon is surprising. Essentially, what this means is that ARIMA probabilistic interval adjusts better to the actual data distribution, but when doing multiple step forecasting, the error in the forecast is bigger and the standard deviations too. This can be confirmed with Figure A1 in the appendix. The fact that CRPS is consistently lower for this model is because it also accounts for the mean value, which has shown to be better for the DLM. In fact, this metric in medium and long-term horizon is half the value for the DLM showing that in terms of sharpness and calibration, the distribution is much better adjusted. This positions the DLM as dominant in series strongly governed by seasonality, by having less error in terms of point forecast and distribution fitting.

Lastly, we will explore the fact that we recover a decreasing behaviour in the ECE metric. This shows that as we increase in horizons, the predictive intervals grow in the correct direction. They resemble better the data and show better calibration across several values of α in Section 3. 4..

Model	# parameters	init time (s)	forecast time (s)	update time (s)
ARIMA	7	8.19×10^{1}	4.86×10^{-3}	8.21×10^{-2}
LSTM	14113	1.64×10^{2}	2.40×10^{-2}	4.71×10^{-3}
DLM	254	3.94×10^{-1}	1.92×10^{-3}	$\mathbf{1.11 \times 10^{-4}}$
Conv1D	8801	1.68×10^{2}	1.61×10^{-2}	1.88×10^{-2}
MLP	4577	7.66×10^{0}	1.04×10^{-2}	1.98×10^{-2}
SMA	1	0	$\mathbf{1.31 \times 10^{-4}}$	0
EMA	1	$\mathbf{2.00 \times 10^{-3}}$	1.68×10^{-4}	0

Table 2 # Parameters and execution times for different models

As mentioned in Section 1, one of the most important characteristics we wanted to confirm in the analysis is the model complexity. This is shown in Table 2 as the average time taken for each task. Our baselines, *SMA* and *EMA*, but with a noticeable decrease in accuracy However, what surprised us the most is the difference between *DLM*, *ARIMA* and then the rest of Deep Learning models, with *LSTM* being the worst.

Even though *DLMs* have a complex initialization algorithm with FFTs involved, it is surprisingly faster than the rest. It does not need to adjust parameters like Deep Learning models, which takes much longer having many more. Also, the fact that *ARIMA* is worse is because the algorithm is not fully optimized. By checking every combination of *(p, d, q)* parameters by brute force it adds a lot of complexity.

Most models lie between similar orders of forecasting time (*DLM* one order faster), but the significant difference is made in the updating times. This is measured every time the model introduces the new data point information. This makes sense as the *DLM* only need to perform a few matrix operations, while Deep Learning models need to forecast and backpropagate. *ARIMA,* which we expected to work better, updates the internal state of the model to adjust the "starting point" of the prediction.

As a conclusion, we have seen that simple models like *EMA, SMA* and, more complex, *MLP*, do not adapt to changes by lagging or shrinking forecasts towards the mean due to a lack of temporal awareness. *Conv1D* and *LSTM* seem to manage short-term forecasts, but Conv1D overfits as the horizon grows. LSTM mitigates well this with the cell structure by maintaining stability between past and added information. *ARIMA* has good performance for some of the proper scoring probability-related metrics in the short term. However, *DLMs* achieve similar performance in short-term predictions in those same metrics, while also providing more accurate and consistent point-wise predictions for the time series, as highlighted in Figure 7.

Reconciliation Experiment

To evaluate the reconciliation techniques, we have designed a script to run them, in charge of pre-processing the data, forecasting and saving results. It reads the data for each of the series and its hierarchy structure (a python dictionary). Then, it relies on model_runner.py for each of the base forecasts with a DLM automatically fitted and mixes all information in several dictionaries. Then, the residuals are computed to calculate the reconciliation matrices. Lastly, they are used to perform a forward pass on point Forecasting, *Gaussian* and *Monte Carlo* methods. The proper scoring evaluation metrics are calculated and all results saved in different files. All this can be found in `reconciliation_experiment.py`.

The goal of this experiment is to understand how these methods work compared to base forecast, which is running the *DLM* on every series.

Prob. Method	Rec. Method	1–6	13–18	25–30
base	base	**11.50**	**11.80**	**13.70**
Gaussian	Classic	8.85	10.05	11.90
	Regression	8.55	10.15	12.10
	MinT OLS	16.00	14.80	13.00
	MinT Sample	**1.00**	**1.10**	**1.60**
	MinT Shrink	3.50	4.30	4.00
	MinT Shrink LW	5.20	2.70	3.60
	WLS_V	12.30	10.40	10.70
	WLS_S	13.90	12.50	11.30
Monte Carlo	Classic	8.60	10.60	11.20
	Regression	9.30	11.90	11.10
	MinT OLS	16.40	15.40	13.60
	MinT Sample	**2.10**	**2.90**	**2.10**
	MinT Shrink	4.20	5.60	4.10
	MinT Shrink LW	5.00	4.40	5.60
	WLS_V	12.70	11.30	10.70
	WLS_S	13.90	13.10	11.70

Table 3 RMSE ranking for probabilistic reconciliation variants across short, medium and long-term horizons.

Table 3 shows us the average ranking results for RMSE. These rankings have been computed by taking the average RMSE error in each node, rank the different models and average the results. The best result in bold and the second best underlined in each of the probabilistic methods. Also, bold has been used to highlight best result, and underlining for second-best. The averaged RMSE values are in Appendix A.

The first thing that is clear is that *MinT Sample* has performed the best across both, ranking between 1st and 3rd most of the time. This makes sense as we have a simple hierarchy, and the matrix is relatively small. The inverse operations can be run without numerical issues. Both *MinT Shrink* methods have performed second best, being a bit surprising that the normal version sometimes is ranked better than the *MinT Shrink LW* method.

Also, the methods that do not depend on residuals, such as *Classic*, *Regression, WLS_S* have shown the lowest performances, even some below the base forecast. This is because they do not incorporate past information from the models and the relationships between nodes are purely deterministic on the structure. Therefore, error is not reduced, but accumulated across the hierarchy and the base forecast performs better.

With Table 3, we can see that in simple structures, with simple addition in the series reconciliation has enhanced forecasts in every method except *MinT OLS, WLS_S and WLS_V*. Our hypothesis was the opposite, but we can understand that *Monte Carlo* works a bit worse than *Gaussian*. Since the reconciliation matrices have improved the forecasts, a closed-form solution would also tend to improve them rather than a more spread average result.

