Realized volatility forecasting with component GARCH enhanced Machine learning models in a Stacked Ensemble approach

A thesis submitted in partial fulfillment for the

degree of

Master of Science

(MSc) Financial Engineering

in the

Decision Sciences Faculty
HEC Montréal

Proposed by: Yissan Yaro

Supervisor: Christian Dorion

December 2023

Abstract

Volatility forecasting is a crucial task in financial markets, as accurate predictions can assist investors, risk managers, and policymakers in making informed decisions, providing valuable insights for risk management, portfolio optimization, and derivative pricing. Our undertaking here is to perform a comparative assessment of performance and robustness of machine learning models on the auto-regressive volatility point-forecasting problem across two dimensions, namely forecasting horizon and asset. It is also a study of the gain from enhancing machine learning models with component GARCH models, and the gain from building stacked ensemble models from our simple models. We use two state of the art econometric models as benchmarks, namely the component GARCH model itself, and the Heterogeneous Autoregressive (HAR) model. Preliminary results suggest the superiority of the CGARCH-enhanced models against the vanilla models only in the case of gradient boosting models, the superiority of CGARCH enhanced gradient boosting models over all other models (including our benchmarks), and that on both dimensions, there is no significant forecasting gain to stacking our models together in an ensemble. According to our evaluation framework, there was no particular outlier with respect to any specific asset. Overall, our most trustworthy results suggest that the predominant model in terms of robustness to both the asset and horizon dimensions appears to be the NG CGARCH model, and that there is an added-value in augmenting boosting models with a component GARCH model. We also find that there is no statistical difference in using either the NG CGARCH or the XG CGARCH for forecasting long term commodities volatility. By undertaking this research, the aim is to contribute to the growing body of knowledge in volatility forecasting. This study is the first documented to explore the use of the Natural Gradient Boosting algorithm for volatility forecasting, as well as to combine the 4 machine learning models with a component GARCH model in such an ensemble framework.

Contents

List of Figures

List of Tables

Acronyms

ADF Augmented Dickey-Fueller test for unit-root

ANN Artificial Neural Network

CART Classification And Regression Trees

CGARCH Component Generalized Autoregressive Conditional Heteroskedastic model

FFN Feed Forward neural Network

GARCH Generalized Autoregressive Conditional Heteroskedastic model

HV Historical Volatility

JB Jarque-Berra test for normality

LSTM Long Short Term Memory network

LSTM CGARCH Feed forward ensemble model combining LSTM and CGARCH models

MCS Model Confidence Set

MGW Multivariate Giacomini-White test

ML Machine learning

MLP Muli Layer Perceptron

NGBOOST Natural gradient Boosting model

NG CGARCH Feed forward ensemble model combining Ngboost and CGARCH models

OF Overvaluation Frequency

QLIKE Quasi Likelihood function

RelU Rectified Linear Unit

RNN Recurrent neural Network

RV Realized Volatility

TRANS CGARCH Feed forward ensemble model combining Transformer and CGARCH

XGBOOST Extreme Gradient Boosting model

XG CGARCH Feed forward ensemble model combining Xgboost and CGARCH models

X-N-L-T Feed forward ensemble model combining Xgboost, NgBoost, LSTM and Transformer

X-N-L-T CGARCH Feed forward ensemble model combining Xgboost, NgBoost, LSTM and Transformer and CGARCH model

1 Introduction

This study aims at evaluating the performance and mostly the robustness of recent machine learning models on the age old problem of multi-horizon volatility forecasting and comparing it with classical state of the art econometrics models. We study the robustness along two axes, which are the forecasting horizon (for which we have 5), and the asset on which the forecast is performed (of which we also have 5) .We also study the benefits of enhancing these machine learning models with a component GARCH model (Gary and Engle [3]) , as well as combining all of them in an ensemble approach. The machine learning architectures (refereed as vanilla models) considered here are the transformer network, the Long Short Term Memory Network, the Natural gradient boosting algorithm, and the Extreme gradient boosting algorithm. The proposed methodologies integrate the strengths of the component GARCH, which captures conditional heteroskedasticity, and the four machine learning architectures which excel in capturing long-range dependencies. The CGARCH model is utilized to estimate conditional volatility, which is then combine with the models as an ensemble. These combinations leverage the ability of the ML models to effectively learn complex patterns, different features and dependencies in the financial time series data. We also evaluate the gains of ensembling all the ML models together, as well as all the ML models with a CGARCH. To evaluate the performance of the proposed approaches, a comprehensive and rigorous empirical study is conducted using a dataset comprising daily realized volatility and returns from our 5 assets. The dataset is divided into training, validation, and test sets, with a static rolling window approach employed to simulate real-time forecasting. The developed models are compared against benchmark models, namely the CGARCH and the HAR model. Various performance metrics such as root mean square error (RMSE), mean absolute error (MAE), Overvaluation Frequency (OF) and Quasi Likelihood (QLIKE) are employed to assess and compare the forecasting accuracy of the models. We consider each forecasting horizon for each asset class as a separate problem. As we have 5 assets and 5 horizons, we find ourselves exploring 25 point forecasting problems, each to which we apply and evaluate 4 single models and 6 ensemble models (among which 5 are CGARCH-enhanced). It can be framed as a comparative study of vanilla models among themselves, vanilla models vs CGARCH enhanced models, and vanilla models vs ensemble models. Essentially, we try to find empirical answer to such questions as: Is there a specific model which demonstrates absolute robustness along both axes, meaning forecasting superiority across horizons and assets? Are there some models that demonstrate superiority for a specific horizon or a specific asset ? Is there a clear benefit across assets and/or horizon to enhancing our machine learning models with CGARCH or to ensembling all of them? This thesis addresses and answer these questions through a rigorous empirical study.

2 Litterature Review

In a comprehensive literature review of multi-horizon volatility forecasting models enhanced by machine learning techniques, it's essential to trace the evolution from traditional econometric models to the integration of advanced machine learning algorithms.
One challenge in studying volatility models' literature is to have a comprehensive enough classification framework. Previous studies by Engle and Patton [4] have attempted this exercise and classify volatility models in two parent categories: Models that formulate conditional volatility as a function of observables, and latent volatility models, often coined stochastic volatility models. All of the models explored in this study fall into the first category with respect to this classification. A subsequent seminal classification framework was proposed in Poon and Clive [5] and Poon and Granger [6] in their Review of volatility forecasting models, in which they classify volatility models in the following families: Time series Models based on past realization of standard-deviation family, the ARCH models family, the Stochastic volatility family, the non-parametric model's family, the options-based models family, and the machine learning models family. Ge et al. [1] conducted a systematic review, highlighting the efficacious deployment of neural networks in financial volatility forecasting, emphasizing the nuanced ability of these models to approximate both linear and nonlinear dynamics without prior knowledge of the data-generating process. Their study provides an in-depth analysis and comprehensive examination of the advances in neural network (NN) applications for financial volatility forecasting. The study examines 35 publications post-2015, identifying issues like the difficulty in meaningful comparisons between models due to a lack of standardization and the disparity between contemporary machine learning (ML) and financial forecasting models. The review identifies several key issues prevalent in the current landscape of neural network-based volatility forecasting. One of the primary challenges noted is the difficulty in conducting easy and meaningful comparisons between different models. This challenge stems from the diverse range of approaches and methodologies employed across studies, making it hard to directly compare their effectiveness or draw conclusive insights. Another challenge is the mere definition of volatility, which varies vastly from one study to another. The review reveals a preference for using Historical Volatility (HV) over theoretically more robust alternative other proxies like Implied Volatility (IV) or Realized Volatility (RV) in the literature, partly due to the ease of access to necessary data and simplicity of the models. Initially, volatility forecasting was dominated by econometric models, with the Autoregressive Conditional Heteroskedasticity (ARCH) model introduced by Engle [7] pioneering the field. The Generalized ARCH (GARCH) model by Bollerslev [8] further refined this approach by allowing past conditional variances to influence current estimates, addressing the persistence often observed in financial volatility. The literature then expanded to include models that account for asymmetries and leverage effects, such as the Exponential GARCH (EGARCH) model proposed by Nelson [9] and the Glosten-Jagannathan-Runkle (GJR) GARCH model (GLOSTEN et al. [10]). These models capture the phenomenon where negative shocks induce greater volatility than positive shocks of the same magnitude, a reflection of the leverage effect. With the advent of high-frequency trading and intraday data, realized volatility measures and proxies using high-frequency returns were proposed, enhancing the accuracy of volatility estimation.Andersen et al. [11] were instrumental in this development, leading to a variety of realized measures such as realized

variance and realized range. The subsequent development of realized volatility estimation and forecasting can be attributed to a series of papers by Barndorff-Nielsen and Shephard, Bandi and Russel, and Ait-Sahalia in which they extensively establish best practices in term of estimating, sampling and forecasting realized volatility. In the landscape of volatility forecasting, the Heterogeneous Autoregressive (HAR) model stands out due to its effectiveness and simplicity in capturing the dynamics of realized volatility. The HAR model can be viewed as a specific autoregressive application of the Ghysels et al. [12] MIDAS (Mixed Data Sampling) approach in the realm of volatility forecasting, where daily, weekly, and monthly volatilities are harmoniously integrated into a single forecasting model. It emerged from the need to address the limitations of existing models in capturing the long memory and multi-scale nature of volatility. Traditional models like ARCH and GARCH, while effective in many contexts, often fell short in accurately representing the persistence and heterogeneous market behaviors influencing volatility. First introduced by Fulvio Corsi's pioneering work in Corsi [13], it presented a novel approach to volatility modeling, allowing capturing the strong persistence observed in realized volatility at multiple scales (usually daily, weekly, and monthly) within a single framework. Its versatility is further evidenced by the introduction of logarithmic transformations of realized volatility or exogenous regressors, enhancing its compatibility with standard time series procedures. Empirical evidence from various markets, including equity, bond, and commodity markets, consistently underscores the HAR model's superior forecasting performance compared to traditional ARCH-family models. This versatility in application underscores the model's adaptability and robustness. Furthermore, recent advancements have seen the integration of machine learning techniques with HAR models, opening new avenues for enhancing forecasting accuracy.

As machine learning started to reshape many domains, its applications in volatility forecasting gained traction. The literature reveals a shift towards leveraging machine learning's ability to handle non-linearities and complex interactions within data. Initial efforts to forecast volatility with neural networks were made by Miranda and Burgess [14] , who used them to predict intraday volatilities for the Spanish stock market or Tino et al. [15] to show how they outperform ARCH models in predicting the volatility of the Austrian stock market. Additionally, an influential study by Hamid and Iqbal [16] found that ANN realized volatility forecasts on the S&P500 outperforms implied volatility from the Barone-Adesi and Whaley model in anticipating future realized volatility More recently, deep learning models, particularly Long Short-Term Memory (LSTM) networks, have been employed for their ability to capture long-term dependencies in time series data. Works by Fischer and Krauss [17] demonstrated the potential of LSTMs in outperforming traditional GARCH models in forecasting volatility. Comparative studies, such as those by Hansen and Lunde [18], have evaluated the effectiveness of these models against traditional GARCH-type models. While machine learning models often outperform in capturing complex market dynamics, they also pose challenges in interpretability and computational intensity. This has led to the development of hybrid models that combine the strengths of econometric models and machine learning. For instance, incorporating GARCH-based features into neural networks or SVMs can yield models that are both interpretable and powerful in forecasting. Hybrid models that combine machine learning with traditional GARCH-type models have emerged, seeking to fuse econometric rigor with machine learning's flexibility. Moreover, the literature reflects an interest in ensemble methods for volatility forecasting. These methods pool predictions

from a collection of different models, capitalizing on the strength of each. The blend of various machine learning models into an ensemble, as proposed by Ramos-Pérez et al. [19], can often yield more accurate and robust forecasts than individual models alone. One of the first study to try to combine a GARCH model with artificial neural networks for return volatility forecasting was Glen and Kamstra [20], followed by Hu and Tsoukalas [21]. Another notable study was conducted by Malliaris and Salchenberger [22] with the aim to forecast an index Implied volatility for risk management purposes using an ANN. Dunis and Huang [23] conducted an experiment somewhat similar to what we intend to do: They explored the relevancy of Recurrent neural networks and neural network regressions for forecasting and trading currency volatility (using options straddles), as well as combinations of these models with GARCH models. They find very promising results from the RNN based volatility trading strategy. A recent study by Christensen et al. [24] finds evidence of the superior predictive ability of neural networks and regression trees over HAR models, especially over long horizon. They attribute the forecast gains in the higher persistence captured by the ML models. Lu et al. [25] demonstrates the outperformance of a stacked EGARCH-ANN model over traditional GARCH modes for forecasting the Chinese energy markets log returns volatility. Similarly, Kristjanpoller and Minutolo [26] and Kristjanpoller and Minutolo [27] respectively show the superiority for GARCH enhanced ANN in forecasting GOLD and OIL prices volatility. A recent attempt to use XgBoost models for volatility forecasting was made by Teller et al. [28]. They find evidence of Xgboost models outperforming HAR models for different forecasting horizon. They also find that using non-linear specifications for the base learners perform better than non-linear ones on longer horizons forecasts. More recently, Transformers, first introduced in the seminal paper "Attention is All You Need" by Vaswani et al. [2], revolutionized the field of deep learning by eschewing the traditional recurrent structures in favor of self-attention mechanisms. In a groundbreaking study, Liu et al. [29] put forth a Transformer model leveraging mixed-frequency data to forecast stock volatility . The versatility of Transformer architectures in finance is further illustrated by their application across a spectrum of assets. As reported in research covered by Ge et al. [30] , the Temporal Fusion Transformer Lim et al. [31], a variant of the canonical Transformer, has been applied to forecast the volatility of diverse assets such as S&P500, NASDAQ100, gold, silver, and oil and outperforms RNNs, MLPs, and GARCH models for each of these assets. Closest to our study, Ramos-Pérez et al. [19] found empirical evidence for the outperformance of an extended transformer architecture called the multi-transformer in generating more accurate one-day ahead forecasts of the S&P500 log returns volatility over GARCH, LSTM, ANN and traditional transformer. They also found evidence of the outperformance of the GARCH-enhanced version of these models. Ge et al. [1] conducted a systematic review, highlighting the efficacious deployment of neural networks in financial volatility forecasting, emphasizing the nuanced ability of these models to approximate both linear and nonlinear dynamics without prior knowledge of the data-generating process. In the context of multi-horizon forecasting, literature underscores the challenges of predicting volatility over different time frames. The inherent uncertainty of the market and the impact of external economic events make multi-horizon forecasting particularly complex. Researchers have investigated models that can simultaneously provide short, medium, and long-term volatility forecasts, with mixed success.