However, the results from the NLL are prove differently, as can be seen in Table 4

Prob. Method	Rec. Method	1–6	13–18	25–30
Base	base	4.614 ± 0.061	$\mathbf{3.756 \pm 0.015}$	3.933 ± 0.011
Gaussian	Classic	4.464 ± 0.051	3.855 ± 0.012	4.013 ± 0.008
	MinT OLS	6.390 ± 0.129	4.018 ± 0.035	4.434 ± 0.034
	MinT Sample	4.377 ± 0.007	4.769 ± 0.002	5.080 ± 0.002
	MinT Shrink	4.204 ± 0.009	4.252 ± 0.003	4.959 ± 0.002
	MinT Shrink LW	$\mathbf{4.150 \pm 0.009}$	4.393 ± 0.003	5.020 ± 0.002
	Regression	4.464 ± 0.051	3.855 ± 0.012	4.013 ± 0.008
	WLS_S	6.686 ± 0.141	4.011 ± 0.036	4.357 ± 0.034
	WLS_V	6.483 ± 0.134	4.004 ± 0.035	4.320 ± 0.031
Monte Carlo	Classic	8.122 ± 0.195	5.848 ± 0.110	6.785 ± 0.117
	MinT OLS	16.659 ± 0.468	10.441 ± 0.246	15.169 ± 0.322
	MinT Sample	4.412 ± 0.059	4.088 ± 0.040	$\mathbf{3.906 \pm 0.030}$
	MinT Shrink	4.599 ± 0.068	4.127 ± 0.050	3.947 ± 0.033
	MinT Shrink LW	4.667 ± 0.071	4.070 ± 0.045	3.917 ± 0.031
	Regression	8.128 ± 0.196	5.845 ± 0.110	6.783 ± 0.117
	WLS_S	16.548 ± 0.466	10.213 ± 0.244	14.479 ± 0.310
	WLS_V	15.822 ± 0.444	9.998 ± 0.237	14.040 ± 0.297

Table 4 NLL for probabilistic reconciliation across different forecast horizons

We have chosen to show NLL here as it assesses how well the probabilistic forecast fits the actual data distribution, since we want to compare *Gaussian* and *Monte Carlo*. Other metrics like *CRPS* and *ECE* are available in Appendix A.

In shorter horizons, *Gaussian* seems to work better than *Monte Carlo* on average. As the forecasting horizon grows, the Gaussian assumption gains strength (as is to be expected from a Bayesian-prior point of view) and it induces worse results in terms of NLL. However, if we compare Monte Carlo with the unreconciled forecast, we can see that there is not much benefit over the longer horizon.

From all the methods what can be inferred is that the better the reconciliation matrix is over short-term prediction, the better performance it has. However, on longer horizons, the results in *Gaussian* seem to be better on matrices that come from a deterministic method such as *Classic, Regression* and *WLS_S*. In

the case of *Monte Carlo*, they continue to work in similar orders, probably given that this method is more realistic as it is based on sampling over the actual predictions. If the matrix is worse, then the reconciled samples will continue to be worse.

To further understand what reconciliation means in terms of change in forecast residual structure, we can calculate the correlation matrices from the residuals of the predictions. Figure 8 measures difference between them when doing reconciliation with *MinT Sample, Gaussian* in a short forecast of $h = 5$:

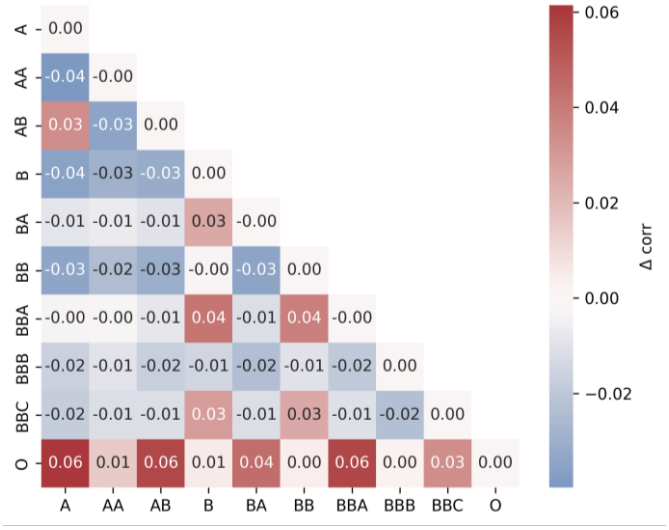

Figure 8 Difference in correlation between base and reconciled forecasts, in terms of correlation between nodes' residuals.

This matrix shows the difference of correlations between the residuals of all nodes in the hierarchy, between base forecasts and the case just mentioned. The idea of this plot is that if series are more uncorrelated, then more information has been extracted from base forecasts and supposedly improved them. What we can see here is that most of the cases correlation has been decreased, especially within the highest levels of the hierarchy.

In conclusion, to achieve the best results through reconciliation, we need to carefully construct the hierarchical structure to better reflect the inner dependencies among time series. We should consider qualitative analysis on the nature of the series, as we could have some levels that are more reliable than others. In this case, with complete information, we have seen that could make sense to use reconciliation methods in short term forecasts as correlation between residuals has decreased and accuracy improved. In a simple hierarchical structure as this one, we have seen that *MinT Sample* has worked the best in terms of point forecast, although *shrinked* methods outperformed in short-term forecasts when seeing it as a probabilistic distribution. We must have in mind that *MinT Sample* is not always the best method in bigger structures than this one. However, in longer term horizons, there was not much difference with base forecasts in terms of NLL, although it continued to rank better in point forecasting (Table 3). And lastly, *Gaussian* seems to work better in short-term and *Monte Carlo* on longer horizons, as the samples widen with horizon due to sampling.

4.2 S&P500 INDEX FORECASTING

To prove its applicability to a real-world scenario, we have decided to use the S&P500 Index. This is a market-capitalisation-weighted financial instrument meant to track the 500 leading publicly traded US companies. It can be subdivided in 11 sectors and those into 74 industries. Therefore, the downwards order in our hierarchy will be:

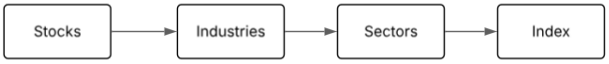

Figure 9 Hierarchical Structure for S&P500

There are several important points related to the data processing that should be mentioned. yfinance API (Aroussi, n.d.) does not provide actual information at industry or sector level, so we have calculated those values in terms of the weighted capitalization of the relevant stocks. The market-cap weighted aggregation formula used is:

$$C_G(d) = \sum_{i=1}^{n} \left(\frac{M_i}{\sum_{j=1}^{n} M_j} \right) C_i(d)$$

Here, M_j is the market capitalization (value of the company in the market) and C_j(d) is the price of the stock on day d. This is the method used to propagate the data throughout the hierarchy, but leaving the index as it was downloaded from the API from the "^GSPC" ticker. Companies which left the index during the last 2 years have been removed for improved consistency within the dataset.

Forecasting Comparison

As in the previous Section, we have started by analysing the time series with the different models before we delve into reconciliation techniques. In this case, to test the performance of the models in real data. Because we want to see the actual average error being committed, we have chosen to show the MAE for this plot, although other values are shown in the appendix.

Model	1–6	13–18	25–30
DLM	109.105 ± 3.611	265.225 ± 7.163	349.332 ± 8.891
SMA	123.758 ± 3.830	236.430 ± 5.675	265.049 ± 8.697
ARIMA	$\mathbf{98.607 \pm 3.255}$	$\mathbf{215.326 \pm 5.718}$	277.411 ± 8.352
LSTM	170.589 ± 3.842	280.635 ± 6.425	272.764 ± 8.643
Conv1D	130.158 ± 3.405	249.092 ± 6.459	336.577 ± 8.397
MLP	194.873 ± 7.116	308.892 ± 9.118	363.408 ± 10.014
EMA	151.096 ± 4.044	239.054 ± 5.957	$\mathbf{260.614 \pm 8.216}$

Table 5 Forecasting MAE for different models across increasing horizons.

The results show a good performance with the *ARIMA*, showing its optimality for near-stationary financial series, as it's the case for the S&P500. It performs well on shorter horizons due to its momentum, although in high h values, it degrades since it cannot extrapolate non-linearities.

The major difference with the synthetic data example is in the *DLM*. Although it performs well in short term, it soon loses accuracy as it's a very reactive model that has a hard time modelling noisy data like financial series.

On the other hand, simple models like *EMA* and *SMA* show that they are better in longer horizons as many models, since predicting averages tends to perform better than reactive models that we have seen.

The problem with Deep Learning methods is that they require big architecture and a lot of data to generalize patterns and trends. Therefore, they can mimic patterns in the short term but tend to carry the error being made in higher horizons without adjusting the series. Similarly, *MLP* treats each vector of inputs as independent variables (i.i.d), so the model tends to learn a mean bias rather than patterns.

However, it is true that the results are quite good given that the actual value of the ^GSPC is around 6000pts, we are only committing between a 2.5% in short-term forecasts and 5% error in longer horizon errors.

When analysing stocks, it is important to know the confidence in the prediction for each horizon. The comparison for different horizons and actual values for the *DLM* and the *ARIMA* are shown in Figure 10:

These plots show an interesting comparison between *DLMs* and *ARIMA*. Generally, adjusting a

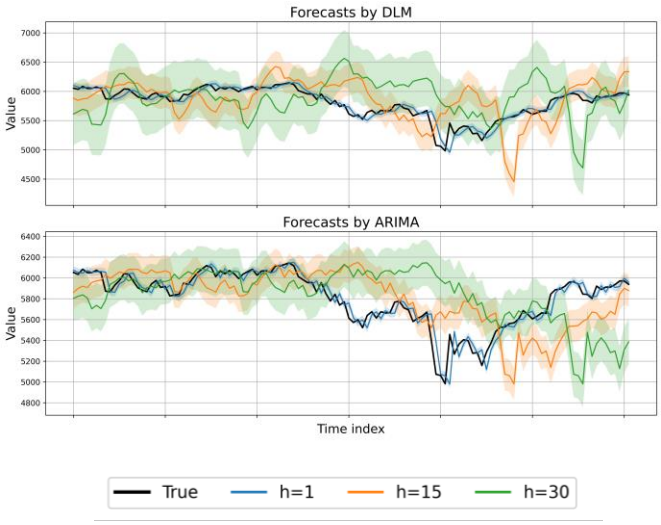

Figure 10 Forecast + 1 stdev for the S&P500

DLM requires to define a prior on the uncertainty that defines how confident you are on your data. In our case, and by estimating it with our algorithm, it has shown that the DLM has a lot more uncertainty than the ARIMA model. However, their behaviour continues to be quite difference, although as it can be seen at the lowest points in the plot, the ARIMA tends to forecast in the future close to linearly, and

the DLM has some seasonality (as the horizon increases, the forecast goes lower and then upwards). Both models have periods that when there is a long regime, (see the orange horizons where the slope is highest), the uncertainty shrinks completely. This is because if your information continues to tell you that you are moving in the correct direction, then you are surer of those predictions.

However, this plot can be better understood if we look at Figure 11:

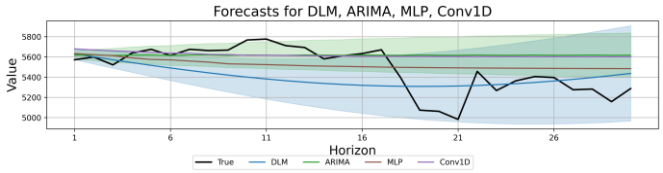

Figure 11 Multi-step forecast with stdev for different models.

The plot shows how the forecasts vary with increasing horizon from a specific index (around the lowest point in Figure 10). The most important characteristic this plot shows is how uncertainty grows with the forecast horizons, and most importantly, how different it is between *ARIMA* and *DLM*.

In Table 1 from Section 4.1 we argued that a lower ECE value shows that the model is better calibrated, which can be seen graphically in Figure 11. The *DLM* distribution tends to include more of the times the true series, while the *ARIMA* has a thinner confidence interval. Although sometimes it is preferable to have a smaller

Moreover, we can see the seasonal component in correctly adapts to the new trend in the data, while all other models behave smoothly, accumulating a lot of error due to their slow change in regime. The models that we have not shown perform worse. Also, we can see how the amplitude of the interval increases noticeable for the DLM for around 600, as this yields that the interval for one standard deviation is 20%.

In conclusion, we have seen that once we have used the models with real data, their performance has gone down noticeably. In fact, it proves that over longer horizons it is sometimes more appropriate to use averaging methods than complex models. In short term horizons ARIMA and DLM work best, with an error around 2%-5% of the value, but as the horizon increases, the uncertainty over the forecast becomes big. Also, Deep Learning models have underperformed since they need more layers and data to generalize better as the series becomes noisier.

Stock reconciliation

Now that we have seen the performance on a pure forecasting task, we can test how well reconciliation methods work on predicting distinct levels of the S&P500. In Table 6 we have shown the average CRPS (proper scoring metrics) values at different levels in the hierarchy, along with their unreconciled forecasts. The model used has been the DLM, as we saw the fastest performance in previous experiments and good accuracy on short forecasts:

Prob. Method	Rec. Method	Index	Sector	Industry	Stock
Base	Base	**178.40 ± 10.48**	**9.92 ± 0.59**	**11.91 ± 0.72**	12.96 ± 0.78
Gaussian	Classic	121791.84 ± 3986.63	668.98 ± 26.37	67.00 ± 3.35	12.96 ± 0.78
	Regression	8953.94 ± 287.90	58.50 ± 2.29	27.77 ± 1.16	12.96 ± 0.78
	MinT OLS	1267.84 ± 45.07	33.62 ± 1.20	27.88 ± 1.15	13.42 ± 0.79
	MinT Sample	488.35 ± 14.03	25.65 ± 0.88	28.54 ± 1.17	22.43 ± 0.88
	MinT Shrink	499.47 ± 13.90	25.80 ± 0.88	28.55 ± 1.16	22.27 ± 0.88
	MinT Shrink LW	451.22 ± 13.28	25.82 ± 0.87	28.50 ± 1.16	21.48 ± 0.85
	WLS_V	671.71 ± 29.59	31.12 ± 1.09	27.60 ± 1.15	13.48 ± 0.79
	WLS_S	5417.19 ± 172.04	59.40 ± 2.31	27.66 ± 1.15	16.74 ± 1.00
Monte Carlo	Classic	166352.57 ± 5257.51	823.19 ± 33.12	71.59 ± 3.66	**12.47 ± 0.68**
	Regression	10580.25 ± 323.58	58.85 ± 2.35	26.98 ± 1.16	**12.47 ± 0.68**
	MinT OLS	1199.39 ± 44.23	31.35 ± 1.14	26.96 ± 1.14	12.80 ± 0.68
	MinT Sample	438.99 ± 12.69	23.22 ± 0.81	27.46 ± 1.15	21.01 ± 0.82
	MinT Shrink	449.04 ± 12.61	23.28 ± 0.81	27.49 ± 1.15	20.74 ± 0.81
	MinT Shrink LW	403.59 ± 11.84	23.28 ± 0.81	27.47 ± 1.14	19.92 ± 0.78
	WLS_V	616.37 ± 27.96	28.97 ± 1.03	26.69 ± 1.13	12.88 ± 0.69
	WLS_S	5544.50 ± 173.08	56.75 ± 2.26	26.71 ± 1.14	15.88 ± 0.90

Table 6 CRPS for probabilistic reconciliation across levels of hierarchy in S&P500

If we recall the meaning CRPS, it measures the accuracy of the probabilistic forecasts by combining a measurement of its sharpness and calibration. This means that a lower CRPS is a better score for a distribution. MAE results will be provided in the appendix.