3 Data

3.1 Data Exploration and Stylized facts

Market volatility, as a measure of risk, uncertainty or market gauge, is one of the most documented and researched subject of modern finance. It has been explored both from the forecasting angle and the measuring angle. As such, the extensive literature on the subject tends to have a lot of contradictory results. Nonetheless, a couple of stylyzed facts are universally recognized when it comes to volatility given the extensive and mathematically sound research that went into proving them. The stylized facts of financial market volatility are rich and complex, intertwining various phenomena like the leverage effect, the impact of outliers, the influence of news, forecastability issues, non-stationarity, and long memory. These factors are interrelated, each contributing to the intricate behavior of financial markets. Starting with the leverage effect, this phenomenon highlights a key asymmetry in how markets respond to information. Typically, negative news tends to have a more pronounced impact on volatility compared to positive news of a similar magnitude. This asymmetric response can be partly explained by the leverage hypothesis, which suggests that negative market movements increase the leverage of firms (debt-to-equity ratio), thereby raising the risk and volatility. The leverage effect is not just a theoretical construct but a practical consideration in volatility modeling, as it challenges the notion of symmetry in market responses and necessitates the use of models like GJR-GARCH and EGARCH that can account for this asymmetry. Closely linked to the leverage effect is the impact of news on market volatility. Financial markets are highly sensitive to new information, and the arrival of news can lead to sudden and significant changes in volatility. The relationship between news and volatility is complex and often depends on the nature of the news, market conditions, and investor sentiment. Positive news might lead to a moderate increase in volatility due to increased trading activity, whereas negative news can cause a sharp spike in volatility, reflecting panic selling or rapid changes in investor sentiment. The impact of outliers is another critical aspect of financial market volatility. Outliers, which can be caused by extraordinary events such as financial crises, geopolitical tensions, or major economic announcements, can disproportionately affect volatility. These outliers often lead to a heavy-tailed distribution of returns, meaning that extreme changes in prices are more common than would be expected in a normal distribution. This heavy-tailed characteristic of financial returns complicates risk management and forecasting, as standard models based on normal distribution assumptions may underestimate the probability and impact of extreme market movements. Forecastability of volatility is an ongoing challenge in financial econometrics. Volatility is inherently difficult to predict due to its dynamic non-stationary nature and sensitivity to a wide range of factors. Non-stationarity implies that the statistical properties of volatility, such as the mean and variance, change over time, making it difficult to model and forecast using traditional time series approaches. Some studies (Perry and R [32] and Pagan and Schwert [33]) even find some evidence for a unit root in the volatility time-series. While models like GARCH and its variants have been developed to improve volatility forecasting, the accuracy of these predictions can vary significantly depending on the model specification, time period, and market conditions. The challenge of forecasting is compounded by the non-stationarity of financial time series, meaning that the statistical properties of these series,

such as mean and variance, change over time. Non-stationarity arises from various factors, including changing macroeconomic conditions, evolving market structures, and the impact of regulatory changes. The concept of long memory in volatility refers to the persistence of shocks in financial markets over time. Long memory implies that the effects of past market movements, especially large shocks, can influence volatility for an extended period. These characteristic challenges the assumption of short memory in traditional time series models and has led to the development of long memory models like FIGARCH which captures the decaying influence of past shocks over time but at a slower rate compared to short-memory models, or component GARCH models. Long memory is a critical concept for understanding the temporal dependencies in financial markets and for developing effective risk management strategies. Contrasting with long memory is the concept of mean reversion, which posits that despite short-term fluctuations, volatility tends to revert back to a long-term average over time. This phenomenon indicates that high or low periods of market volatility are temporary and will eventually move back towards a historical average. Mean reversion is a key consideration in risk management and financial modeling, as it implies that extreme market conditions are not permanent and that forecasts should account for this reversion to the mean over the long term. Volatility clustering, another critical aspect, refers to the tendency of high-volatility periods to be followed by high-volatility periods and low-volatility periods to be followed by low-volatility periods. This phenomenon, observable in financial time series, suggests that volatility exhibits a serial correlation – large changes in prices are often clustered together, followed by periods where prices change minimally. This clustering effect is a crucial factor in volatility forecasting, as it necessitates models that can account for these changing regimes in market conditions. These concepts are deeply interrelated. Long memory in volatility suggests a persistence of the effects of market shocks, implying that past market events can influence future volatility for a prolonged period. However, the presence of mean reversion within this framework indicates that while these effects are persistent, they don't last indefinitely, with volatility eventually returning to its long-term average. Meanwhile, volatility clustering reflects how these phenomena can manifest in actual market behavior, with periods of persistent high or low volatility, followed by eventual reversion to mean levels. The interplay of these factors complicates the task of volatility modeling and forecasting. Traditional models like GARCH have been augmented with features to capture long memory and mean reversion. For instance, models like FIGARCH or EGARCH incorporate elements to account for the long-lasting effects of past volatility and the asymmetric impact of market shocks, respectively. Yet, capturing these dynamics accurately remains a challenge due to the inherent complexity and evolving nature of financial markets. In essence, financial market volatility landscape is marked by a asymmetry, exogeneous factors, and a delicate balance between persistence and mean reversion, with volatility clustering providing observable evidence of these dynamics, making it a challenging yet fascinating subject for study in financial economics.

This study focuses on forecasting the volatility of 5 assets that have been picked for specific reasons, because they have different degrees of sensitivity to each of the aforementioned stylized facts. We attempt to forecast the volatility of the S&P500, the NASDAQ100, the Russel 2000, GOLD prices and OIL prices. As one of the most mainstream and followed gauge for the health of the American stock market, the S&P500 have historically been impacted by Market sentiment, geopolitical events, and macroeconomic data. It has historically

demonstrated high cyclicality, due to its inclusion of the biggest stock in some cyclical US industries. As per the NASDAQ, it has historically proven much more volatile and fat-tailed than the S&P500, due to to the highly uncertain nature of technology businesses, the highly innovation-driven industry, the constant evolution of business condition, and the relatively low barriers to entry. Some of these companies, although very successful are often in early stages of their growth and have not yet established a track record of profitability. Additionally, technology companies are often more reliant on a small number of products or services, which can make them more susceptible to market fluctuations. Finally, the technology industry is subject to rapid changes, which can lead to increased uncertainty and volatility in stock prices. Similarly, the RUSSEL 2000 has historically exhibited more volatility due to the leverage effect and its longer memory, mainly due to the high sensitivity of smaller companies to market and economic conditions, especially to interest rates. It can be seen by remarking how smaller companies perform worse than average in recession or high interest rate environment, and above average in an expanding economy. They have more debt on their balance sheet, and can therefore better leverage a favorable market environment, but are less resilient to economic shocks. With regard to oil prices, it is a well documented fact that they exhibit extreme volatility clustering or severe fat tails around major supply disruption (Kang et al. [34]), as well as slow mean reversion and very quick reaction to random shocks. On top of supply and demands shifts, oil prices are very sensitive to geopolitical events ,instability, weather,natural disaster, as well as any minor disruption in the highly fragmented chain of production. Gold, as an asset, has an interesting and long history of being a reserve of value, of historically being the prime vehicle of wealth protection for being inflation proof, and the value reserve for many central bank backed-currencies. Additionally, just like oil prices, Gold prices are driven by supply and demands dynamics, as well as geopolitical shocks, interest rates, but most importantly, inflation and investor behavior. Gold prices are very prone to behavioral drivers like herding or panic, and also display a strong asymmetry because negative shocks tend to increase volatility more than positive shocks. What makes the attempted exercise of forecasting the volatility of these assets is the different degrees in which we find the documented stylized facts in each of them. We can see a display of some descriptive statistics of our returns and volatility series in the table below, illustrated on Figure 2 and Figure 1. The descriptive statistics table also shows us the p-values from the ADF and Jarque-Berra tests. On the considered sample window for all 5 assets, both tests return very small p-values, which is in line with what has been documented so far in most of the literature. This means that for each asset studied here, at a threshold of 1%, the realized volatility series are non gaussian and non-stationary (and might contain a unit root), as can be visualized on Figure 2. Except for Gold, as per the left panel of Figure 3, the volatility series exhibit strong auto-correlations at shorter lags (1 to 5), as well as moderate to strong auto-correlation until approximately lag 25 to 30. All the volatility series also exhibit strong partial auto-correlation for short lags, but mostly nonexistent partial auto-correlation after lag 10, as per the right panel of Figure 3 .

3.2 Realized Volatility

As demonstrated by Barndorff-Nielsen and Shephard [35], realized volatility is a universally accepted and used proxy for the quadratic variation of a semi-martingale (in this case our

Series	Statistic	S&P 500	NASDAQ	RUSSEL 2000	GOLD	OIL
Volatility	count	4532	4532	4532	4262	3961
	mean	0.007864	0.008562	0.012046	0.007272	0.014699
	std	0.008819	0.006121	0.015333	0.007962	0.013223
	min	0.000980	0.000854	0.001909	0.000233	0.001608
	25%	0.004239	0.005123	0.006708	0.004320	0.009599
	50%	0.005785	0.007052	0.008775	0.005799	0.012736
	75%	0.008755	0.010048	0.012711	0.008107	0.016869
	max	0.298011	0.102214	0.655744	0.300187	0.676301
	median	0.005785	0.007052	0.008775	0.005799	0.012736
	range	0.297031	0.101359	0.653834	0.299954	0.674693
	skew	12.467591	4.521276	20.544564	19.106197	32.352277
	kurtosis	308.086588	38.135491	743.945957	601.578134	1585.863597
	Range	1990 - 2023	1990 - 2023	1990 - 2023	2000 - 2023	2000 - 2023
	ADF test	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
	JB test	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
Returns	mean	0.000317	0.000455	0.000305	0.000333	0.000289
	std	0.011989	0.013588	0.015370	0.011146	0.026335
	min	-0.127652	-0.131492	-0.153991	-0.098206	-0.282206
	25%	-0.004212	-0.005449	-0.007041	-0.004915	-0.012533
	50%	0.000704	0.000988	0.000883	0.000468	0.001185
	75%	0.005693	0.007269	0.008397	0.006280	0.013320
	max	0.109572	0.111594	0.089763	0.086432	0.319634
	median	0.000704	0.000988	0.000883	0.000468	0.001185
	range	0.237224	0.243086	0.243754	0.184637	0.601840
	skew	-0.511164	-0.402677	-0.570084	-0.339674	0.054205
	kurtosis	12.893379	7.423571	7.819972	5.213697	17.578105

Here we see the the relevant descriptive statistics of the two kinds of series used in this study : Realized volatility and returns

return process). Assuming that our price process is described by the simple diffusion process:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

where:

- S_t is the stock price at time t,
- μ is the drift coefficient (representing the rate of return),
- σ is the diffusion (volatility) coefficient,
- W_t is a Wiener process (or Brownian motion).

The quadratic variation of this stochastic process , is a measure of the cumulative variance of the process over a time interval. For a GBM S_t represented by the stochastic differential equation $dS_t = \mu S_t dt + \sigma S_t dW_t$, the quadratic variation over an interval $[0, T]$ is given by:

$$[S]_T = \lim_{\max \Delta t_i \to 0} \sum_i (S_{t_{i+1}} - S_{t_i})^2$$

if the limit exists.This represents the cumulative variance of the stock price process over the time interval $[0, T]$, In a series of papers, Barndorff-Nielsen and Shephard propose using the realized volatility as a proxy for the quadratic variation of a semi-martingale. Let us denote the intraday return as:

$$R^n_{t+i\tau} = \log(S^n_{t+i\tau}) - \log(S^n_{t+(i-1)\tau})$$

where S_t is the stock price at time t, $\tau = \frac{1}{252}$ is the length of a business day in years, and n is the number of periods per day.
The realized variance $RV^{(n)}_{t,t+\tau}$ is defined as:

$$RV^{(n)}_{t,t+\tau} = \sum_{i=1}^{n} \left(R^n_{t+i\tau}\right)^2$$

The daily quadratic variation satisfies:

$$QV_{t,t+\tau} = \lim_{n\to\infty} \sum_{i=1}^{n} \left(\log(S^n_{t+i\tau}) - \log(S^n_{t+(i-1)\tau})\right)^2 = \lim_{n\to\infty} \sum_{i=1}^{n} \left(R^n_{t+i\tau}\right)^2$$

Hence, as n tends to infinity:

$$QV_{t,t+\tau} \approx RV^{(n)}_{t,t+\tau}$$

For a given $n \in \mathbb{N}$, the realized volatility is a proxy for the daily quadratic variation.
Using realized volatility to model variance grants us the ability to bypass reliance on theoretical models by directly utilizing a variable that can be observed. This approach presents a clear benefit; however, challenges arise when it comes to the actual application of this method. Research indicates that micro-structure noise, which stems from various factors like bid-ask spreads and the discrete nature of price formation, exists within financial markets. This noise, as discussed in works by AMIHUD and MENDELSON [36], HARRIS and RAVIV [37], and Madhavan [38], has the potential to skew the accurate estimation of realized volatility.

3.3 Collection, construction and preprossessing

The data used in this study is extracted from the Trades and Quotes (TAQ) database. Since our assets of interest were not directly observable on the market, we used the following ETFs as proxies:

Asset	Proxy's ticker	Proxy's name
S&P500	SPY	SPDR S&P 500 ETF
NASDAQ	IYW	iShares US Technology ETF
RUSSEL 2000	IWM	iShares Russell 2000 ETF
GOLD	IAU	iShares Gold Trust
OIL	USO	United States Oil ETF

For each ETF, we extracted a series of millisecond intraday trade prices between December 2005 and December 2021. The raw data consisted, for each ETF, of nation-wide aggregated trades data at each millisecond of the day. Following the findings of Ait-Sahalia et al. [39], Bandi et al. [40] and Bandi and Russell [41] proposing that it is reasonably safe to treat five-minutes sampled returned as noise free on most liquid assets in the recent years, we chose to extract prices at five-minute intervals, from which intraday returns are computed, which are then summed as per the formula above to obtain our daily realized volatility series.

Data cleaning and standardization are crucial steps in preparing time series data for forecasting with machine learning. These steps ensure that the data is in a suitable format and quality for the algorithms to process and learn from. The process typically involves handling missing values, removing outliers, and transforming the data to a common scale. Time series data often has missing values due to various reasons like errors in data collection or transmission.With our raw data being market closing prices, the data was already free of week-ends and holidays missing data usually encountered in similar exercises , therefore did not need much cleaning. For the rare occurrences where some trading days data point were missing, we chose to handle it with linear interpolation. This method involves estimating missing values in a linear fashion from neighboring data points. If y_t is a missing value at time t, and y_{t-1} and y_{t+1} are known, linear interpolation is used as follow :

$$y_t = \frac{y_{t-1} + y_{t+1}}{2}$$

Furthermore, machine learning algorithms often require data to be on a similar scale, typically achieved through normalization or standardization. To effectively improve the training of our models, we chose to re-scale each volatility timeseries using a z-score normalization often referred to as standardization. This method transforms the data to have zero mean and a standard deviation of one. For a data point x_t, the standardized value z_t is:

$$z_t = \frac{x_t - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation of the series.

4 Forecasting models

We present every [**A-M-H**] model final architecture in the results section.
We use the following nomenclature throughout the study to refer to our models:

- [**A-M-H**] : Model M for H steps ahead forecasts on asset A
- [**M-H**] : Group of models M for H steps ahead forecasts
- [**A-M**] : Group of models M applied on Asset A ,

With

- **A** ∈ [S&P500, NASDAQ 100, RUSSEL 2000, OIL, GOLD]
- **M** ∈ [CGARCH, HAR, XgBoost, NgBoost, LSTM, Transformer, XG CGARCH, NG CGARCH, LSTM CGARCH, TRANS CGARCH, X-N-L-T, X-N-L-T CGARCH]
- **H** ∈ $[5, 10, 15, 20, 60]$

4.1 Benchmarks

4.1.1 HAR

The Heterogeneous Autoregressive (HAR) model, introduced by Corsi [13] primarily for modeling and forecasting financial volatility, stands out for its ability to encapsulate different components of volatility operating at varying time scales. This approach is particularly adept at capturing the multi-scale nature of financial market volatility, making it highly suitable for multi-day ahead forecasting The HAR model is built upon the concept that volatility can be influenced by factors operating at different time horizons. It typically includes daily, weekly, and monthly volatility components, allowing it to capture the varying effects of information arriving at different frequencies. The purpose of the HAR model is to account for the long memory and persistent nature of volatility. It integrates different time scales (daily, weekly, monthly) into the volatility forecasting process, reflecting the varying investment horizons and information-processing speeds of market participants. This multi-scale approach allows the HAR model to capture both short-term fluctuations and long-term trends in volatility, making it particularly useful for forecasting over multiple horizons. The mathematical representation of the HAR model for daily data is typically as follows:

$$RV_{t+1|t} = \alpha + \beta_d RV_t + \beta_w RV_t^{(w)} + \beta_m RV_t^{(m)} + \epsilon_{t+1}$$

Here, $RV_{t+1|t}$ is the forecasted realized volatility at time $t+1$, given information up to time t. The model includes three key terms:

- RV_t : The daily realized volatility.

- $RV_t^{(w)}$: The weekly realized volatility, often calculated as the average of daily volatilities over the past week.
- $RV_t^{(m)}$: The monthly realized volatility, typically the average of daily volatilities over the past month.