Table 6 shows that reconciliation has only improved the forecasts at the stock level, and the base forecast still ranks as the best in the other three. It is coherent that *Gaussian* and *base* forecasts are the same for

Classic and *Regression* at this level since the summing matrix does not affect them. However, by sampling 1000 forecasts in *Monte Carlo,* we have improved it, showing the effect of averaging the results. At this level, the methods that are based on residuals have underperformed considerably, as the information provided by the reconciliation matrix is not useful.

Levels sector and industry behave quite similar. In fact, *MinT* methods tend to stay in the same order as the stock CRPS values. However, they are still around 2.5x the base forecasts CRPS values. The rationale behind this could be that errors are accumulated when reconciling the forecasts instead of improving them. Following the same trend as in stock level, *Monte Carlo* tends to work better on point forecasting than *Gaussian*, but the standard error is a bit bigger. Again, thanks to sampling effect rather than closed-form equations. Here we see that when using different weights for each series, *Classic* method performs badly, same as *Regression* and *WLS$_S$*. *MinT OLS*, which is also deterministic on the hierarchical structure performs surprisingly better than them.

However, the biggest errors are seen at the *Index* level compared to the base forecast. Clearly, *MinT Sample, MinT Shrink* and *MinT Shrink* have worked the best with these models, as *Monte Carlo* for the probabilistic reconciliation. This makes sense as they are methods capable of minimizing residuals covariance with the reconciliation matrices, although not good enough as in the Section 4.1 experiment with synthetic data.

According to (Gneiting & Raftery, 2007), a relationship between MAE and CRPS can be done following that they are in the same scale. If CRPS is considerably lower than MAE, then the distribution prediction is better than just point forecasting and is well calibrated. In Table 7 we can see the same plot as Table 6, but with the MAE error:

Prob. Method	Rec. Method	Index	Sector	Industry	Stock
base	base	**248.10 ± 13.79**	**13.48 ± 0.75**	**15.73 ± 0.86**	**17.16 ± 0.94**
Gaussian	Classic	171006.54 ± 5262.74	891.07 ± 33.68	87.02 ± 4.02	**17.16±0.94**
	Regression	11522.17 ± 327.50	67.21 ± 2.49	28.37 ± 1.19	**17.16±0.94**
	MinT OLS	1369.18 ± 45.55	35.29 ± 1.22	28.17 ± 1.15	17.58 ± 0.94
	MinT Sample	608.84 ± 15.38	29.07 ± 0.93	29.11 ± 1.18	28.95 ± 1.10
	MinT Shrink	619.73 ± 15.21	29.03 ± 0.92	29.06 ± 1.17	28.47 ± 1.07
	MinT Shrink LW	565.78 ± 14.66	28.77 ± 0.91	28.96 ± 1.16	27.28 ± 1.02
	WLS$_V$	769.32 ± 30.78	32.86 ± 1.12	27.90 ± 1.15	17.65 ± 0.95
	WLS$_S$	5950.12 ± 174.05	62.44 ± 2.33	28.00 ± 1.16	21.61 ± 1.17
Monte Carlo	Classic	171006.01 ± 5262.64	891.07 ± 33.68	87.03 ± 4.02	17.17 ± 0.94
	Regression	11523.19 ± 327.57	67.22 ± 2.49	28.37 ± 1.19	**17.16±0.94**
	MinT OLS	1369.21 ± 45.55	35.29 ± 1.22	28.17 ± 1.15	17.59 ± 0.94
	MinT Sample	609.04 ± 15.40	29.07 ± 0.93	29.11 ± 1.18	28.95 ± 1.10
	MinT Shrink	619.90 ± 15.22	29.03 ± 0.92	29.06 ± 1.17	28.48 ± 1.07
	MinT Shrink LW	565.91 ± 14.65	28.77 ± 0.91	28.96 ± 1.16	27.28 ± 1.02
	WLS$_V$	769.29 ± 30.79	32.86 ± 1.12	27.90 ± 1.15	17.66 ± 0.95
	WLS$_S$	5950.14 ± 174.03	62.44 ± 2.33	28.00 ± 1.16	21.62 ± 1.17

Table 7 MAE at different levels in the hierarchy per reconciliation method

If we compare the values in the CRPS table, then we can see that generally distribution forecasts improve point. This is most noticeable at the Index level, where there is a difference of around 150pts. In levels like Industry the difference is small, which means that the distributions are poorly calibrated. This can be checked in the NLL plot provided in Appendix B.

However, if we just analyse the errors from Table 7, we can see that reconciliation has not improved predictions. There are several reasons for this that we will highlight in the following paragraphs.

This dataset has around 630 unique time series that makes a huge covariance matrix. As highlighted in Section 3.3, one of the key issues with *Mint* methods is numerical instability with inverse matrices. In (Wickramasuriya, Athanasopoulos, & Hyndman, 2018), the authors mention that estimating W is very hard when the number of samples is small compared to the number of series, as the matrix becomes rank deficient and singular. That is the reason *MinT Shrink* was created, to decrease the condition of the matrix to make it more inversible. Having a matrix this big, with numbers in the thousands could easily induce near zero values after the first inverse. These results can be seen in the appendix.

At that point, we were testing without scaling our data. Once we changed that, results improved drastically. This partly fixed the issue, but wanting to improve the condition number we developed the current algorithm and implemented *MinT Shrink*

LW. Given that this method optimizes finding the shrinkage constant λ, it has achieved the best results, demonstrating the importance of the process. However, this is not something that should be exploited as information gained from the residuals is lost. These results can also be found in the appendix.

Secondly, and for the results shown in Table 6, we also tweaked the summing matrix to instead of being filled with 1s and 0s, it is estimated with the actual weights using the regression algorithm. As mentioned in Section 3.3, this reinterpretation of the coherent constraint for a reconciliation matrix still holds, as the nodes are still logically connected. This also improved the results considerably.

Lastly, there is an issue with the method used when constructing the dataset. We have followed a bottom-up technique to calculate the weight of each stock towards each group but stopped at the sector level. We have assumed that the ^GPSC value downloaded from yfinance matches what should be without method. That does need to be necessarily true, as there are changes in the index that we have not included for simplicity and could end up breaking the coherent constraint. Also, we must account that market capitalization in the S&P500 can shift noticeably. We have assumed that in the two years of data we have available market caps are constant with today's value. Table A.6 in the appendix, provided at (Sather, 2023) shows how there are some sectors that can have shifts of ~4% in one year, or even more if there are redefinitions. This is not accounted in our index, along with companies that leave or are added to the index.

The last analysis we wanted to do is quantitatively evaluate the overhead time reconciliation means over just predicting with the base forecast. In Tables 8 and 9 we can see different averaged time metrics for reconciliation methods and probabilistic methods from these experiments.

Prob. Method	Forward Pass
Gaussian	1.0167

Table 9 Forward pass for probabilistic methods

As we expected, *Gaussian* takes around 1/3s of what *Monte Carlo* probabilistic reconciliation takes. This is because *Gaussian* only consists of performing one big matrix multiplication, while *Monte Carlo* needs to do it over many K (1000 for us) samples.

From Table 9 we can see that *Classic* is clearly the fastest method when inferring the reconciliation matrix. *Regression* is fast even though it has many fittings it needs to do It is in line with *MinT OLS*, WLS_S and WLS_V. These are inferred from the summing matrix, and for WLS_V, only needs to calculate variance from each node. Even though we expected *MinT Sample* to take the longest, *MinT Shrink* took more time because it had to estimate λ. If we were to choose the best method from these, probably *MinT Shrink LW* is the most appropriate.

In conclusion, we have seen that good real-world results with reconciliation techniques are hard to get. You need to have full knowledge on your series

Rec. Method	Matrix Init
Classic	**0.0410**
Regression	0.2844
MinT OLS	0.4861
MinT Sample	1.4914
MinT Shrink	1.6066
MinT Shrink LW	1.4693
WLS_V	0.5113
WLS_S	0.4918

Table 8 Matrix Initialization time per reconciliation method

and maintain the coherent requirement for reconciliation. Results have not been particularly good compared to the base forecast, although the trend we saw with *MinT Sample, Shrink* and *Shrink LW* leading in the synthetic experiment continues to hold. These methods have also been the slowest when creating their reconciliation matrix, although that was expected as they had the heavy computations to calculate covariance matrices Lastly, the CRPS value has shown that *Monte Carlo* tends to work better in terms of sharpness and calibration over *Gaussian*, given the averaging

effect on the reconciled forecasts after sampling. This comes at the cost of taking 3x longer to perform the forward pass to get the reconciled predictions.

5 BUSINESS IMPLEMENTATION

In the introduction for this project, we wanted to find a time-series forecasting model that provided accurate, reliable and fast predictions for environments where there is an important time-constraint. As an example, we previously saw mentioned (Naveiro, Rodríguez, & Ríos Insúa, 2018) as an example of DLMs in cybersecurity given its scalability capabilities. In this Section, we will devise an infrastructure applicable to a financial trading desk, based on an actual project:

We're designing an internal data-ingestion and forecasting server that pulls market data, stores it for on-demand access, runs live predictive models, and tracks performance. Key requirements:

- ***Efficient data collection**. Handle multiple sources (Bloomberg, Murex, YFinance), respecting API rate limits.*
- ***Fast, reliable storage**. Allow real-time writes and query reads so traders can access up-to-date information.*
- ***Live forecasting engine**. Run a primary model (plus a secondary backup) on the freshest data and log all outputs.*
- ***Comprehensive monitoring**. Track data-ingestion health, API usage, forecast accuracy, and system performance.*

To fulfil this, we have devised the following infrastructure:

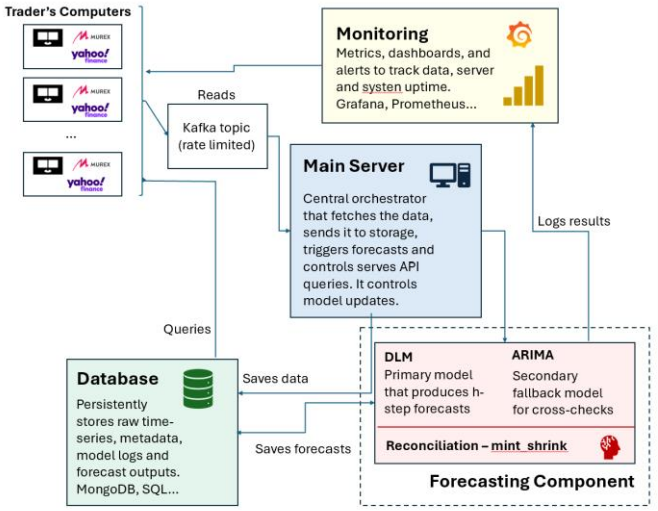

Figure 12 Trading Desk Infrastructure Example

This component division ensures that all traders machines contribute to the server's querying capacity. Each of them will connect to the main server with a fast Kafka topic. The server will distribute the data for storage and forecasting, while managing monitoring and user queries.

As shown in Figure 12, the forecasting component will provide traders information to anticipate market trends. As we saw in Section 4, DLM models provide fast, reliable short-term forecasts. To address their long-term issues shown in Section 4.2, we will run an ARIMA model as a backup, enhancing robustness and allowing cross-evaluation between models. Both will deliver uncertainty estimates, highly valuable for traders as a risk measure. We have discarded all deep learning models due to their complexity compared to these cases.

After forecasting, a hierarchical reconciliation step is applied. After seeing the complexity of this technique, we still believe it is still useful in cases with more data points and a static hierarchical structure. That conditions could still be applied to some financial products, and hence, for this case. Given the data scale and performance needs, we will use *MinT Shrink LW* with *Monte Carlo*, a combination that improves base forecasts and balances computational efficiency as shown in Section 4.1.

Key forecasting processes, like model initialization and update will run automatically. DLM and ARIMA models are easy to update, and thanks to our algorithms developed in Sections 3.1.3 and 3.1.4, they will fit automatically. This redundant infrastructure also prevents any forecasting down time by having DLMs and ARIMA running in parallel and will also allow us to compare its forecasts to trigger model updates.

6 CONCLUSIONS AND FUTURE WORK

In this project we have overviewed the main forecasting models along with different hierarchical reconciliation techniques, including probabilistic methods. All the combinations have been tested in a controlled environment with synthetic data and with S&P500 Index data from the last 2 years. These experiments have been done in scope of finding suitable models and techniques to summarize a lot of data into improved predictions in environments where having fast predictions is key.

Initial forecasting experiments have shown that *ARIMA* and *DLM* models have worked the best in short-term horizons. In the first case, it continued to be one of the best performing in longer forecasting periods, as the error did not increase that much. However, DLMs have shown a very good performance in initial steps and when the data easily resembles its components. In the synthetic experiment, where the model easily adjusted to seasonality it performed well in longer horizons. However, in noisier data from the S&P500, performance dropped noticeably.

Still, these two models provide a probabilistic advantage compared to deep learning. Many of the real-world application of time-series forecasting, and hierarchical reconciliation, require some knowledge of confidence in these forecasts to be aware of how much risk there is attached to them.

From Table 1 we saw that *DLM* was better than *ARIMA*. This was strengthened by Figure 11, where the *DLM* distribution forecast seems better calibrated although considerable wider.

Deep Learning models have provided average results but are not a viable alternative given their issues with scalability and reliance on training processes. This is a big downfall that is solved efficiently by *DLM* and *ARIMA*. Lastly, *SMA* and *EMA* provide a good baseline to start with these models, especially in series that are very constant. The results on the S&P500 show that on longer horizons sometimes averaging results in better results.

The next step in the project was testing the reconciliation methods. To do so, we decided to use the *DLM* given its accuracy on short horizons and, specially, its adaptability and speed to make new forecasts and update. These reconciliation techniques are meant to improve base forecasts for a set of time series, but as it has been seen, it is not always the case.

Section 4.1 showed the true benefits of reconciliation on a very simple structure. There has been a clear difference between the models that rely on past residuals and not. In that second group, *Classic, Regression, MinT OLS and WLS S* have provided worse results compared to forecasting upper time series separately. However, when there is full information from past forecast, then it is better to use any of the other *MinT* methods. As shown in Sections 3.3 and 4.2, it would make most sense to use *MinT Shrink LW* as the resources needed to estimate the covariance matrix are high and this simplifies them.

Although some point reconciliation methods have seemed useful, probabilistic reconciliation has not been that positive. Initially, *Montecarlo* gave improved results compared to the base forecasts, but in real data was unproductive. *Gaussian* on the other hand behaves similarly in longer-term forecasts but has the added benefit of a lower computing time.

However, S&P500 experiment showed the big drawbacks of using reconciliation methods. Reconciliation results were far away from base forecasts. Although this was not the expected result, it makes sense given the low data availability and singular matrices that need to be inversed and produce numerical instability. The best methods identified in 4.1 continued to perform best within reconciliation techniques.