The parameters α, β_d, β_w, and β_m are estimated using regression techniques, and ϵ_{t+1} is the error term.
The HAR model assumes that variance is a continuous, integrable function over the chosen period (daily, weekly, monthly), and the aggregated volatilities are representative of different market participant horizons. For the HAR model to provide reliable forecasts, the time series data must satisfy certain statistical properties. Ideally the data should be ergodic, ensuring that time averages converge to ensemble averages, which is vital for the consistency of the estimated parameters.The model assumes a linear relationship between past realized volatility and future volatility. The error term ϵ_{t+1} is generally assumed to be normally distributed with mean zero and constant variance. The coefficients β_d, β_w, and β_m are estimated using ordinary least squares regression.
For multi-day ahead forecasting, the HAR model leverages its ability to incorporate information from various time horizons. The forecasting process can be iteratively applied to predict future volatilities. For instance, a k-day ahead forecast is computed as follows:

- **Initial Forecast** : Use the HAR model to forecast the next day's volatility RV_{t+1}.
- **Rolling Forecasts** : Update the input variables (daily, weekly, and monthly volatilities) by rolling forward one day and recalculating these variables. Then forecast the next day's volatility using the updated inputs.
- **Iteration** : Repeat the rolling forecast process for the number of days ahead required RV_{t+k}

4.1.2 Component GARCH

The Component GARCH model represents a significant evolution in volatility modeling, offering a nuanced understanding of the dynamics of financial market volatility, particularly suited to capture the nuanced behavior of volatility. As detailed by Gary and Engle [3], the model decomposes the conditional variance of asset returns into two distinct elements: a permanent component q_t that captures the long-term variance, and a transitory component that captures short-term fluctuations. The permanent component is envisioned as a highly persistent process, potentially following a random walk, which is indicative of its stability and influence over extended periods. This is mathematically characterized by an autoregressive parameter ρ close to one, as empirically evidenced in their paper for the S&P 500 and NIKKEI indices, suggesting the presence of a unit root and hence a non-reverting process that embeds the impact of new information over time.
The original GARCH(1,1) process as per Bollerslev [8] is expressed as:

$$r_t = m_t + \epsilon_t$$

$$h_t = \omega + \alpha\epsilon_{t-1}^2 + \beta h_{t-1}$$

Mathematically, the Component GARCH model modifies the GARCH(1,1) representation of h_t and can be expressed with the following equations from the Gary and Engle [3] paper: The conditional variance equation:

$$h_t = q_t + \alpha(\epsilon_{t-1}^2 - q_{t-1}) + \beta(h_{t-1} - q_{t-1})$$

The equation for the permanent component:

$$q_t = \omega + \rho q_{t-1} + \phi(\epsilon_{t-1}^2 - h_{t-1})$$

where:

- h_t is the conditional variance at time t,
- q_t is the permanent component of the variance at time t,
- ϵ_{t-1} is the innovation or shock from the previous time period,
- α and β are parameters capturing the responsiveness of the conditional variance to the innovation and the lagged variance, respectively,
- ω, ϕ, and ρ are parameters specific to the permanent component, with ρ close to one indicating the non-reverting characteristic of q_t.

The transitory component, represented by $\alpha(\epsilon_{t-1}^2 - q_{t-1}) + \beta(h_{t-1} - q_{t-1})$, captures the short-term fluctuations, typically reverting to the long-term trend.
For the Component GARCH model to be meaningful and for its forecasts to be reliable, certain stationarity and convergence conditions must be met. The stationarity condition for the Component GARCH model can be formulated as:

$$(\alpha + \beta)(1 - \rho) + \rho < 1$$

This condition implies that both the permanent component (captured by ρ) and the transitory component (captured by $\alpha + \beta$) must be covariance stationary
The decomposition in the Component GARCH model implies that in the long run, there is no difference between the conditional variance and its trend. As the forecasting horizon extends, the forecast of the conditional variance converges to the current expectation of the trend plus a constant drift:

$$h_{t+k} = k\omega + q_t \text{ as } k \to \infty$$

This property is crucial for multi-step forecasting, as it suggests that in the long term, the conditional variance is influenced primarily by the permanent component of volatility.
A key requirement of any volatility model is that the volatility estimate should remain non-negative over time. This is crucial for the model's practical applicability and interpretability. The Component GARCH model maintains this property under certain parameter constraints. Specifically, the parameters must satisfy certain inequality constraints to

ensure non-negative conditional variances. Engle and Lee's empirical analysis using daily stock indices demonstrated the model's efficacy in capturing both the short-term and long-term dynamics of market volatility. The model's parameters were found to be significant, indicating the presence and importance of both permanent and transitory components in the conditional variance. This empirical validation underscores the model's utility in practical applications, particularly in financial risk management and strategic investment planning To produce multi-day ahead volatility forecasts using the Component GARCH model, we first need to understand the underlying mathematical structure of the model and then proceed to the forecasting methodology. This explanation will be detailed, focusing heavily on the mathematical equations and their implications.
For forecasting volatility multiple days ahead, we focus on predicting h_{t+k} for $k > 0$, where k is the number of days ahead of the forecast. The forecasting involves the following steps:

- For the permanent component q_t, the forecast q_{t+k} is given by:

  $$q_{t+k} = [(1 - \rho^k)/(1 - \rho)]\omega + \rho^k q_t$$

  where ρ is the autoregressive root introduced in the trend equation.
  If $\rho = 1$, which indicates a unit root, the equation simplifies to a linear trend:

  $$q_{t+k} = k\omega + q_t$$

- The forecast of h_{t+k} involves predicting the difference between the conditional variance and its trend:

  $$h_{t+k} - q_{t+k} = (\alpha + \beta)^k (h_t - q_t)$$

  As $k \to \infty$, this difference converges to zero, indicating that in the long run, the conditional variance aligns with its trend.

- The final forecast for the conditional variance h_{t+k} is obtained by combining the forecasts of q_{t+k} and $h_{t+k} - q_{t+k}$:

  $$h_{t+k} = q_{t+k} + (\alpha + \beta)^k (h_t - q_t)$$

  For large k, as $(\alpha + \beta)^k$ approaches zero, the forecast converges to the permanent component:

  $$h_{t+k} \approx q_{t+k}$$

4.2 Single models

4.2.1 Transformer

The transformer model, introduced by Vaswani et al. [2] in 2017, marked a departure from recurrent neural network architectures. Its key innovation, the self-attention mechanism, allows it to process input sequences in parallel, leading to significant improvements in training efficiency and model performance. Unlike RNNs, which process sequences iteratively, Transformers use self-attention mechanisms to weigh the importance of different parts of

the input data, processing the entire sequence simultaneously and allowing for parallel computation.The architecture comprises an encoder and a decoder, each consisting of multiple layers. The encoder maps an input sequence of symbol representations (words, subwords, or bytes) into a continuous representation, and the decoder generates an output sequence. The transformer is not the first neural network architecture to exploit an encoder-decoder scheme (Generative Adversarial networks, Autocencoders and others made successful use of this framework), it is the first to implement multiple self-attention units in both the encoder and the decoder. The key components of the transformer are : Key components of the Transformer include:

- **Embedding Layers**: Converts tokens into vectors of fixed dimensionality.
- **Positional Encoding**: Adds information about the position of each token in the sequence.
- **Self-Attention**: Allows the model to consider other tokens in the input sequence when encoding a token.
- **Multi-Head Attention** : Extends self-attention by running it through multiple attention "heads", capturing different aspects of token relationships.
- **Layer Normalization and Feed-Forward Networks** : Each sub-layer (self-attention, feed-forward) in the Transformer has a normalization step and is followed by a feed-forward network. The Transformer's ability to handle sequences in parallel provides a significant performance advantage over sequential models, leading to its adoption in various state-of-the-art domains.
- **Input embedding and Positional encoding** : As mentioned earlier, Transformer layers lack any recurrence mechanism. Consequently, it becomes essential to incorporate information regarding the relative positioning of observations in the time series into the model. This is achieved by augmenting the input data with positional encoding. In our study, we build on the work of Ramos-Pérez et al. [19] by using a wave function as a positional encoder.

First, The transformer converts input tokens into vectors of fixed dimensionality,

$$\mathbf{E}_i = \text{Embed}(x_i)$$

Where:

- x_i is the i-th element of the input sequence.
- $\mathbf{E}_i$ is its embedding representation.

Then, the Positional Encoding is computed :

$$\mathbf{P}_i = \cos\left(\pi \frac{pos}{N_{\text{pos}} - 1}\right) = \sin\left(\frac{\pi}{2} + \pi \frac{pos}{N_{\text{pos}} - 1}\right)$$

where pos = (0, 1, . . . , Npos 1) is the position of the observation within the time series and Npos maximum lag
The final input representation for each item in the sequence is:

$$\mathbf{Z}_0 = \mathbf{E} + \mathbf{P}$$

Where $\mathbf{Z}_0$ represents the input to the first encoder layer.

Self-attention and Multi-head attention The self-attention mechanism (See Figure 5)in transformers is a pivotal component ant the main innovation driving the success of this architecture across domains. It allows the model to weigh the significance of different parts of the input differently. Also known as intra-attention,it is a mechanism that equips a model to weigh the importance of different parts of the input data. It is a form of attention mechanism that computes the representation of a sequence by relating all positions of the sequence to each other. Unlike previous attention mechanisms that focused on aligning two different sequences (e.g., in sequence-to-sequence models), self-attention focuses on deriving relationships within a single sequence.The fundamental idea is to compute a representation of a sequence by relating different positions of the same sequence. For example, in a sentence, the meaning of a particular word might depend on the context provided by other words in the sentence. Self-attention provides a way for each word to consider the entire sentence when establishing its context. It involves three key components: Queries (Q), Keys (K), and Values (V). These components are derived from the input data but represent different aspects of that data:

- **Queries (Q)**: These are representations of the input used to score how much focus each element of the input sequence should have with respect to other elements. It can be thought of as a request issued by an element to decide which other elements to focus on

- **Keys (K)**: Keys are used in tandem with queries to compute attention scores. Each key is paired with an input element and is used to determine the amount of attention an element of the sequence should get.

- **Values (V)**: Values are the actual representations of the input elements. Once the attention scores are computed using queries and keys, these scores are used to weigh the values, resulting in a weighted sum that represents the output of the self-attention layer for each element

These vectors are not handcrafted; they are learned during the training process. Each query, key, and value is generated by multiplying the input embeddings by their respective weight matrices, which are parameters learned by the model. Let's consider an input sequence represented by a matrix X, where each row of X corresponds to an element in the sequence (like a past realization of realized volatility) :

First, we perform linear transformations on the input matrix X to obtain the query, key, and value matrices. This is done using trainable weight matrices W^Q, W^K, and W^V:

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$

Here, Q, K, and V are the resulting query, key, and value matrices, respectively.
Next, we compute Attention scores. These scores determine how much focus should be put on other parts of the input sequence when encoding a particular element. The attention score between a query and a key is usually computed using the dot product, although other methods like cosine similarity can also be used.
The raw attention scores indicate the compatibility between queries and keys and are computed as follows:

$$\text{Score} = QK^T$$

Since the magnitude of the dot product grows with the dimensionality of the input, which could lead to very large values and, in turn, push the softmax function into regions where it has extremely small gradients, it is common practice to scale the scores by the square root of the dimensionality of the keys (d_k) :

$$\text{Score} = \frac{QK^T}{\sqrt{d_k}}$$

To turn the scores into probabilities, the softmax function is applied to each row of the scaled attention scores. This step ensures that the weights sum up to 1, thus forming a proper probability distribution.

$$\text{Attention Weights} = \text{softmax}(\text{Score})$$

Finally, the attention weights are used to create a weighted sum of the values. The result is a matrix where each row represents the output of the self-attention layer for each element of the sequence:

$$\text{Output} = \text{Attention Weights} \times V$$

The final output to a self-attention unit is summarized as :

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-head Self Attention In practice, self-attention is often implemented through a mechanism known as multi-head attention (Figure 6). This involves running multiple self-attention operations in parallel, each with its own set of linear transformations (i.e., its own W^Q, W^K, W^V). The outputs of these multiple attention heads are then concatenated and linearly transformed again to produce the final output. This allows the model to jointly attend to information from different representation subspaces, capturing various aspects of the input sequence.
If h is the number of heads, the multi-head attention
can be represented as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O$$

where each head is defined as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

and W^O is the weight matrix for the linear transformation applied after concatenation.
One of the key advantages of self-attention is its ability to handle long-range dependencies with ease. Unlike RNNs, which process sequences step by step, self-attention processes an entire sequence at once, allowing it to consider the full context in one step. This leads to significantly improved training times since parallelization becomes feasible.
Moreover, self-attention provides an interpretability aspect to the neural network's decision-making process, as the attention weights can be analyzed to understand which parts of the input sequence are deemed important when processing a particular element.
Feed-Forward Networks In most versions of the transformer architecture, The output of the attention mechanism is passed through a feed-forward network, which is applied to each position separately and identically. This can be represented as:

$$\text{FFN}(x) = \text{ReLU}\left(xW_1 + b_1\right)W_2 + b_2$$

In this study, as per Ramos-Pérez et al. [19], we only use the encoder part of the transformer's architecture, represented by the left half part of image (A) on Figure 7.

4.2.2 Long-Short Term Memory Network (LSTM)

Long Short-Term Memory (LSTM) network, introduced in 1997 by Hochreiter and Schmidhuber [42], represent a significant advancement in the field of deep learning, particularly for tasks involving sequential data such as time series analysis and natural language processing. Their unique architecture allows them to remember long-term dependencies, addressing the vanishing gradient problem common in traditional recurrent neural networks (RNNs). They were introduced by Hochreiter and Schmidhuber in 1997 to overcome the limitations of standard RNNs. The key innovation in LSTM is its ability to maintain a long-term state or memory, which is adjusted through structures called gates.

An LSTM unit consists of a cell (which carries the state across sequence steps, see Figure 8) and three types of gates that regulate the flow of information: the input gate, the forget gate, and the output gate. These gates control the extent to which new input should be incorporated into the memory, the degree to which the existing memory should be forgotten, and how much of the memory should contribute to the output, respectively.
The central concept in LSTM is the cell state, denoted as C_t, which runs straight down the entire chain of the network. It has the ability to carry relevant information throughout the processing of the sequence. Because of the gated mechanism, information can be added or removed from the cell state.

- **Forget Gate (f_t)**: This gate decides what information should be discarded from the cell state. It looks at h_{t-1} (the previous hidden state) and x_t (the current input) and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents "completely keep this" while a 0 represents "completely get rid of this".
- **Input Gate (i_t) and Candidate Cell State ($\tilde{C}_t$)**: The input gate decides which values to update, and a candidate layer creates a vector of new candidate values that could be added to the state.
- **Output Gate (o_t)**: The output gate decides what the next hidden state h_t should be. The hidden state contains information about previous inputs and is used for predictions.

The operations within an LSTM unit can be formulated as follows:

- **Forget Gate**:
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
- **Input Gate**:
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
- **Cell State Update**:
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
- **Output Gate and Hidden State**:
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

Where:

- σ represents the sigmoid function.
- W **and** b terms denote weight matrices and bias vectors, respectively.

- $[h_{t-1}, x_t]$ denotes the concatenation of the previous hidden state and the current input.

Training LSTM networks involves backpropagation through time (BPTT), a process similar to back-propagation in other neural networks, but with the unrolled sequence of states. It requires careful handling of gradients, as they can grow or decay exponentially over long sequences (the problem LSTM was designed to address).

4.2.3 Extreme Gradient Boosted tree (XgBoost)

Weak Learners Boosting starts with a base learning algorithm to build weak learners. A weak learner is a model that is only slightly correlated with the true classification. It's often a simple model (like a decision tree or a linear regressor), but it could be any machine learning algorithm that provides better-than-random predictions.

Decision trees A decision tree is a widely used non-parametric machine learning algorithm that can be used for both classification and regression tasks. In the context of regression, a decision tree predicts the value of a target variable by learning simple decision rules inferred from the feature data.
The model is structured like an inverted tree with a series of binary splits or decisions. Each internal node of the tree represents a decision on a certain feature, each branch represents the outcome of the decision, and each leaf node represents a final output value (the predicted value for regression tasks). A decision tree splits the data starting from the root node and makes decisions that lead to leaf nodes, where the final continuous output is given.
The entire dataset starts at the root of the tree. The algorithm selects a feature and a split point on that feature to divide the data into two subsets.
The choice of which feature to split on and where to split it is typically based on minimizing a cost function like the Mean Squared Error (MSE). The algorithm considers every feature and every possible value of that feature to determine the best split.
This process is recursively applied to each derived subset. The tree continues to grow until a stopping criterion is met (like a maximum depth of the tree, a minimum number of samples required to split a node, etc.).
For making a prediction, a new data point is passed down the tree. At each node, a decision is made based on the value of the corresponding feature until a leaf node is reached. The value at the leaf node gives the predicted output.

Regression trees As a more specialized decision tree, regression trees (Figure 9), a pivotal element in the realm of machine learning, stand out for their efficacy in managing regression problems through a structured, tree-based approach. The origins and evolution of regression trees are intertwined with the development of decision trees, a journey marked by significant contributions like ID3 by Quinlan [43] CART (Classification And Regression Trees) by Breiman et al. [44], and C4.5 by Quinlan in 1993. These foundational algorithms set the stage for the creation of regression trees, extending the capabilities of decision trees to not just handle categorical outcomes, as in classification problems, but also continuous outputs typical in regression scenarios. The inception of the CART algorithm, in particular, is a cornerstone in this narrative. It introduced a methodology that was capable of dealing

with both classification and regression tasks, making it a versatile tool in predictive modeling.

Breiman et al. [44] CART algorithm forms the conceptual basis for regression trees. It operates by systematically dividing the predictor space into distinct segments, forming a binary tree structure. Just like a general decision tree, this structure is characterized by its internal nodes, which correspond to the input features, and branches, which represent the decisions or splits made based on these features. The culmination of these branches is the terminal nodes or leaves, which hold the predicted values or leaf weights. These leaf weights are essentially the outcomes of the regression tree for the various partitions of the input space. As explained in Teller et al. [28], for a given input element x_i in a data set S of size n, the regression tree delineates the input space into K distinct regions $R_1, ..., R_K$. The model assigns a predicted value $\hat{y}_i$ to each input element, calculated as the sum of the estimated leaf weights $\hat{w}_k$ for the region to which x_i belongs, expressed through the equation:

$$\hat{y}_i = \sum_{k=1}^{K} \hat{w}_k I(x_i \in R_k)$$

Here, I represents the indicator function, determining whether the input element falls within a specific region R_k. The estimation of leaf weights $\hat{w}_k$ for each region is achieved by minimizing a loss function, typically a squared error loss, across the data points within that region.

The process of determining the best way to split the input space and form these regions is a critical aspect of the CART algorithm. It employs a method known as recursive binary splitting, combined with least squares optimization, to minimize the residual sum of squares (RSS). This optimization is pivotal in the context of regression trees as it guides the algorithm in evaluating all possible combinations of input variables and their respective threshold split candidates. The RSS is calculated by the equation:

$$\text{RSS} = \sum_{i=1}^{n} \sum_{k=1}^{K} (y_i - \hat{w}_k)^2 I(x_i \in R_k)$$

This equation reflects the greedy nature of regression trees, as the algorithm at each step of tree growth makes locally optimal choices aiming to find the most effective global solution. However, a notable challenge with regression trees is their propensity to over-fit the training data. This over-fitting arises when the algorithm creates overly complex trees that perfectly fit the training data specifics but fail to generalize well to new, unseen data. To mitigate this, the implementation of regression trees incorporates strategies like stopping criteria and tree pruning. Stopping criteria involve setting limits on the growth of the tree, such as imposing a maximum depth or specifying a minimum number of instances per node. Tree pruning, on the other hand, involves the systematic removal of parts of the tree that contribute little to its predictive power. This pruning can be undertaken in either a top-down or bottom-up fashion and is crucial in simplifying the tree to prevent over-fitting.

In the broader context of machine learning, ensemble models have built upon and enhanced the basic structure of regression trees. Ensemble methods, recognized for their superior performance over single-algorithm approaches, combine multiple weak learners, often decision

trees, to create a more robust model. As stated above, Boosting, a key ensemble technique, iteratively adds new learners to previously fitted ones, focusing on improving predictions, especially for data points that earlier models did not accurately predict.

Boosting and Extreme Gradient Boosting Boosting is a machine learning ensemble technique that aims to create a strong classifier from a number of weak classifiers.With boosting, base learners are estimated sequentially, and each base learner aims to reduce the error of its predecessors. The first base learner is trained on the entire dataset and its predictions are used to evaluate its performance, typically using a loss function like mean squared error for regression or log loss for classification. In each subsequent iteration, the boosting algorithm adjusts the weight of training instances based on the performance of the previous model. Instances that were misclassified or had higher errors are given more weight, making the algorithm focus more on these in the next iteration.A new model is then trained on the reweighted data. The goal is for this new model to perform better on the instances that the previous model got wrong. The predictions from all models are then combined to make a final prediction. This is typically done through a weighted vote or average, where more accurate models have a bigger influence on the final prediction. As the process continues, the boosting algorithm iteratively improves the accuracy of the overall model.The process is repeated, and new models are added until either a preset number of models are added or no further improvements can be made. The ensemble of weak learners evolves into a single strong learner. To avoid overfitting, boosting algorithms usually include regularization techniques, such as limiting the number of weak learners or shrinking the contribution of each through a learning rate

Extreme Gradient Boosting (XGBoost) , recently intoduced by Chen and Guestrin [45], is an advanced implementation of gradient boosting algorithm. It has gained popularity due to its efficiency and effectiveness in solving various classification and regression problems. In the most common implementation of XgBoost as well as for our study, the base learners are regression trees. Each tree is a set of binary rules (splits) based on feature values. The trees are grown greedily, selecting the splits that maximize the gain until a specified maximum depth is reached.

Given a dataset with n examples and m features $\{(x_1, y_1), ..., (x_n, y_n)\}$, a tree ensemble model uses K additive functions to predict the output:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i),\ f_k \in F$$

where F is the space of regression trees, $\{f_k\}$ are the individual trees, and K is the number of trees.

XGBoost optimizes a regularized learning objective that combines a convex loss function $l(y_i, \hat{y}_i)$ (like squared loss for regression or logistic loss for classification) with a regularization term:

$$\text{Obj}(\Theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

The regularization term $\Omega(f)$ penalizes the complexity of the model. It is defined as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

where

- T is the number of leaves in the tree
- w is the vector of scores on the leaves
- γ is the complexity control for the trees
- λ is the L2 regularization term on the leaf weights.

XGBoost uses a gradient descent algorithm to minimize the loss function. The model is updated using the gradients of the loss function with respect to the model's predictions.
For a given loss function $l(y, \hat{y})$, XGBoost calculates the first and second-order gradients (gradient and Hessian) for each instance. These are used to update the model in the direction that reduces the loss. In each iteration, XGBoost finds the best split by choosing the feature and split point that yield the highest gain.After growing a tree, XGBoost prunes it using the concept of 'depth-first' approach, where splits that provide negative gain are removed. XGBoost introduces a shrinkage factor to slow down the learning in each step, adding robustness to the model. It also uses column (feature) sampling to prevent overfitting, similar to Random Forests. XGBoost effectively combines decision trees, gradient boosting, and regularization, making it a powerful tool for predictive modeling in various machine learning tasks.

4.2.4 Natural Gradient Boosted tree (NGBoost)

Natural Gradient Boosting, proposed in 2020 in Duan et al. [46], is an ensemble model for probabilistic prediction via gradient boosting. The main innovation of this model is to generalize gradient boosting to probabilistic regression by treating the output variable conditional distribution's parameters as targets for a boosting algorithm. It therefore comes inherently with a straightforward way to compute confidence intervals. Its attractiveness relies in its great flexibility, in the sense that it can be used with any base (weak) learner,any family of parametric probability distributions, and any scoring criteria. Knowing the dynamic, jump-displaying non gaussian nature of most assets'realized volatility, the NgBoost model appeared to be an interesting candidate for modeling this variable. This study is the first in the literature to attempt to use NgBoost to volatility forecasting.
The three key elements of the NGBoost algorithm are : The base learner, The parametric probability distribution, the scoring rule.
Base Learners: NGBoost can use any regression algorithm as its base learner. Common choices include decision trees and linear models.

Parametric Probability Distributions: The method assumes a parametric form for the conditional probability distribution $P(y|x)$. It estimates the parameters of this distribution as functions of the input features x.

Proper Scoring Rules: NGBoost uses scoring rules to compare the estimated probability distribution to the observed data. The scoring rule must be proper, meaning it assigns the best score to the true distribution. Common scoring rules include the Logarithmic Score for Maximum Likelihood Estimation (MLE) and the Continuous Ranked Probability Score (CRPS), often considered a more robust alternative to MLE.

The key innovation in NGBoost is the use of the natural gradient rather than the conventional gradient for parameter updates. The natural gradient takes into account the information geometry of the parameter space, leading to more efficient and stable learning. The natural gradient $\tilde{\nabla}$ is defined as:

$$\tilde{\nabla} S(\theta|y) \propto I_S(\theta)^{-1} \nabla S(\theta|y)$$

where

- $I_S(\theta)$ is the Fisher Information Matrix for the scoring rule S
- $\nabla S(\theta|y)$ is the standard gradient of the scoring rule with respect to the parameters θ.

Here is a typical training loop for fitting a NgBoost model, illustrated by Figure 10 from Duan and all:

- **Initialization**: Initialize the parameters θ to fit the marginal distribution of y.

- **For Each Boosting Stage**:
  - Compute the natural gradients of the scoring rule with respect to the predicted parameters.
  - Fit a set of base learners to these natural gradients.
  - Update the parameters θ by moving in the direction of the negative scaled natural gradients.

- **Output**: The final model predicts a probability distribution for a new input x, with the parameters θ obtained as an additive combination of the outputs from all the boosting stages.

  NGBoost learns the parameters sequentially, where each base learner is trained to correct the residuals (errors) of the ensemble so far. The use of the natural gradient makes the updates more efficient and invariant to the choice of parametrization, leading to better training dynamics and performance.

4.3 Ensemble models

Ensemble learning in machine learning is a robust methodology that involves combining multiple models to improve the overall performance, accuracy, and robustness of predictions. The core idea behind ensemble learning is that a group of already trained learners can come together to form a stronger learner, thereby improving the model's predictions. This approach is particularly beneficial in reducing errors that might arise from a single model misspecification as well as the risk of overfitting, in enhancing the accuracy of predictions, and dealing with the bias-variance trade-off more effectively. The development of ensemble models stems from the realization that different models capture different patterns in the data, and thus, combining these models can lead to more comprehensive and accurate predictions. There are primarily three types of ensembles learning methods: Bagging, Boosting, and Stacking, each with its unique mechanism and purpose. As mentioned above, two of the models already covered (XgBoost and NgBoost) are Boosting Ensemble models. Since we covered the Boosting mechanism in detail, we here focus on the two remaining methods.

Bagging (Bootstrap Aggregating) Bagging involves creating multiple models, each trained on a random subset of the training data. These subsets are drawn with replacement, known as bootstrapping. Decision trees are commonly used in bagging, with Random Forest being a quintessential example. In a Random Forest, numerous decision trees are constructed, and their outputs are aggregated, typically by averaging, to yield the final prediction. The primary advantage of bagging is its ability to reduce variance, making the model less prone to overfitting. However, it does not necessarily reduce bias, which means if the base learners are biased, the ensemble model will also be biased.

Stacking Stacking , or Stacked generalization, is a technique that involves training multiple models on the same data and then using another model, often referred to as a meta-learner, to combine their predictions. The base learners are typically different kinds of algorithms, making stacking a heterogeneous approach. The meta-learner, which could be a different algorithm, is trained on the outputs of the base learners, learning how to best combine their predictions to improve accuracy. Stacking is particularly effective when the base models are significantly diverse, as the meta-learner can effectively capture the different aspects of the data represented by each base model. There exist numerous ways to construct a meta-learner or to combine models' predictions using Stacked generalization, the most common being averaging, weighted averaging or feeding the Single models' outputs to a subsequent machine learning model.
In comparing these methods, bagging is known for its simplicity and effectiveness in reducing overfitting, especially in the case of complex models like decision trees. Boosting, while more complex, is generally more powerful in terms of performance, capable of improving both bias and variance but at the cost of being more sensitive to noise and outliers. Stacking, being the most sophisticated, can yield even better results by optimally combining diverse models, but it comes with the complexity of training and tuning multiple layers of models. In this study we chose to make use of stacking by using single layer feed-forward neural networks as

meta-learners, because of the flexibility provided by these architectures. (See Hornik et al. [47])

4.3.1 X-N-L-T

Our first ensemble model consists of a combination of the four previous one with a single layer 4 neurons feed-forward network as an ensembling method. We choose to have a Feed-Forward aggregation instead of the averaging aggregation frequently encountered in the literature to allow for more flexibility. We especially try to account for the fact that models' forecast performance are often conditional on market conditions by letting the Feed-Forward Network learn the weight to attribute to models forecasts depending on market conditions. Essentially, the X-N-L-T model (Figure 11) is as follow :

$$\sigma_{t+k} = FFN(X_t) = ReLU(w_1 Transformer(X_t) + w_2 LSTM(X_t) + w_3 XgBoost(X_t) + w_4 NgBoost(X_t))$$

4.3.2 CGARCH enhanced models

This category of models consists of combinations of each of the 5 previous models with our EGARCH model using a 2 neurons single layer Feed-Forward network, which is essentially a multi-linear regression with a non linearity. For each the 5 ML models, the FFN takes as input the output forecasts from the ML model and the EGARCH model.

CGARCH enhanced Transformer (Figure 12)

$$\sigma_{t+k} = FFN(X_t) = ReLU(w_1 Transformer(X_t) + w_2 CGARCH(X_t))$$

CGARCH enhanced Long-Short Term Memory Network(Figure 13)

$$\sigma_{t+k} = FFN(X_t) = ReLU(w_1 LSTM(X_t) + w_2 CGARCH(X_t))$$

CGARCH enhanced Extreme Gradient Boosted tree (Figure 14)

$$\sigma_{t+k} = FFN(X_t) = ReLU(w_1 XgBoost(X_t) + w_2 CGARCH(X_t))$$

CGARCH enhanced Natural Gradient Boosted tree (Figure 15)

$$\sigma_{t+k} = FFN(X_t) = ReLU(w_1 NgBoost(X_t) + w_2 CGARCH(X_t))$$

X-N-L-T - CGARCH (Figure 16)

$$\sigma_{t+k} = FFN(X_t) =$$

$$ReLU(w_1 Transformer(X_t) + w_2 LSTM(X_t) + w_3 XgBoost(X_t) + w_4 NgBoost(X_t) + w_5 CGARCH(X_t))$$

5 Empirical methodology

As stated above, our study's goal is to compare various learning algorithms on 20 forecasting problems. The problems differ on two axes : Assets and horizon, but essentially comes down to producing a point forecast for an asset realized daily volatility at t+h. We outline below how we built our datasets for the machine learning models.

5.1 Rolling window

For this experiment, since our explanatory variables consist of lags of our target variables, we adopt a static sliding rolling window method for building our input and target pairs dataset . The sliding step size is 1. The sliding rolling window methodology, illustrated on (Figure 17), takes m lags of the realized volatility in order to forecast our target horizon. Let's denote the time series as $\{X_t\}$, where $t = 1, 2, \ldots, T$, and T is the total number of observations. For a rolling window of size m, the training data at time t (for $t > m$) would be $\{X_{t-m}, X_{t-m+1}, \ldots, X_{t-1}\}$. The model uses this data to predict X_{t+h}.
At each time step t, with $t > m$, the model is trained on the window of observations:

$$W_t = \{X_{t-m}, X_{t-m+1}, \ldots, X_{t-1}\}$$

Based on the training data in W_t, the model predicts the next value $\hat{X}_{t+h}$, which is then compared to the actual observed value X_{t+h} to compute the prediction error.
After the prediction at time t, the window is updated by including the next observation X_t and excluding the oldest observation X_{t-m}, and the model moves to the next time step $t+1$. The updated window W_{t+1} becomes:

$$W_{t+1} = \{X_{t-m+1}, X_{t-m+2}, \ldots, X_t\}$$

This process continues until the end of the time series.