In Section 5 we showed a possible implementation for a real-world case scenario for a trading desk, where there is special need for fast, reliable and accurate forecasts. We concluded that the best combination to implement this research would be using DLMs as primary forecasting method, with ARIMA being used as cross-checker and validation. Following this step, reconciliation could be done using (*MinT Shrink LW, Montecarlo),* making the model and reconciliation matrices updates simple and easily triggered when needed.

6.2 *FUTURE WORK*

Although the project has assessed a varied amount of forecasting and reconciliation methods, there are more that could be evaluated.

Although there are not many different models in time-series forecasting, prophet could have been added to the comparison. However, the research could be most benefitted by including probabilistic deep learning methods so further comparison with *ARIMA* and *DLM* could be done. These methods were evaluated, although they were ultimately discarded due to computational limitations (which is incidentally one of the main research lanes for these models).

Moreover, there are not many more different state-of-the-art reconciliations techniques. However, the methods used still show space for improvement, especially in areas where there is not much information about the upper series. For example, a good idea that could be evaluated is changing the summing matrix for the regression method, as this one has more information about the relationship of the series. The most different thing that could be tested is making W reconciliation matrix temporal dependent. There are versions of pure probabilistic reconciliation techniques and structure discovery to build interdependence. that could be worth testing with more resources. A good example is a full Bayesian model of the interaction parameters, although it is high resource demanding.

Lastly, it could be positive to test reconciliation methods with more financial data. One of the main reasons the techniques did not perform as expected was due to the need of a minimum length in the series, which we did not have available. Or this could be tested in another application.

7 BIBLIOGRAPHY

Athanasopoulos, G., Hyndman, R. J., Kourentzes, N., & Panagiotelis, A. (2023). *Forecast reconciliation: A review.* Retrieved from https://robjhyndman.com/papers/hf_review.pdf

Box, G. E., & Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control.* Holden-Day. Retrieved from https://books.google.es/books/about/Time_Series_Analysis.html?id=1WVHAAAAMAAJ&redir_esc=y

Dive Into Deep Learning. (2022). *10.1 Long Short-Term Memory (LSTM).* Retrieved from Dive Into Deep Learning: https://d2l.ai/chapter_recurrent-modern/lstm.html

Fortune Business Insights. (2025). *Big Data Analytics Market.* Fortune Business Insights. Retrieved from https://www.fortunebusinessinsights.com/big-data-analytics-market-106179

Gneiting, T., & Raftery, A. E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American*

Statistical Association, 102(477), 359-378. doi:https://doi.org/10.1198/016214506000001437

Grand View Research. (2024). *High Frequency Trading Market Size.* Next Generation Technologies. Grand View Research. Retrieved from https://www.grandviewresearch.com/industry-analysis/high-frequency-trading-market-report

Gratton, P. (2024, November 22). *What Is a Bloomberg Terminal (BT)?Functions, Costs, and Alternatives*. Retrieved from Investopedia: https://www.investopedia.com/terms/b/bloomberg_terminal.asp

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780. Retrieved from https://www.bioinf.jku.at/publications/older/2604.pdf

Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: Principles And Practice. In R. J. Hyndman, & G. Athanasopoulos, *Forecasting: Principles And Practice* (3rd ed.). Melbourne, Australia: OTexts. Retrieved from https://otexts.com/fpp3/single-level.html

Jani Data Diaries. (2024, November 7). *Choosing the Best Model: A Friendly Guide to AIC and BIC*. Retrieved from Medium: https://medium.com/@jshaik2452/choosing-the-best-model-a-friendly-guide-to-aic-and-bic-af220b33255f

Lea, C., Vidal, R., Reiter, A., & Hager, G. D. (2016). Temporal Convolutional Networks: A Unified Approach to Action Segmentation. *G. Hua & H. Jégou (Eds.), Computer Vision – ECCV 2016 Workshops, 9915*, pp. 47-54. doi:https://doi.org/10.1007/978-3-319-49409-8_7

Leodit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis, 88*(2), 365-411.

Locruz, B., Lasala, P., & Lekuona, A. (2000). Graphical dynamic linear models: specification, use and graphical transformations. *International Journal of Approximate Reasoning, 24*(1), 83-102.

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

Musbah, H., El-Hawary, M., & Aly, H. (2019). Identifying Seasonality in Time Series by Applying Fast Fourier Transform. *2019 IEEE Electrical Power and Energy Conference (EPEC).* Montreal, QC, Canada: IEEE. doi:10.1109/EPEC47565.2019.9074776

Naveiro, R., Rodríguez, S., & Ríos Insúa, D. (2018). Large Scale Automated Forecasting for Monitoring Network Safety and Security. *Applied Stochastic Models in Business and Industry, 35*(3). doi:10.1002/asmb.2436

Palande, C., & Recasens, J. (2019, October 11). *Hierarchical Time Series 101*. Retrieved from Medium: https://medium.com/opex-analytics/hierarchical-time-series-101-734a3da15426

Panagiotelis, A., Gamakumara, P., Athanasopoulos, G., & Hyndman, R. J. (2023). Probabilistic forecast reconciliation: Properties, evaluation and score optimisation. *European Journal of Operational Research, 306*(2), 693-706. doi:https://doi.org/10.1016/j.ejor.2022.07.040

Quinn, C. O., Corliss, G. F., & Povinelli, R. J. (2024). Cross-Temporal Hierarchical Forecast Reconciliation of Natural Gas Demand. *MDPI Energies, 17*(13), 3077. doi:https://doi.org/10.3390/en17133077

Sather, A. (2023, July 12). *Historical S&P 500 Industry Weights – [20+ Year History]*. Retrieved from einvestingforbeginners: https://einvestingforbeginners.com/historical-sp-500-industry-weights-20-years/

statsmodels. (2024, October 3). *statsmodels.tsa.arima.model.ARIMA*. Retrieved from statsmodels.org: https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html

Taieb, S. B., Taylor, J. W., & Hyndman, R. J. (2017). Coherent Probabilistic Forecasts for Hierarchical Time Series. *Proceedings of Machine Learning Research, 70*, 3348-3357. Retrieved from https://proceedings.mlr.press/v70/taieb17a.html

Taylor, S. J., & Letham, B. (2017). Forecasting at Scale. 25. doi:https://doi.org/10.7287/peerj.preprints.3190v2

Tchoketch-Kebir, H., & Madouri, A. (2024). Research Leadership and High Standards in Economic Forecasting: Neural Network Models Compared with Etalon ARIMA Models. *Business Ethics and Leadership*, 220-233. doi:http://doi.org/10.61093/bel.8(1).220-233.2024

Wall Street Prep. (2025, April 7). *Bloomberg vs. Capital IQ vs. Factset vs. Refinitiv*. Retrieved from Wall Street Prep: https://www.wallstreetprep.com/knowledge/bloomberg-vs-capital-iq-vs-factset-vs-thomson-reuters-eikon/

West, M., & Harrison, J. (1997). *Bayesian Forecasting and Dynamic Models* (2 ed.). New York: Springer. doi:https://doi.org/10.1007/b98971

Wickramasuriya, S. L., Athanasopoulos, G., & Hyndman, R. J. (2018). Optimal Forecast Reconciliation for Hierarchical and Grouped Time Series Through Trace Minimization. *Journal of the American Statistical Association, 114*(526), 804-819. doi:https://doi.org/10.1080/01621459.2018.1448825