At T_0 we use the values between $[T_{0-m}$, T_0] as input to As discussed, the number of lags m was an hyperparameter for each [A-H-M] model.Because of computing resources constraints, we estimate all the hyperparameters only on the 60 days ahead forecasting version of each asset class for each model. We then use the same architecture for the N-days ahead forecasting problems.

5.2 Dataset partitionning

An important part of our research consists in estimating and selecting among a substantial set of models varying by hyperparameters. We adopt a variation of a popular model selection methodology used by Kelly and Xiu [48], the fixed training -validation split. Since we evaluate and compare the performance of both single and ensemble models, for each forecasting horizon and each asset, we divide our dataset in four parts : a training set D1 for the single models (50% of dataset), a validation set D2 for the single models(10% of dataset), a training set D3 for the ensemble models(20% of our dataset), and a final test set D4 (20% of dataset), on which we test both the single and ensemble models. Our main

concern with proceeding with such a split was to prevent data leakage between the ensemble models and the single models, since the ensemble model's inputs consists of outputs from the single models. The splits are temporally ordered, meaning that historically, D1 occurs before D2, which occurs before D3, which occurs before D4. We proceed in such a way in order to prevent information leakage backward in time (See Kelly and Xiu [48]). The final test set D4 was only visited once at the end for generating the results tables after all models had been trained on their respective training and validation sets. This means that none of the models (either simple or ensemble) had seen the data in D4 during training or validation, only at the end for generating evaluation and comparison metrics.

5.3 Hyperparameter tuning

Here, we detail in Table 1 and Table 2 all the hyperparameters tested on the [**A-60**] forecasting models. The tuning is performed by grid-search.

6 Results, comparison, and discussion

6.1 Final models architectures

In this section, we review the final values found by our grid-search hyper-parameters tuning process on the [**A-60**] models. The results are displayed in Table 3.

6.2 Results and Discussion

In this section, we discuss and compare the final out of sample score of our models. First of all, in order to provide rigorous statistical backing for our discussions, we display the results of models vs benchmarks MGW tests for equivalent predictive abilities for each [**A-H**] problem, which can be found in table 4. The displayed value in each cell gives us the p-value of for the MGW test testing for :
H_0**: Model in row has a statistically different predictive ability than model in Column (Benchmark)**.
Secondly, tables 5 to 9 show for each horizon the results of pairwise equivalent predictive ability MGW test for every pair combination of our models. The displayed value in each cell represents the p-value for the test :
H_0 **: Model in row has a statistically different predictive ability than model in Column**
Moreover, we display the results of the MCS test at $\alpha = 0.01$ for [**A-H**] groups in table 10. This essentially tells us what's the smallest best set of superior models for each [**A-H**] problem with a 99% confidence level, or framed differently, it shows us the set of models for which predictive ability is superior to the rest, but not differentiable among themselves at a 99% confidence level.
Finally, The out of sample performance measured by MAE, RMSE, QLIKE and OF of each [**M-H**] models grouped by assets are found in tables 11 to 15. We also display the In-Sample MAE, RMSE and OF for each [**M-H**] groups from tables 16 to 20.

To evaluate and compare our model's predictive ability, we use an extensive framework to insure the robustness of our findings. First, we use the MGW superior predictive ability test results displayed from tables 4 to 9 in terms of p-value, to statistically assess and compare models' superiority.

Subsequently, we review the more classical out-of-sample performance metrics displayed from tables 11 to 15. To complete our analysis, we undertake an extensive review of the results displayed in Table 10, which allows us to perform a multidimensional comparison of our models: This single table informs us on asset-wise, horizon-wise, as well as asset and horizon-wise assessment of model's superiority according to the MCS.

MGW equivalent predictive ability test Against Benchmarks

To rigorously evaluate the predictive power of various financial models, we delve into an analysis based on the MGW test for equivalent predictive ability. The p-values obtained from these tests are critical indicators, with lower values suggesting that a model's predictive performance is statistically different from the benchmark model's. We set a significance threshold at **0.01** to delineate models that meaningfully outperform the benchmarks.

The first thing we can point out from analyzing this table is that almost none of our models is simultaneously robust to both horizons and assets in terms of being different from the benchmarks. For the first 4 assets, the CGARCH-enhanced models exhibit exceptional predictive power across all horizons, with almost all of them being statistically different to the benchmarks for every horizons On the S&P500 volatility, The single models, as far as they are concerned, display some mixed performance across the different horizons. Most of them are unable to beat both benchmarks for every horizon but nonetheless, most models demonstrates different predictive ability against the CGARCH model. The most consistently different single model for forecasting S&P500 volatility on multiple horizon seems to be the XgBoost, and the most consistently different of the enhanced models is the Trans CGARCH model. Similar to the S&P 500, some various performance is nonetheless recorded for the single models, with the boosting based models (and their enhanced versions) demonstrating the most robustness to horizon for the NASDAQ. On the RUSSEL 2000 volatility nonetheless aside from the two models discussed below, their is no clear model that demonstrates consistent different predictive ability against the benchmarks. The MGW results for GOLD and oil seem to be equivalent to the S&P's, with almost no model being unequivocally different to both benchmarks, but most models beating the CGARCH models.
It is interesting to note that most of the different predictive ability results against both benchmarks are observed for the longer term horizons, 20 days and 60 days. Dissecting the results from the horizon dimension, it appears that the models that most consistently beat the benchmarks on every horizon are the enhanced models, notably the enhanced boosting models. In fact, both NG CGARCH and XG CGARCH demonstrate different predictive ability against the benchmarks on all [**A-H**] problems.

Model Confidence Set Best : models at 99.9% Confidence level (Table 10)

This section analyzes the model confidence sets constructed for each [**A-H**] problem. Before delving into the comparative analysis, we tabulate the frequency of each model's appearance in the MCS results across assets and horizons to discern the most dominant performers:

- **NG GARCH** : Contained in 100% of Model Confidence Sets
- **Transformer** : Contained in 100% of Model Confidence Sets
- **LSTM** : Contained in 92% of Model Confidence Sets
- **XgBoost** : Contained in 72% of Model Confidence Sets

- **X-N-L-T** : Contained in 64% of Model Confidence sets

The MCS results, when viewed in totality, reveal the dominance of certain models that are robust across horizons and assets. The NG GARCH model particularly stands out for its adaptability and resilience, indicating that it encapsulates a balance of responsiveness and structural understanding that is applicable to different market conditions, asset-specific conditions, regimes and forecasting windows.
Its dominant presence across most assets and horizons is indicative of its robustness. The model's hybrid nature, combining the CGARCH framework's ability to model time-varying volatility with the predictive power of boosted weak learners optimized by natural gradient descent likely contributes to its wide applicability. The addition of CGARCH components helps to address volatility clustering, a common characteristic in financial time series. It is the only enhanced model appearing in the top 5 of most recurring confidence sets models. Its omnipresence in virtually all the [**A-H**] forecasting problems best models sets may suggest that while the model is generally the versatile of the ones presented in this study.
The Transformer model also appears in all the best models set. This suggests the effectiveness of the underlying self-attention mechanism in extracting relevant feature in diverse market conditions and, and its ability to do so without being augmented by an econometric model.
In term of robustness to both dimensions, we also notice the recurrence of the HAR model in the best sets of many of our [**A-H**] problems, which is in line with its state of the art status in the volatility forecasting literature when it comes to econometric models. Its versatility seems more pronounced for forecasting the realized volatility of the NASDAQ and Crude oil, regardless of the horizon considered.

Models Out-of-Sample Performance Metrics

Following the results of Patton [49] on the robustness of QLIKE and MSE (RMSE) to volatility proxies for model comparison, we chose to use these two metrics as performance metrics to compare our models in this section. The two others, namely MAE and OF, are displayed for additional informative purposes, and are not used to rank our models. For all the models, regardless of asset and horizon, MAE is almost always in line with RMSE in terms of ranking, so there is no loss of information in disregarding MAE. While all the 4 most robust superior models filtered from the MCS test score lower RMSE and MAE than both benchmarks across assets and horizons, We restrict the following results discussion to only mention models that simultaneously beat our two Benchmarks on both RMSE and QLIKE.
An interesting thing to note is that aside from a few occurrences, there are only a few models that beat both benchmarks on QLIKE, and this superiority is not robust to assets and horizons. The CGARCH model demonstrates an almost absolutely robust superiority according to QLIKE. This result is in-line with what is expected from a statistical model designed and trained by maximizing its likelihood to fit the distribution of the data. The only models to occasionally breach this robust superiority of the CGARCH are the CGARCH enhanced models. This outlines the gain in terms of goodness of fit from augmenting machine learning models with statistical or econometric ones. Aside from the CGARCH, The X-N-L-T CGARCH and LSTM CGARCH display an almost absolute robust superiority across assets for the two longer horizons according to QLIKE.
In terms of RMSE, none of our machine learning models outperforms the benchmark models in a significant way . In other words, the RMSE of all the models (including the benchmarks) are not significantly different from one another across assets and horizons.

Additional Results

MGW pair-wise different predictive ability test between models

Tables 5 to 9 show us the p-values from pairwise predictive ability MGW test between all our models. We perform the analysis at a threshold of 1%. For tables 6 to 10, The displayed value in each cell represents the p-value for the test

H_0**: Model in column as a different predictive ability than model in row**

It is important to start by pointing out that there is no model displaying statistically different forecasting ability against all other models, regardless of asset or horizon. A significant and surprising pattern seems to be the consistence with which some pairs of models forecasting abilities remain statistically non different across assets and horizon, namely the following pairs : XgBoost and NgBoost, LSTM and Transformer. This lets us know that according to the MGW test, for realized volatility forecasting, there is no difference between XgBoost or NgBoost and between LSTM and Transformer, regardless of assets or forecasting horizon It is also interesting to note that in almost all cases for the Stock Indices, the CGARCH enhanced models demonstrate a significantly different forecasting ability compared to their simple versions. This is not the case for the commodities. There is therefore an added-value to enhancing machine learning models with CGARCH models in forecasting the realized volatility of Stock Indices. Also, it is noteworthy that across assets and horizons, the CGARCH enhanced models display the most statistically different forecasting ability when compared pairwise to other models on the same [**A-H**] problem. Particularly, on every [**A-H**] problem, our enhanced gradient boosted models display a forecasting performance statistically different from virtually all the other models. The only occurrences of any of these enhanced boosted models performance being statistically non-different happens against the other enhanced boosting model on the commodity at the **h-60** horizon. This is an interesting finding, as it tells us that on the longer forecasting horizons, there is no difference in using either an enhance NgBoost or NgBoost to forecast the commodities volatility. In other words, NG CGARCH and XG CGARCH are statistically different from all other models on every [**A-H**] problem, but not always statistically different from each other, especially on long horizons. This result further ensures us that there is an actual value in augmenting boosted ensemble models with econometric models, but still leaves us unclear on the differentiation between NgBoost and XgBoost, even when enhanced.

Overall, the extensive evaluation framework solicited in this study didn't point us towards a unanimously superior model demonstrating robustness across our two analysis dimensions. Nonetheless, their was a distinctive tendency of the enhanced boosting boosted models to stand out, particularly the NG CGARCH model. This model was able to demonstrate absolute robustness in term of difference in predictive ability against all other models. Additionally, the NG CGARCH model is contained in virtually 100% of model confidence sets. The only time it wasn't able to demonstrate absolute robustness was on the performance metrics. Even without scoring high on those, based on its robustness demonstrated from the MGW and MCS tests, it seems reasonable to consider that the main finding of this study is the superiority of the NG CGARCH model against all the models in this study. In fact, this result is not surprising given the fact the NG CGARCH comes down to a stacked ensemble of models, first by the addition of te CGARCH to NgBoost but also by the fundamental nature of the NgBoost model, which is itself a boosted ensemble of weak learners. The use of multiple learners to extract different features from our models is very similar to what is done by the attention layers in the transformer, or a convolution layer in a convectional neural network. In this sense, it is not surprising that the transformer would also appear in 100% of model confidence sets. On top of the features learned by all the weak learners, the addition of the

economic properties and stylized facts captured by a parametric model like CGARCH allows us to elegantly infuse some domain knowledge to our final model. Additionally, it is worth mentioning that although the NG CGARCH emerges as a majority winner in terms of robustness, its predictive ability is indistinguishable from the XG CGARCH's according to the MGW test. Also, the same test informs us of the superior predictive ability of enhanced boosting models compared to their non enhanced counterpart. We know that the superiority of XgBoost over HAR and LSTM for 1 day ahead realized volatility forecasting on multiple stocks was already established by Teller et al. [28] ; Similarly, the superiority of GARCH enhanced transformers over GARCH, LSTM, ANN, Transformer, and GARCH enhanced LSTM and Transformer for 1 day ahead S&P500 historical volatility forecasting was established by Ramos-Pérez et al. [19]. Moreover, it turned out that there was no clear edge or from aggregating all our models together using a FFN for ensembling, with or without a component GARCH enhancement. Neither the X-N-L-T nor the X-N-L-T CGARCH demonstrated robust superiority on any dimension. This result is hardly surprising; The literature suggests that there is rarely an edge to be gained from merely combining outputs from different specialized architectures without infusing any domain knowledge or intelligent leveraging each architecture's strength on a different part of the problem. Reflecting on our study, our results for the non-boosting models do not seem conclusive or clear cut enough to extract any valuable or trustworthy findings regarding these models. Nonetheless, the most interesting and trustworthy results we find is that there is an added value to enhancing boosting models with a component GARCH model, which is demonstrated by the significant gains in QLIKE and the robust MGW test low p-value obaited between enhanced and non-enhanced boosting models. Moreover, the NG CGARCH shines the best in term of forecasting power and robustness to forecasting horizon and asset. The superiority of a boosting model is in line with what we've been seeing in the last two biggest time-series forecasting competitions, the Makridakis et al. [50] M4 and Makridakis et al. [51] M5 competitions, where Boosting models have come out in the top most robust models for forecasting more than 40000 time-series.

7 Conclusion and Directions for Future Work

The undertaking of this thesis was to compare and explore the robustness of different machine learning architectures in forecasting the realized volatility of 5 popular and liquid assets. We used intraday prices sampled at a 5mn frequency from the TAQ database collected from 2005 to 2021 for 5 assets proxied by ETFs. The robustness was tested along two axes, the forecasting horizon, and the asset on which we perform the forecast. Besides, we also studied the added value from enhancing our plain machine learning models with a component GARCH model in a stacked ensemble manner using a single layer feed-forward neural network. We adopted a rigorous processing and partitioning of our data to make sure that the final evaluation was made on a subset (D4, or test set) that was never seen during training. In order to obtained sound and trustworthy results from our study, we used an extensive evaluation framework comprising the classical performance metrics encountered in academia (RMSE, QLIKE), as well as two statistical tests to assess and differentiate forecasting power among models. From this framework the main conclusions was that the boosted models perform better when enhanced with a CGARCH, and that, restricted to the assets and horizons considered, the NG CGARCH is the best model for forecasting realized volatility regardless of asset or horizon. Nonetheless, there are several avenues for improvement that could be undertaken from our results. First of all, a major shortcoming of the study is that it compared only the pure auto-regressive forecasting power of the architectures, in the sense that we didn't include any exogenous regressor than lags of our response variable, not even the returns series. A more complete and thorough study could investigate the robustness of these architecture when enhanced with exogenous regressors such as macro-economic variables or other market variables related to the assets. Secondly, the number of assets and horizons could be easily extended in order to have a more comprehensive assessment of robustness. Furthermore, given that the state of the art econometric models for volatility forecasting are most of the time regime switching hidden markov models, it would be interesting to explore whether or not regime switching versions of our architectures demonstrate superior robustness and predictive ability against regime switching benchmarks. It would also be interesting to explore the extent to which the current version of these architectures (non regime switching) are inherently able to capture the different volatility regimes filtered by a state of the art state space filter (like the Hamilton filter). Also, this study could be extended to look at the multiple assets volatility forecasting problem as an actual multivariate forecasting problem. Besides, we only studied point-forecasting, which means we're only forecasting the volatility of one point in time into the future; a more useful variation would be to study how well the models perform in term of range forecasting, which is forecasting the volatility of every day between today and the forecasting horizon. Another shortcoming of our study is the lack of confidence intervals around our forecasts, or for our parameters. Since most applications using volatility forecasts as input are heavily sensitive to the uncertainty around the forecasts, it is critical to have a statistical assessment of the uncertainty around a model's parameters and forecasts in the form of confidence intervals. To this effect, a relatively new framework for ML forecasts uncertainty quantification known as conformal prediction has emerged. This framework allows us to produce confidence interval for any forecasting model, even non parametric and non statistical ones. Quantifying uncertainty would also allow us to more rigorously benchmark our models against Bayesian forecasting models like Gaussian processes. Since our models are very data-hungry deep learning models that have been proven to get asymptotically better with more training data, another critical improvement to the study that may alternate the results is to augment the size of the training data by going further in the past to gather more volatility clusters and long-term behavior. We were limited in this capacity by the hardware constraints that come with treating high frequency data and the availability of historical intraday series on our source database. Also, exploring more alternative

training frameworks, loss functions, or switching to a quasi likelihood maximizing framework instead of loss minimization might have given more interesting results. An interesting avenue in this regard is to experiment with other kind of weak learners in the boosting models instead of only focusing on trees and linear regressors. Moreover, we could improve the ensemble models built from our single models by experimenting with more ensembling schemes that allow to leverage each architecture's strength instead of the naive non-linear weighted averaging we used here. Lastly one of the most promising avenue in our opinion is to extend the number of architectures studied to include the state of the arts models for time series forecasting. The most promising direction would be to experiment with the following novel architectures which currently constitutes the state of the art of ML for time-series forecasting: state space models like the N-BEATS by Chapados, Bengio, Oreshkin et al. [86] (that outperformed every model on the Makridakis et al. [50] M4 dataset), or the N-Hits Challu et al. [90], Physics Informed Neural Networks (Cuomo et al. [88]), the temporal fusion transformer (Lim et al. [31]), the attentional copulas transformer by Ashok et al. [87] from ServiceNow and MILA, or more recently the mamba model by Gu and Dao [91] which is, as of our final submission, the current frontier in ML time-series forecasting.

References

[1] Wenbo Ge, Pooia Lalbakhsh, Leigh Isai, Artem Lenskiy, and Hanna Suominen. Neural network–based financial volatility forecasting: A systematic review. *ACM Computing Surveys*, 55:1–30, 01 2023. doi: 10.1145/3483596.

[2] Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. URL `https://papers.neurips.cc/paper/7181-attention-is-all-you-need.pdf`.

[3] Lee Gary and Robert F Engle. A permanent and transitory component model of stock return volatility, 10 1993. URL `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5848`.

[4] Robert F Engle and Andrew J Patton. 2 - what good is a volatility model?*, 01 2007. URL `https://www.sciencedirect.com/science/article/pii/B9780750669429500042`.

[5] Ser-Huang Poon and Clive. Forecasting volatility in financial markets: A review. *Journal of Economic Literature*, 41:478–539, 06 2003. doi: 10.1257/002205103765762743.

[6] Ser-Huang Poon and Clive Granger. Practical issues in forecasting volatility. *Financial Analysts Journal*, 61:45–56, 2005. URL `https://www.jstor.org/stable/4480636`.

[7] Robert F Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50:987–1007, 07 1982.

[8] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31:307–327, 04 1986. doi: https://doi.org/10.1016/0304-4076(86)90063-1.

[9] Daniel B Nelson. Conditional heteroskedasticity in asset returns: A new approach. *Econometrica*, 59:347–370, 03 1991. doi: 10.2307/2938260.

[10] LAWRENCE R GLOSTEN, RAVI JAGANNATHAN, and DAVID E RUNKLE. On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance*, 48:1779–1801, 12 1993.

[11] Torben G Andersen, Tim Bollerslev, Francis X Diebold, and Paul Labys. Modeling and forecasting realized volatility. *SSRN Electronic Journal*, 2001. doi: 10.2139/ssrn.267792.

[12] Eric Ghysels, Arthur Sinko, and Rossen Valkanov. Midas regressions: Further results and new directions. *Econometric Reviews*, 26:53–90, 02 2007. doi: 10.1080/07474930600972467.

[13] F Corsi. A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7:174–196, 11 2008. doi: 10.1093/jjfinec/nbp001.

[14] Gonzalez Miranda and N Burgess. Modelling market volatilities: the neural network perspective. *The European Journal of Finance*, 3:137–157, 06 1997. doi: 10.1080/135184797337499.

[15] P Tino, C Schittenkopf, and G Dorffner. Financial volatility trading using recurrent neural networks. *IEEE Transactions on Neural Networks*, 12:865–874, 07 2001. doi: 10.1109/72.935096.

[16] Shaikh A Hamid and Zahid Iqbal. Using neural networks for forecasting volatility of sp 500 index futures prices. *Journal of Business Research*, 57:1116–1125, 10 2004. doi: 10.1016/s0148-2963(03)00043-2.

[17] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270:654–669, 2018. URL `https://econpapers.repec.org/RePEc:eee:ejores:v:270:y:2018:i:2:p:654-669`.

[18] Peter R Hansen and Asger Lunde. A forecast comparison of volatility models: does anything beat a garch(1,1)? *Journal of Applied Econometrics*, 20:873–889, 2005. doi: 10.1002/jae.800.

[19] Eduardo Ramos-Pérez, Pablo Alonso-González, and José Javier Núñez-Velázquez. Multi-transformer: A new neural network-based architecture for forecasting sp volatility. 9:1794–1794, 09 2021. doi: 10.3390/math9151794.

[20] Donaldson R Glen and Mark Kamstra. An artificial neural network-garch model for international stock return volatility. *Journal of Empirical Finance*, 4:17–46, 1997. URL `https://econpapers.repec.org/RePEc:eee:empfin:v:4:y:1997:i:1:p:17-46`.

[21] Michael Y Hu and Christos Tsoukalas. Combining conditional volatility forecasts using neural networks: an application to the ems exchange rates. *Journal of International Financial Markets, Institutions and Money*, 9:407–422, 1999. URL `https://econpapers.repec.org/RePEc:eee:intfin:v:9:y:1999:i:4:p:407-422`.

[22] Mary Malliaris and Linda Salchenberger. Using neural networks to forecast the sp 100 implied volatility. *Neurocomputing*, 10:183–195, 03 1996. doi: 10.1016/0925-2312(95)00019-4.

[23] Christian L Dunis and Xuehuan Huang. Forecasting and trading currency volatility: an application of recurrent neural regression and model combination. *Journal of Forecasting*, 21:317–354, 2002. doi: 10.1002/for.833.

[24] Kim Christensen, Mathias Siggaard, and Bezirgen Veliyev. A machine learning approach to volatility forecasting. *Journal of Financial Econometrics*, 06 2022. doi: 10.1093/jjfinec/nbac020.

[25] Xunfa Lu, Danfeng Que, and Guangxi Cao. Volatility forecast based on the hybrid artificial neural network and garch-type models. *Procedia Computer Science*, 91:1044–1049, 2016. doi: 10.1016/j.procs.2016.07.145.

[26] Werner Kristjanpoller and Marcel C Minutolo. Gold price volatility: A forecasting approach using the artificial neural network–garch model. *Expert Systems with Applications*, 42:7245–7251, 11 2015. doi: 10.1016/j.eswa.2015.04.058.

[27] Werner Kristjanpoller and Marcel C Minutolo. Forecasting volatility of oil price using an artificial neural network-garch model. *Expert Systems with Applications*, 65:233–241, 12 2016. doi: 10.1016/j.eswa.2016.08.045.

[28] Andreas Teller, Uta Pigorsch, and Christian Pigorsch. Short- to long-term realized volatility forecasting using extreme gradient boosting. *SSRN Electronic Journal*, 2022. doi: 10.2139/ssrn.4267541.

[29] Wenting Liu, Zhilun Gui, Guilin Jiang, Li Tang, Lijun Zhou, Wei Leng, Xulong Zhang, and Yujiang Liu. Stock volatility prediction based on transformer model using mixed-frequency data. *arXiv (Cornell University)*, 09 2023. doi: 10.48550/arxiv.2309.16196.

[30] Wenbo Ge, Pooia Lalbakhsh, Leigh Isai, Artem Lensky, and Hanna Suominen. Comparing deep learning models for the task of volatility prediction using multivariate data. *arXiv (Cornell University)*, 06 2023. doi: 10.48550/arxiv.2306.12446.

[31] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *arXiv:1912.09363 [cs, stat]*, 09 2020. URL `https://arxiv.org/abs/1912.09363`.

[32] Perry and Philip R. Time-variance relationship of security returns: Implications for the return-generating stochastic process. *Journal of Finance*, 37:857–870, 1982. URL `https://ideas.repec.org/a/bla/jfinan/v37y1982i3p857-70.html`.

[33] Adrian R. Pagan and G.William Schwert. Alternative models for conditional stock volatility. *Journal of Econometrics*, 45:267–290, 07 1990. doi: 10.1016/0304-4076(90)90101-x.

[34] Sang Hoon Kang, Sang-Mok Kang, and Seong-Min Yoon. Forecasting volatility of crude oil markets. *Energy Economics*, 31:119–125, 01 2009. doi: 10.1016/j.eneco.2008.09.006.

[35] Ole E. Barndorff-Nielsen and Neil Shephard. Estimating quadratic variation using realized variance. *Journal of Applied Econometrics*, 17:457–477, 2002. doi: 10.1002/jae.691.

[36] YAKOV AMIHUD and HAIM MENDELSON. Trading mechanisms and stock returns: An empirical investigation. *The Journal of Finance*, 42:533–553, 07 1987. doi: 10.1111/j.1540-6261.1987.tb04567.x.

[37] MILTON HARRIS and ARTUR RAVIV. The theory of capital structure. *The Journal of Finance*, 46:297–355, 03 1991.

[38] Ananth Madhavan. Market microstructure: A survey. *Journal of Financial Markets*, 3:205–258, 08 2000. doi: 10.1016/s1386-4181(00)00007-0.

[39] Yacine Ait-Sahalia, Per A Mykland, and Lan Zhang. Ultra high frequency volatility estimation with dependent microstructure noise, 05 2005. URL `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=731035`.

[40] Federico M. Bandi, Jeffrey R. Russell, and Chen Yang. Realized volatility forecasting and option pricing. *Journal of Econometrics*, 147:34–46, 11 2008. doi: 10.1016/j.jeconom.2008.09.002.

[41] F. M. Bandi and J. R. Russell. Microstructure noise, realized variance, and optimal sampling. *The Review of Economic Studies*, 75:339–369, 2008. URL `https://www.jstor.org/stable/20185035`.

[42] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997. doi: https://doi.org/10.1162/neco.1997.9.8.1735.

[43] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 03 1986. doi: 10.1007/bf00116251. URL `http://www.hunch.net/~coms-4771/quinlan.pdf`.

[44] Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Taylor Francis, 01 1984. URL https://books.google.ca/books/about/Classification_and_Regression_Trees.html?id=JwQx-WOmSyQC&redir_esc=y.

[45] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, page 785–794, 2016. doi: 10.1145/2939672.2939785.

[46] Tony Duan, Anand Avati, Daisy Yi Ding, Khanh K Thai, Sanjay Basu, Andrew Y Ng, and Alejandro Schuler. Ngboost: Natural gradient boosting for probabilistic prediction, 06 2020. URL https://arxiv.org/abs/1910.03225.

[47] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 01 1989. doi: 10.1016/0893-6080(89)90020-8. URL https://researchers.dellmed.utexas.edu/en/publications/multilayer-feedforward-networks-are-universal-approximators.

[48] Bryan T Kelly and Dacheng Xiu. Financial machine learning. *Social Science Research Network*, 07 2023. doi: 10.2139/ssrn.4501707. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4501707.

[49] Andrew J Patton. Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics*, 160:246–256, 01 2011. doi: 10.1016/j.jeconom.2010.03.034.

[50] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36:54–74, 01 2020. doi: 10.1016/j.ijforecast.2019.04.014. URL https://www.sciencedirect.com/science/article/pii/S0169207019301128.

[51] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38, 01 2022. doi: 10.1016/j.ijforecast.2021.11.013.

[52] Shihao Gu, Bryan T Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning, 09 2019. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3159577.

[53] Christian T Brownlees, Robert F Engle, and Bryan T Kelly. A practical guide to volatility forecasting through calm and storm. *SSRN Electronic Journal*, 2011. doi: 10.2139/ssrn.1502915.

[54] Francis X Diebold and Robert S Mariano. Comparing predictive accuracy. *Journal of Business Economic Statistics*, 20:134–144, 01 2002. doi: 10.1198/073500102753410444.

[55] Andrea Bucci. Forecasting realized volatility: a review, 12 2017. URL https://mpra.ub.uni-muenchen.de/83232/.

[56] Francesco Giordano, La Rocca, and Cira Perna. Forecasting nonlinear time series with neural network sieve bootstrap. *Computational Statistics Data Analysis*, 51:3871–3884, 05 2007. doi: 10.1016/j.csda.2006.03.003.

[57] Peter R Hansen, Asger Lunde, and James M Nason. The model confidence set. *Econometrica*, 79:453–497, 2011. URL https://www.jstor.org/stable/41057463.

[58] Stavros Degiannakis. Multiple days ahead realized volatility forecasting: Single, combined and average forecasts, 2018. URL https://mpra.ub.uni-muenchen.de/96272/.

[59] Daniel Borup, Jonas Nygaard Eriksen, Mads Markvart Kjær, and Martin Thyrsgaard. Predicting bond return predictability. *SSRN Electronic Journal*, 2020. doi: 10.2139/ssrn.3513340.

[60] Jia Zhai, Yi Cao, and Xiaoquan Liu. A neural network enhanced volatility component model. *Quantitative Finance*, 20:783–797, 02 2020. doi: 10.1080/14697688.2019.1711148.

[61] Andrés Vidal and Werner Kristjanpoller. Gold volatility prediction using a cnn-lstm approach. *Expert Systems with Applications*, 157:113481, 11 2020. doi: 10.1016/j.eswa.2020.113481.

[62] Hemanth Kumar, B Patil, and India .

[63] Ha Young Kim and Chang Hyun Won. Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with Applications*, 103:25–37, 08 2018. doi: 10.1016/j.eswa.2018.03.002. URL https://www.sciencedirect.com/science/article/pii/S0957417418301416.

[64] Ines Wilms, Jeroen Rombouts, and Christophe Croux. Multivariate volatility forecasts for stock market indices. *International Journal of Forecasting*, 09 2020. doi: 10.1016/j.ijforecast.2020.06.012.

[65] Nima Nonejad. An overview of dynamic model averaging techniques in time-series econometrics. *Journal of Economic Surveys*, 35:566–614, 01 2021. doi: 10.1111/joes.12410.

[66] Michael McAleer and Marcelo C Medeiros. Realized volatility: A review. *Econometric Reviews*, 27:10–45, 02 2008. doi: 10.1080/07474930701853509.

[67] Štefan Lyócsa and Peter Molnár. Volatility forecasting of strategically linked commodity etfs: gold-silver. *Quantitative Finance*, 16:1809–1822, 2016. URL https://ideas.repec.org/a/taf/quantf/v16y2016i12p1809-1822.html.

[68] Chun Liu and John M Maheu. Forecasting realized volatility: A bayesian model-averaging approach. *Journal of Applied Econometrics*, 24:709–733, 2009. URL https://www.jstor.org/stable/25608757.

[69] Christian Dorion and Nicolas Chapados. Volatility forecasting and explanatory variables: A tractable bayesian approach to stochastic volatility. *SSRN Electronic Journal*, 2012. doi: 10.2139/ssrn.1747945.

[70] Rémi Galarneau-Vincent, Geneviève Gauthier, and Frédéric Godin. Foreseeing the worst: Forecasting electricity dart spikes. *Energy Economics*, 119:106521, 03 2023. doi: 10.1016/j.eneco.2023.106521. URL https://www.sciencedirect.com/science/article/abs/pii/S0140988323000191.