Wong, J. (2024, August 20). *https://medium.com/@ycwong.joe/a-brief-history-of-time-series-models-38455c2cd78d*. Retrieved from Medium: https://medium.com/@ycwong.joe/a-brief-history-of-time-series-models-38455c2cd78d

8 APPENDIX A – SYNTHETIC SERIES

FORECASTING

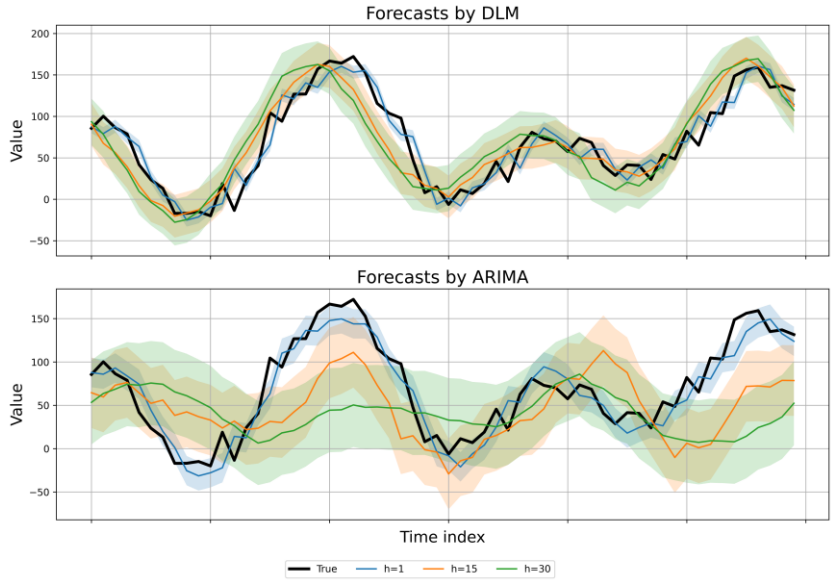

Figure A.1 Forecast prediction ± 1stdev for DLM and ARIMA

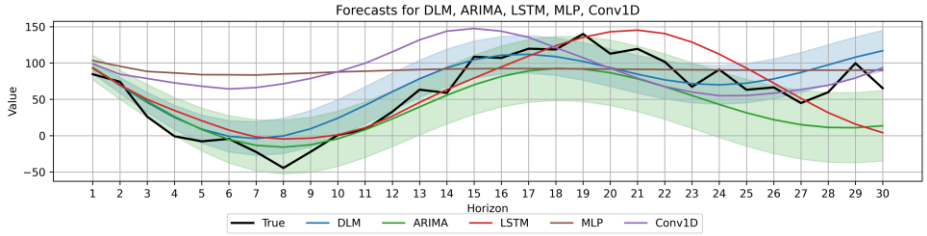

Figure A.2 Multi Step forecast across different horizons

RECONCILIATION

Prob. Method	Rec. Method	1–6	7–12	13–18	19–24	25–30
Base	Base	10.34 ± 0.23	10.64 ± 0.24	10.44 ± 0.23	12.20 ± 0.27	13.63 ± 0.31
Gaussian	Classic	9.53 ± 0.21	9.97 ± 0.22	10.17 ± 0.23	11.73 ± 0.26	13.00 ± 0.29
	MinT OLS	10.43 ± 0.23	10.52 ± 0.24	10.28 ± 0.23	12.07 ± 0.27	13.50 ± 0.30
	MinT Sample	9.16 ± 0.21	9.35 ± 0.21	9.07 ± 0.20	10.92 ± 0.25	10.50 ± 0.24
	MinT Shrink	9.17 ± 0.21	9.35 ± 0.21	9.11 ± 0.20	10.93 ± 0.25	10.52 ± 0.24
	MinT Shrink LW	9.17 ± 0.21	9.35 ± 0.21	9.10 ± 0.20	10.93 ± 0.25	10.51 ± 0.24
	Regression	9.53 ± 0.21	9.97 ± 0.22	10.17 ± 0.23	11.73 ± 0.26	13.00 ± 0.29
	WLS_S	10.08 ± 0.23	10.21 ± 0.23	10.03 ± 0.23	11.85 ± 0.27	13.26 ± 0.30
	WLS_V	10.00 ± 0.22	10.18 ± 0.23	9.99 ± 0.22	11.85 ± 0.27	13.38 ± 0.30
Monte Carlo	Classic	9.53 ± 0.21	9.97 ± 0.22	10.17 ± 0.23	11.73 ± 0.26	13.00 ± 0.29
	MinT OLS	10.43 ± 0.23	10.52 ± 0.24	10.28 ± 0.23	12.07 ± 0.27	13.50 ± 0.30
	MinT Sample	9.16 ± 0.21	9.35 ± 0.21	9.07 ± 0.20	10.92 ± 0.25	10.51 ± 0.24
	MinT Shrink	9.17 ± 0.21	9.35 ± 0.21	9.11 ± 0.20	10.94 ± 0.25	10.53 ± 0.24
	MinT Shrink LW	9.17 ± 0.21	9.35 ± 0.21	9.10 ± 0.20	10.93 ± 0.25	10.51 ± 0.24
	Regression	9.53 ± 0.21	9.97 ± 0.22	10.17 ± 0.23	11.74 ± 0.26	13.00 ± 0.29
	WLS_S	10.08 ± 0.23	10.21 ± 0.23	10.03 ± 0.23	11.85 ± 0.27	13.26 ± 0.30
	WLS_V	10.00 ± 0.22	10.18 ± 0.23	9.99 ± 0.22	11.85 ± 0.27	13.38 ± 0.30

Table A.1 RMSE averaged for different horizons using reconciliation techniques and Regression as S matrix

Prob. Method	Rec. Method	1–6	7–12	13–18	19–24	25–30
Base	Base	0.144 ± 0.026	0.08 ± 0.015	0.075 ± 0.015	0.077 ± 0.016	0.095 ± 0.020
Gaussian	Classic	0.128 ± 0.024	0.125 ± 0.024	0.127 ± 0.025	0.119 ± 0.024	0.123 ± 0.026
	MinT OLS	0.246 ± 0.045	0.137 ± 0.025	0.110 ± 0.020	0.118 ± 0.021	0.136 ± 0.025
	MinT Sample	0.257 ± 0.054	0.182 ± 0.037	0.290 ± 0.061	0.263 ± 0.057	0.314 ± 0.067
	MinT Shrink	0.226 ± 0.047	0.171 ± 0.035	0.249 ± 0.053	0.241 ± 0.052	0.301 ± 0.065
	MinT Shrink LW	0.216 ± 0.045	0.173 ± 0.035	0.264 ± 0.056	0.247 ± 0.054	0.309 ± 0.066
	Regression	0.128 ± 0.024	0.125 ± 0.024	0.127 ± 0.025	0.119 ± 0.024	0.123 ± 0.026
	WLS_S	0.252 ± 0.047	0.154 ± 0.028	0.121 ± 0.022	0.122 ± 0.023	0.134 ± 0.025
	WLS_V	0.250 ± 0.046	0.153 ± 0.028	0.116 ± 0.021	0.118 ± 0.022	0.132 ± 0.025
Monte Carlo	Classic	0.308 ± 0.057	0.272 ± 0.050	0.255 ± 0.046	0.263 ± 0.048	0.269 ± 0.050
	MinT OLS	0.361 ± 0.069	0.327 ± 0.062	0.311 ± 0.059	0.317 ± 0.060	0.317 ± 0.061
	MinT Sample	0.150 ± 0.027	0.204 ± 0.037	0.157 ± 0.029	0.135 ± 0.024	0.100 ± 0.019
	MinT Shrink	0.178 ± 0.032	0.211 ± 0.038	0.143 ± 0.026	0.156 ± 0.028	0.103 ± 0.019
	MinT Shrink LW	0.187 ± 0.034	0.211 ± 0.038	0.134 ± 0.024	0.148 ± 0.027	0.100 ± 0.018
	Regression	0.308 ± 0.057	0.272 ± 0.050	0.255 ± 0.046	0.263 ± 0.048	0.269 ± 0.050
	WLS_S	0.362 ± 0.069	0.329 ± 0.062	0.311 ± 0.059	0.317 ± 0.060	0.320 ± 0.061
	WLS_V	0.361 ± 0.069	0.328 ± 0.062	0.307 ± 0.058	0.311 ± 0.059	0.315 ± 0.060