[71] Qingfeng Liu, Qingsong Yao, and Guoqing Zhao. Model averaging estimation for conditional volatility models with an application to stock market volatility forecast. *Journal of Forecasting*, 01 2020. doi: 10.1002/for.2659.

[72] Zhifeng Dai, Huiting Zhou, Xiaodi Dong, and Jie Kang. Forecasting stock market volatility: A combination approach. *Discrete Dynamics in Nature and Society*, 2020:1–9, 06 2020. doi: 10.1155/2020/1428628.

[73] Chuong Luong and Nikolai Dokuchaev. Forecasting of realised volatility with the random forests algorithm. *Journal of Risk and Financial Management*, 11:61, 10 2018. doi: 10.3390/jrfm11040061.

[74] Hao Zhu, Lu Bai, Lidan He, and Zhi Li. Forecasting realized volatility with machine learning: Panel data perspective. *Journal of Empirical Finance*, 73:251–271, 09 2023. doi: 10.1016/j.jempfin.2023.07.003.

[75] Eduardo Ramos-Pérez, Pablo J Alonso-González, and José Javier Núñez-Velázquez. Forecasting volatility with a stacked model based on a hybridized artificial neural network. *Expert Systems with Applications*, 129:1–9, 09 2019. doi: 10.1016/j.eswa.2019.03.046. URL `https://www.sciencedirect.com/science/article/pii/S0957417419302209`.

[76] Soheil Almasi Monfared and David Enke. Volatility forecasting using a hybrid gjr-garch neural network model. *Procedia Computer Science*, 36:246–253, 2014. doi: 10.1016/j.procs.2014.09.087.

[77] Eunsuk Chong, Chulwoo Han, and Frank C Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 10 2017. doi: 10.1016/j.eswa.2017.04.030.

[78] Peter Reinhard Hansen, Asger Lunde, and James M Nason. Choosing the best volatility models: The model confidence set approach. *SSRN Electronic Journal*, 2003. doi: 10.2139/ssrn.399060.

[79] Melike Bildirici and Özgür Ömer Ersin. Improving forecasts of garch family models with the artificial neural networks: An application to the daily returns in istanbul stock exchange. *Expert Systems with Applications*, 36:7355–7362, 05 2009. doi: 10.1016/j.eswa.2008.09.051.

[80] Shaikh Hamid. Primer on using neural networks for forecasting market variables. URL `https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=71a031fac46fe8eaf1472d7355b87e92b4f51a5a`.

[81] Peter Carr, Liuren Wu, and Zhibai Zhang. Using machine learning to predict realized variance. *arXiv.org*, 09 2019. doi: 10.48550/arXiv.1909.10035. URL `https://arxiv.org/abs/1909.10035v1`.

[82] in .

[83] Hugo Gobato Souto and Amir Moradi. Forecasting realized volatility through financial turbulence and neural networks. *The Poznań University of Economics Review*, 9, 07 2023. doi: 10.18559/ebr.2023.2.737.

[84] Felix Gers, J Urgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm learning to forget: Continual prediction with lstm. *IEE*, 2:850–855, 1999. URL `https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e10f98b86797ebf6c8caea6f54cacbc5a50e8b34`.

[85] Jonathan H. Wright. Testing for a unit root in the volatility of asset returns. *Journal of Applied Econometrics*, 14:309–318, 1999. URL https://www.jstor.org/stable/223181.

[86] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv:1905.10437 [cs, stat]*, 02 2020. URL https://arxiv.org/abs/1905.10437.

[87] Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Nicolas Chapados, and Alexandre Drouin. Tactis-2: Better, faster, simpler attentional copulas for multivariate time series. *arXiv (Cornell University)*, 10 2023. doi: 10.48550/arxiv.2310.01327.

[88] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92, 07 2022. doi: 10.1007/s10915-022-01939-z.

[89] Simon van Norden and Robert John Vigfusson. Regime-switching models: A guide to the bank of canada gauss procedures. *SSRN Electronic Journal*, 1996. doi: 10.2139/ssrn.50565. URL https://www.bankofcanada.ca/wp-content/uploads/2010/05/wp96-3.pdf.

[90] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. N-hits: Neural hierarchical interpolation for time series forecasting. *arXiv (Cornell University)*, 01 2022. doi: 10.48550/arxiv.2201.12886.

[91] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 12 2023. URL https://arxiv.org/abs/2312.00752.

[92] Lan Zhang, Per A Mykland, and Yacine Aït-Sahalia. A tale of two time scales. *Journal of the American Statistical Association*, 100:1394–1411, 12 2005. doi: 10.1198/016214505000000169.

[93] Engle, r.f. and lee, g. (1999) a permanent and transitory component model of stock return volatility. cointegration, causality and forecasting a festschrift in honor of clive w.j. granger. oxford university press, new york. - references - scientific research publishing, 1999. URL https://scirp.org/reference/referencespapers?referenceid=1232518.

[94] Editor. Predicting volatility with heterogeneous autoregressive models, 05 2020. URL https://research.macrosynergy.com/predicting-volatility-with-heterogeneous-autoregressive-models/.

[95] Fulvio Corsi, Stefan Mittnik, Christian Pigorsch, and Uta Pigorsch. The volatility of realized volatility. *Econometric Reviews*, 27:46–78, 2008. URL https://econpapers.repec.org/RePEc:taf:emetrv:v:27:y:2008:i:1-3:p:46-78.

[96] A. D. Gordon. A review of hierarchical classification. *Journal of the Royal Statistical Society. Series A (General)*, 150:119, 1987. doi: 10.2307/2981629.

[97] Steven L. Salzberg. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16:235–240, 09 1994. doi: 10.1007/bf00993309.

A Training, testing, and evaluation process

A.1 Loss functions and Evaluation Metrics

We used 2 loss functions in the training process : the Huber-Loss function on the training set, and the MSE on the validation set. We evaluate and compare each **[A-H-M]** model variant with 2 loss functions : the RMSE loss and the Quasi Likelihood (Q-LIKE) function. We also use two additional metrics to inform us on a different view of models performance : The MAE loss and the Overvaluation frequency .

Huber Loss Function Because of its superior robustness we use the Huber Loss function as a loss function on the training set. The Huber loss function integrates both linear and quadratic scoring approaches. It uses hyperparameter delta δ which is tuned according to the data. The computed loss is linear (L1 loss) for values superior to δ, and quadratic (MSE Loss) for values inferior to δ.Huber Loss particularly useful in regression tasks where data may contain outliers or non-standard distributions, ensuring stable and robust training of the model.

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

Mean Absolute Error (MAE) MAE measures the average magnitude of errors in a set of predictions, without considering their direction. It is calculated as:

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

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of observations. It provides an average of absolute differences between the predictions and actual observations.

Mean Squared Error (MSE) MSE measures the average of the squares of the errors. It is more sensitive to larger errors compared to MAE and is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of observations. It is particularly useful when large errors are more significant than smaller ones.

Quasi Likelihood Function This metric is often used in statistical models to approximate the likelihood when the exact likelihood is either unknown or difficult to compute. In a machine learning context, it serves as a means to assess the likelihood of the model forecasts given the data :

$$QLIKE = \ n^{-1}\sum_{t=1}^{n}\left(\log\left(\hat{y}_t^2\right) + y_t^2\hat{y}_i^{-2}\right)$$

where y_t is the actual value, $\hat{y}_t$ is the predicted value, and n is the number of observations.

Overvaluation Frequency Function This metric helps us in evaluating the proportion of forecasts that were above the actual realized values for each models. Is is defined as :

$$OF_{t,\bar{T}} = \frac{1}{T}\sum_{t=1}^{T} I_{\hat{y}_t > y_t}$$

where y_t is the actual value, $\hat{y}_t$ is the predicted value, and n is the number of observations.

A.2 Over-fitting prevention

In machine learning, overfitting occurs when a model learns the training data too well, including its noise and outliers, which reduces its ability to perform well on unseen data. To prevent overfitting, several techniques are used, such as dropout regularization, batch normalization, and weight decay. In machine learning, overfitting is a common problem where a model performs well on training data but poorly on unseen data. Regularization techniques are essential for preventing overfitting, allowing models to generalize better from the training data to unseen data. We will discuss three fundamental regularization techniques: dropout regularization, batch normalization, and weight decay.

Dropout Regularization Dropout regularization is a stochastic technique to prevent overfitting in neural networks. It works by randomly deactivating a subset of neurons in a layer during each training iteration, which forces the network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.
For a fully connected layer with input vector $\mathbf{x}$ and output vector $\mathbf{y}$, without dropout, the relationship is given by:

$$\mathbf{y} = \phi(\mathbf{W}\mathbf{x} + \mathbf{b})$$

where ϕ is the activation function, $\mathbf{W}$ is the weight matrix, and $\mathbf{b}$ is the bias vector. When dropout is applied, this becomes:

$$\mathbf{y} = \phi((\mathbf{W} \odot \mathbf{D})\mathbf{x} + \mathbf{b})$$

where $\mathbf{D}$ is a diagonal matrix where each diagonal element D_{ii} is an independent Bernoulli random variable with probability p of being 1.
During training, dropout is applied, and during testing, it is not used but the weights are scaled by p to account for the reduced number of active neurons, which ensures that the expected sum of the inputs remains the same.

Batch Normalization Batch normalization (BN) is a technique to normalize the inputs of a layer to mitigate the problem of internal co-variate shift. BN standardizes the outputs of the previous layer by subtracting the batch mean and dividing by the batch standard deviation.
For a given feature x, BN transforms it as:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$
$$y_i = \gamma \hat{x}_i + \beta$$

where

- m is the size of the mini-batch
- μ_B is the mini-batch mean
- σ_B^2 is the mini-batch variance
- ϵ is a small constant for numerical stability
- γ and β are learnable parameters of the model.

  Batch Normalization allows for higher learning rates and reduces the strong dependence on initialization.

Weight Decay Weight decay is a form of L2 regularization that discourages large weights in the model through a penalty on the loss function proportional to the sum of the squares of the weights. This leads to a simpler model and helps to prevent over-fitting.
The loss function with weight decay is given by:

$$L_{new} = L_{original} + \frac{\lambda}{2N} \sum_{i=1}^{N} w_i^2$$

where

- $L_{original}$ is the original loss without regularization
- w_i are the weights of the model
- N is the total number of weights
- λ is the regularization parameter.

During training, this has the effect of updating the weights as if they are decaying towards zero, hence the name weight decay.
We use these three techniques in combination to mitigate the risk of overfitting and improve the model's generalization to new, unseen data.

A.3 Training and testing

Each [**A-H-M**] with different hyper-parameters was trained on the first training set D1 (with Huber loss), then used to produce forecasts on the validation set D2. The final specification of each [**A-H-M**] model was selected based on the forecast performance on the validation set (with MSE), which are considered pseudo-out-of sample forecasts (Kelly and Xiu [48]). For each [**A-H-M**] model, we find the best model through the tuning on D1 and D2, which is then re-trained on the full D1 + D2 set (60% of data). The training is done by minimizing the Huber Loss on D1, and using the MSE to compare and validate results on D2. We chose to use the Huber loss function on the training set because of its flexibility and adaptability in terms of scoring, in the

sense that it provides us with the benefits of both a L1 (MAE) and a L2 (MSE) scoring function. In particular, the flexibility provided by its δ hyper-parameter is well-suited for volatility series because they exhibit non-stationary, auto-correlation, and idiosyncratic jumps. The δ parameter is a way to allow the loss function itself to adapt to the volatility regime by switching from linear to quadratic scoring depending on the size of the error. The choice of using the MSE on the validation set was driven by its status as a standard of model evaluation on validation and test sets in the deep learning literature. Subsequently, we proceed to building the ensemble dataset as explained above, by making forecasts on the D3 dataset using the trained single models. The outputs from the single models forecasts on D3 are then used as input to the ensemble models training (with Huber loss) on D3 with the actual realized values as target. The final comparison of all our models evaluated on the (never seen by any model during training) D4 test set is performed using two functions that have been proved by Patton [49] to be robust to volatility proxies : the Quasi Likelihood and the (Root) Mean-Squared Error function. Our choice of these two was purely motivated by a need for an unbiased and robust ranking of our models which demonstrates robustness to proxies or sampling frequency, or signal-to-noise ratio (Patton [49]). Additionally, it seemed appropriate to use the Q-LIKE to have better comparability since one of our benchmark is a statistical distribution likelihood maximizing model (CGARCH).It also allowed us to have a preliminary assessment of how well our models fitted the volatility distributions without directly being optimized or trained to do so.

A.4 Evaluation

To evaluate and compare our final models forecasting accuracy across assets and horizons, we supplement our performance metrics with 2 evaluation frameworks : the Model Confidence Set method from Hansen et al. [78]. as suggested by Brownlees et al. [53] and the Multivariate Giacomini-White test from Borup et al. [59]. We succinctly discuss the two framework in this section.

Model Confidence set The MCS is a statistical method for model comparison. It provides a set of models from a larger collection, with a certain level of confidence that the set includes the best model according to a chosen loss function.
Given a set M^0 containing a finite number m_0 of models, indexed by $i = 1, \ldots, m_0$, the loss associated with model i in period t is denoted by L_{it}. The relative performance of models i and j in period t is given by:

$$d_{ijt} = L_{it} - L_{jt}$$

where $\mathbb{E}[d_{ijt}]$ is assumed to be finite and constant over t for all $i, j \in M^0$.
The set of superior models, denoted by M^*, is defined as:

$$M^* = \{i \in M^0 : \mu_{ij} < 0 \text{ for all } j \in M^0\}$$

where $\mu_{ij} = \mathbb{E}[d_{ijt}]$. The MCS algorithm determines M^* through a sequence of significance tests, eliminating models that are significantly inferior.
The hypotheses tested are of the form:

$$H_0^M : \mu_{ij} = 0 \text{ for all } i, j \in M$$

where $M \subseteq M^0$, against the alternative $H_A^M : \mu_{ij} \neq 0$ for some $i, j \in M$. The MCS procedure aims to construct a subset $M_\alpha^* \subseteq M^0$ that contains all of M^* with a pre-specified coverage probability

$1-\alpha$, based on an equivalence test θ_M and an elimination rule e_M. The equivalence test θ_M tests H_0^M for different M. The elimination rule e_M removes models statistically inferior, reducing M until it only contains indistinguishable models in terms of performance. The test statistic for this test is the well-known test used in Diebold and Mariano [54] :

$$t_{ij} = \frac{\mu_{ij}}{\sqrt{var(d_{ij})}}$$

Where $var(d_{ij})$ is the estimate variance of d_{ij}.

The final set M_α^* is a model confidence set containing the best-performing models with high probability 1 - α , offering a statistically robust selection method. In other words, the MCS contains all superior models with a given probability (its coverage probability) under certain assumptions. Bootstrap methods are often employed for practical implementation of the MCS procedure, especially when the number of models is large.

Multivariate Giacomini-White The Multivariate Giacomini-White test extends the framework of Giacomini and White (2006) to a multivariate setting for real-time assessment and identification of state-dependencies in predictability. This test compares the predictive ability of multiple forecasting methods under varying conditions.
Consider a setting with $p+1$ forecasting methods ($p \geq 1$), indexed by $i = 1, \ldots, p+1$. The forecast of the target variable $y_{t+\tau}$ at time t using the i-th method is denoted as:

$$f_{ti+\tau} = \alpha_{ti} + \beta_{it} x_{it}$$

where α_{ti} and β_{it} are parameters estimated from the data, and x_{it} is the vector of predictors. A rolling window forecasting scheme is assumed for estimation.
The Multivariate Giacomini-White test statistic assesses whether these $p+1$ methods have equal predictive ability. It does this by evaluating the forecasting performance of each method against a loss function $L_{t+\tau}$, which measures the prediction error for the forecast $f_{ti+\tau}$. The test can be seen as a generalization of the Diebold-Mariano test that measures the Conditional Predictive Ability instead of the Unconditional Predictive Ability. The test, like the Diebold-Mariano variant, measures the statistical significance of the differences of two models forecasts. It is an asymptotic 2 test.
The test statistic in the Multivariate Giacomini-White test plays a crucial role in determining the statistical significance of differences in predictive abilities across various forecasting methods.