Table A.2 ECE for different horizons using reconciliation techniques and base forecasts

9. APPENDIX B – S&P500 EXPERIMENT

FORECASTING

		Horizon		
Metric	Model	1–6	13–18	25–30
CRPS	ARIMA	75.14 ± 2.86	159.77 ± 4.81	210.22 ± 6.84
	DLM	82.16 ± 3.10	187.89 ± 5.55	241.85 ± 5.84
NLL	ARIMA	7.00 ± 0.16	7.44 ± 0.07	7.72 ± 0.08
	DLM	6.79 ± 0.11	7.30 ± 0.04	7.49 ± 0.02
ECE	ARIMA	0.12 ± 0.01	0.15 ± 0.01	0.12 ± 0.01
	DLM	0.09 ± 0.01	0.06 ± 0.00	0.03 ± 0.00

Table A.3 Probabilistic Metrics for DLM and ARIMA

Figure A.3 Point forecasts for all models

RECONCILIATION

Prob. Method	Rec. Method	Index	Sector	Industry	Stock
Base	Base	295.02 ± 25.52	14.36 ± 1.21	16.12 ± 1.34	17.15 ± 1.43
Gaussian	Classic	102785.09 ± 687.86	9617.06 ± 70.11	624.74 ± 7.70	17.15 ± 1.43
	Regression	273.61 ± 23.18	667.03 ± 6.31	207.01 ± 1.67	17.15 ± 1.43
	Mint_ols	563.95 ± 36.58	504.64 ± 5.03	194.83 ± 1.57	34.07 ± 1.65
	Mint_shrink	5397.90 ± 28.22	217.75 ± 1.24	222.91 ± 1.46	215.68 ± 1.41
	Mint_shrink_lw	5316.36 ± 28.07	210.79 ± 1.42	222.27 ± 1.47	208.10 ± 1.37
	WLS_V	3061.08 ± 25.71	232.48 ± 2.80	205.09 ± 1.50	58.58 ± 1.66
	WLS_S	1565.25 ± 31.03	674.76 ± 6.20	201.22 ± 1.60	63.11 ± 2.27
Monte Carlo	Classic	102785.03 ± 687.86	9617.06 ± 70.11	624.74 ± 7.70	17.15 ± 1.43
	Regression	273.62 ± 23.18	667.03 ± 6.31	207.01 ± 1.67	17.15 ± 1.43
	Mint_ols	563.95 ± 36.58	504.64 ± 5.03	194.83 ± 1.57	34.08 ± 1.65
	Mint_shrink	5397.90 ± 28.22	217.75 ± 1.24	222.91 ± 1.46	215.68 ± 1.41
	Mint_shrink_lw	5316.36 ± 28.07	210.79 ± 1.42	222.27 ± 1.47	208.10 ± 1.37
	WLS_V	3061.08 ± 25.71	232.48 ± 2.80	205.09 ± 1.50	58.58 ± 1.66
	WLS_S	1565.25 ± 31.03	674.76 ± 6.20	201.22 ± 1.60	63.11 ± 2.27

Table A.4 MAE per levels of hierarchical structure and using regression as S matrix

Prob. Method	Rec. Method	Index	Sector	Industry	Stock
Base	Base	295.02 ± 25.52	14.36 ± 1.21	16.12 ± 1.34	17.15 ± 1.43
Gaussian	Classic	102785.09 ± 687.86	9617.06 ± 70.11	624.74 ± 7.70	17.15 ± 1.43
	Regression	273.61 ± 23.18	667.03 ± 6.31	207.01 ± 1.67	17.15 ± 1.43
	Mint_ols	300.00 ± 24.69	278.42 ± 4.41	195.17 ± 2.43	203.69 ± 2.30
	Mint_sample	5881.27 ± 29.21	261.70 ± 1.37	231.13 ± 1.65	220.15 ± 1.85
	Mint_shrink	5600.84 ± 28.02	236.20 ± 1.28	224.64 ± 1.39	214.90 ± 1.45
	Wls_v	1007.61 ± 30.73	181.36 ± 3.09	195.07 ± 2.12	205.76 ± 1.87
	Wls_s	30778.33 ± 222.10	3070.99 ± 25.10	179.97 ± 3.80	142.93 ± 2.05
Monte Carlo	Classic	102785.03 ± 687.86	9617.06 ± 70.11	624.74 ± 7.70	17.15 ± 1.43
	Regression	273.62 ± 23.18	667.03 ± 6.31	207.01 ± 1.67	17.15 ± 1.43
	Mint_ols	300.00 ± 24.69	278.42 ± 4.41	195.17 ± 2.43	203.69 ± 2.30
	Mint_sample	5881.26 ± 29.21	261.70 ± 1.37	231.13 ± 1.65	220.15 ± 1.85
	Mint_shrink	5600.85 ± 28.02	236.20 ± 1.28	224.64 ± 1.39	214.90 ± 1.45
	Wls_v	1007.62 ± 30.74	181.36 ± 3.09	195.07 ± 2.12	205.76 ± 1.87
	Wls_s	30778.33 ± 222.10	3070.99 ± 25.10	179.97 ± 3.80	142.93 ± 2.05

Table A.5 MAE per levels of hierarchical structure and using usual S matrix.

10. APPENDIX C – ADDITIONAL DOCUMENTS

	2023	2022	2021	2020	2019	2018	2017	2016	2015	2014	2013	2012	2011	2010
Communication Services	9%	8%	15%	14%	14%	13%	13%	13%	13%	10%	38%	33%	33%	32%
Consumer Discretionary	11%	9%	13%	13%	10%	10%	10%	9%	10%	9%	6%	6%	6%	6%
Consumer Staples	7%	8%	6%	7%	8%	8%	9%	10%	10%	10%	7%	8%	9%	8%
Energy	4%	5%	3%	2%	4%	4%	5%	6%	6%	7%	6%	7%	7%	7%
Financials	13%	14%	11%	10%	13%	13%	15%	15%	14%	16%	11%	11%	10%	12%
Health Care	13%	15%	13%	12%	13%	14%	12%	12%	14%	14%	9%	8%	8%	8%
Industrials	8%	9%	7%	8%	8%	8%	9%	10%	9%	10%	7%	7%	7%	8%
Information Technology	27%	23%	25%	26%	23%	22%	19%	17%	16%	16%	11%	14%	13%	12%
Materials	2%	3%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%
Real Estate	2%	3%	3%	2%	3%	3%	3%	3%	3%	3%	2%	2%	2%	2%
Utilities	2%	3%	2%	2%	3%	3%	3%	3%	3%	3%	2%	2%	3%	2%

Table A.6 Sector weights for S&P500 index during the years, according to (Sather, 2023)