- **Loss Function**: A loss function L is defined to quantify the prediction error of each forecast. This could be mean squared error, absolute error, or any other metric appropriate for the analysis.
- **Forecast Error**: The forecast error at time t for the horizon τ is the difference between the forecast $f_{ti+\tau}$ and the actual observed value $y_{t+\tau}$.
- **Loss Differential**: The loss differential for methods i and j at time t is computed as $d_{ijt} = L(y_{t+\tau}, f_{ti+\tau}) - L(y_{t+\tau}, f_{tj+\tau})$.
- **Test Statistic Computation**: Aggregate these differentials over time to form the test statistic. This involves creating a vector of loss differentials and using it to form a multivariate statistic.

- **Statistical Significance**: Perform hypothesis testing using the test statistic to determine if there is a statistically significant difference in predictive performance between the forecasting methods.

List of Tables

Table 1: Hyperparameters tested for each architecture

Model	Hyperparameter	Tested values
Transformer	Attention Heads	2 – 4 - 8
	Hidden dimension size	1 – 4 – 8 – 16 – 32 – 64 – 128 – 256 – 512 - 1024
	Query, Key and Value Dimension	8 – 16 – 32 – 64
	Activation functions	ReLu, tanH, sigmoid
LSTM	Hidden Size	1 – 2 – 4 – 8
	Input Feed Forward dimensions	1 – 4 – 8 – 16 – 32 – 64 – 128 – 256 – 512 - 1024
	LSTM layers	1 – 2 – 4 – 8
	Activation functions	ReLu, tanH, sigmoid
XGBOOST	max_depth	2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10
	max_leaf_nodes	2 - 4 - 8 - 16 - 32
	num_estimators	50 - 60 - 70 - 80 - 100 - 200 - 150 - 500
	base learner	linear regressor, tree
	criterion	gini, entropy, log_loss
NGBOOST	Gradient	Natural gradient - Ordinary Gradient
	Distribution	Normal - Student T
	num_estimators	50 - 60 - 70 - 80 - 100 - 200 - 150 - 500
	base learner	linear regressor, tree
	criterion	gini, entropy, log_loss

This table shows us all the hyperparameters tested for every architecture studied. The first column displays the architectures in each cell, the second column shows the hyperparameters tested.

Table 2: Training Hyperparameters tested

Hyperparameter	Tested values
Learning Rate	0.0001 – 0.001 – 0.01 – 0.1 – 1
Gradient Clipping value	0.5 – 1 – 2 - 5 – 10 – 25 – 50
Batch size	16 – 32 – 64 - 128
Optimizers	Adam, SGD
Scheduler's step size	1 – 3 – 5
L1 Weight Regularization	0 - 0.01
L2 Weight Regularization	0 - 0.01
Weight_decay	0.0001 - 0.000001
Num Epochs	100 - 200 - 500 - 1000 - 5000
Input Sequence Length	5 - 10 - 20 -30
Optimization Algorithm	Adaptive Moment Estimation - Stochastic Gradient Descent

This table shows us the training hyperparameters tested for each combination of hyperparameters in Table 1

Table 3: Final hyperparameters retained for A-60 models

Model	Hyperparameter	S&P500	NASDAQ	RUSSEL	GOLD	OIL
Transformer	Attention Heads	8	8	8	4	4
	Hidden dimension size	256	256	128	128	128
	Query, Key and Value Dimension	32	32	32	32	32
	Activation functions	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid
LSTM	Hidden Size	8	8	8	4	4
	Input Feed Forward dimensions	1	1	1	1	1
	LSTM layers	1	1	1	1	1
	Activation functions	ReLu	ReLu	ReLu	ReLu	ReLu
XGBOOST	max_depth	5	5	5	5	5
	max_leaf_nodes	8	8	8	8	8
	num_estimators	500	500	500	500	500
	base learner	Tree	Tree	Tree	Tree	Tree
	criterion	Entropy	Entropy	Entropy	Entropy	Entropy
NGBOOST	Gradient	4	4	4	4	4
	Distribution	Student T	Student T	Student T	Student T	Student T
	num_estimators	500	500	500	500	500
	base learner	Tree	Tree	Tree	Tree	Tree
	criterion	gini	gini	gini	gini	gini

Table 4: **MGW Superior predictive ability test p-value against benchmark models**

		S&P500		NASDAQ		RUSSEL		GOLD		OIL	
		HAR	CGARCH	HAR	CGARCH	HAR	CGARCH	HAR	CGARCH	HAR	CGARCH
Horizon											
h-5	**XgBoost**	0.002	0.000	0.002	0.000	0.173	0.327	0.868	0.000	0.578	0.000
	NgBoost	0.962	0.005	0.217	0.683	0.007	0.010	0.868	0.000	0.159	0.018
	LSTM	0.628	0.000	0.008	0.008	0.650	0.170	0.000	0.000	0.305	0.000
	Transformer	0.891	0.001	0.660	0.275	0.625	0.471	0.003	0.000	0.838	0.000
	X-N-L-T	0.000	0.013	0.534	0.849	0.778	0.866	0.044	0.000	0.000	0.859
	XG CGARCH	0.000	0.000	0.001	0.009	0.007	0.006	0.004	0.000	0.005	0.009
	NG CGARCH	0.000	0.000	0.009	0.005	0.003	0.008	0.002	0.000	0.003	0.000
	LSTM CGARCH	0.000	0.000	0.317	0.819	0.395	0.551	0.000	0.364	0.145	0.000
	TRANS CGARCH	0.000	0.000	0.010	0.074	0.035	0.110	0.060	0.000	0.021	0.002
	X-N-L-T CGARCH	0.000	0.000	0.000	0.007	0.478	0.630	0.087	0.000	0.000	0.995
h-10	**XgBoost**	0.031	0.000	0.034	0.000	0.700	0.777	0.316	0.000	0.622	0.000
	NgBoost	0.307	0.000	0.567	0.078	0.542	0.692	0.001	0.183	0.070	0.000
	LSTM	0.038	0.000	0.008	0.000	0.682	0.015	0.001	0.000	0.439	0.000
	Transformer	0.160	0.000	0.063	0.000	0.322	0.031	0.001	0.000	0.555	0.000
	X-N-L-T	0.047	0.000	0.381	0.052	0.003	0.001	0.096	0.000	0.000	0.001
	XG CGARCH	0.000	0.008	0.000	0.001	0.007	0.006	0.006	0.000	0.004	0.000
	NG CGARCH	0.008	0.007	0.000	0.004	0.006	0.002	0.008	0.001	0.000	0.000
	LSTM CGARCH	0.828	0.000	0.000	0.059	0.000	0.137	0.293	0.000	0.048	0.000
	TRANS CGARCH	0.417	0.000	0.000	0.000	0.316	0.792	0.303	0.000	0.143	0.000
	X-N-L-T CGARCH	0.226	0.005	0.001	0.280	0.001	0.000	0.000	0.432	0.000	0.001
h-15	**XgBoost**	0.048	0.000	0.010	0.000	0.492	0.731	0.114	0.000	0.338	0.000
	NgBoost	0.339	0.002	0.616	0.000	0.061	0.004	0.006	0.000	0.011	0.025
	LSTM	0.022	0.000	0.236	0.002	0.185	0.042	0.000	0.000	0.755	0.000
	Transformer	0.079	0.000	0.016	0.000	0.052	0.574	0.000	0.000	0.580	0.000
	X-N-L-T	0.281	0.000	0.491	0.067	0.008	0.059	0.031	0.000	0.001	0.179
	XG CGARCH	0.001	0.000	0.005	0.000	0.009	0.005	0.006	0.000	0.000	0.006
	NG CGARCH	0.002	0.006	0.000	0.007	0.009	0.004	0.000	0.007	0.008	0.000
	LSTM CGARCH	0.000	0.000	0.941	0.000	0.000	0.000	0.012	0.000	0.662	0.000
	TRANS CGARCH	0.206	0.000	0.003	0.209	0.326	0.065	0.963	0.000	0.578	0.000
	X-N-L-T CGARCH	0.000	0.703	0.600	0.007	0.071	0.019	0.001	0.000	0.005	0.069
h-20	**XgBoost**	0.003	0.000	0.089	0.000	0.055	0.960	0.169	0.000	0.298	0.677
	NgBoost	0.785	0.000	0.292	0.006	0.109	0.004	0.019	0.000	0.140	0.292
	LSTM	0.741	0.000	0.000	0.000	0.112	0.405	0.000	0.000	0.968	0.000
	Transformer	0.027	0.000	0.029	0.000	0.827	0.011	0.000	0.000	0.091	0.000
	X-N-L-T	0.440	0.000	0.000	0.000	0.327	0.023	0.000	0.000	0.693	0.000
	XG CGARCH	0.008	0.000	0.009	0.000	0.001	0.000	0.004	0.000	0.001	0.009
	NG CGARCH	0.000	0.005	0.005	0.000	0.000	0.000	0.009	0.000	0.004	0.006
	LSTM CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.777	0.000	0.002	0.169
	TRANS CGARCH	0.060	0.000	0.000	0.000	0.000	0.000	0.825	0.000	0.405	0.000
	X-N-L-T CGARCH	0.000	0.000	0.000	0.000	0.104	0.025	0.005	0.000	0.103	0.175
h-60	**XgBoost**	0.000	0.000	0.000	0.000	0.000	0.000	0.034	0.000	0.218	0.170
	NgBoost	0.093	0.000	0.827	0.000	0.189	0.153	0.462	0.002	0.256	0.008
	LSTM	0.000	0.000	0.003	0.000	0.085	0.116	0.000	0.000	0.002	0.463
	Transformer	0.000	0.000	0.020	0.000	0.096	0.109	0.000	0.000	0.013	0.190
	X-N-L-T	0.030	0.000	0.789	0.000	0.000	0.000	0.613	0.000	0.365	0.076
	XG CGARCH	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.002
	NG CGARCH	0.006	0.000	0.000	0.000	0.002	0.000	0.000	0.000	0.000	0.000
	LSTM CGARCH	0.000	0.132	0.000	0.000	0.485	0.233	0.000	0.016	0.284	0.025
	TRANS CGARCH	0.000	0.000	0.000	0.103	0.000	0.000	0.000	0.000	0.042	0.418
	X-N-L-T CGARCH	0.982	0.000	0.000	0.829	0.000	0.000	0.000	0.000	0.000	0.000

In this table, we're able to see for each forecasting horizon (first column) and each model at each horizon (second column), what is the p-value of the statistical test for difference in predictive ability between the model and the benchmarks. With a threshold at 1%, we're able to say for example that for the h-5 horizon, the XgBoost Model has a different predictive ability than both our benchmarks for the S&P500, given that the p value are respectively 0.002 and 0.000

Table 5: **Pairwise Superior predictive ability test p-value results for the S&P500**

Horizon		HAR	CGARCH	**XgBoost**	**NgBoost**	LSTM	Transformer	X-N-L-T	XG CGARCH	NG CGARCH	LSTM CGARCH	TRANS CGARCH	XNLT CGARCH
h-5	HAR	1.000	0.000	0.002	0.962	0.628	0.891	0.000	0.000	0.000	0.000	0.000	0.000
	CGARCH	0.000	1.000	0.000	0.005	0.000	0.001	0.013	0.000	0.000	0.000	0.000	0.000
	XgBoost	0.002	0.000	1.000	**0.060**	0.030	0.019	0.000	0.000	0.000	0.000	0.000	0.000
	NgBoost	0.962	0.005	**0.060**	1.000	0.852	0.931	0.000	0.000	0.000	0.000	0.000	0.000
	LSTM	0.628	0.000	0.030	0.852	1.000	0.447	0.000	0.000	0.000	0.000	0.000	0.000
	Transformer	0.891	0.001	0.019	0.931	0.447	1.000	0.000	0.000	0.000	0.000	0.000	0.000
	X-N-L-T	0.000	0.013	0.000	0.000	0.000	0.000	1.000	0.000	0.003	0.000	0.000	0.000
	XG CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.002	0.005
	NG CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.000	1.000	0.000	0.000	0.000
	LSTM CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	TRANS CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.000	1.000	0.000
	X-N-L-T CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.005	0.000	0.000	0.000	1.000
h-10	HAR	1.000	0.000	0.031	0.307	0.038	0.160	0.047	0.000	0.004	0.828	0.417	0.226
	CGARCH	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.008	0.007	0.000	0.000	0.005
	XgBoost	0.031	0.000	1.000	0.203	0.360	0.158	0.345	0.000	0.001	0.096	0.105	0.016
	NgBoost	0.307	0.000	0.203	1.000	0.940	0.677	0.918	0.000	0.001	0.405	0.508	0.093
	LSTM	0.038	0.000	0.360	0.940	1.000	0.465	0.658	0.000	0.000	0.001	0.073	0.000
	Transformer	0.160	0.000	0.158	0.677	0.465	1.000	0.464	0.000	0.001	0.280	0.150	0.020
	X-N-L-T	0.047	0.000	0.345	0.918	0.658	0.464	1.000	0.000	0.000	0.000	0.047	0.000
	XG CGARCH	0.000	0.008	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
	NG CGARCH	0.008	0.007	0.001	0.001	0.000	0.001	0.000	0.000	1.000	0.000	0.007	0.009
	LSTM CGARCH	0.828	0.000	0.096	0.405	0.001	0.280	0.000	0.000	0.000	1.000	0.442	0.000
	TRANS CGARCH	0.417	0.000	0.105	0.508	0.073	0.150	0.047	0.000	0.007	0.442	1.000	0.016
	X-N-L-T CGARCH	0.226	0.005	0.016	0.093	0.000	0.020	0.000	0.000	0.009	0.000	0.016	1.000
h-15	HAR	1.000	0.000	0.048	0.339	0.022	0.079	0.281	0.001	0.002	0.000	0.206	0.000
	CGARCH	0.000	1.000	0.000	0.002	0.000	0.000	0.000	0.000	0.006	0.000	0.000	0.703
	XgBoost	0.048	0.000	1.000	0.026	0.321	0.288	0.013	0.006	0.000	0.000	0.018	0.000
	NgBoost	0.339	0.002	0.026	1.000	0.193	0.160	0.672	0.002	0.002	0.000	0.686	0.018
	LSTM	0.022	0.000	0.321	0.193	1.000	0.830	0.067	0.007	0.000	0.000	0.004	0.000
	Transformer	0.079	0.000	0.288	0.160	0.830	1.000	0.086	0.009	0.000	0.000	0.007	0.000
	X-N-L-T	0.281	0.000	0.013	0.672	0.067	0.086	1.000	0.006	0.006	0.000	0.939	0.001
	XG CGARCH	0.001	0.000	0.006	0.002	0.007	0.009	0.006	1.000	0.000	0.000	0.000	0.000
	NG CGARCH	0.002	0.006	0.000	0.002	0.000	0.000	0.006	0.000	1.000	0.000	0.000	0.003
	LSTM CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	TRANS CGARCH	0.206	0.000	0.018	0.686	0.004	0.007	0.939	0.000	0.000	0.000	1.000	0.005
	X-N-L-T CGARCH	0.000	0.703	0.000	0.018	0.000	0.000	0.001	0.000	0.003	0.000	0.005	1.000
h-20	HAR	1.000	0.000	0.003	0.785	0.741	0.027	0.440	0.008	0.000	0.000	0.060	0.000
	CGARCH	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.005	0.000	0.000	0.000
	XgBoost	0.003	0.000	1.000	0.001	0.025	0.409	0.010	0.007	0.000	0.000	0.728	0.000
	NgBoost	0.785	0.000	0.001	1.000	0.690	0.080	0.644	0.008	0.000	0.000	0.025	0.000
	LSTM	0.741	0.000	0.025	0.690	1.000	0.001	0.305	0.004	0.000	0.000	0.091	0.000
	Transformer	0.027	0.000	0.409	0.080	0.001	1.000	0.035	0.007	0.000	0.000	0.642	0.000
	X-N-L-T	0.440	0.000	0.010	0.644	0.305	0.035	1.000	0.009	0.003	0.000	0.007	0.000
	XG CGARCH	0.008	0.000	0.007	0.008	0.004	0.007	0.009	1.000	0.000	0.000	0.008	0.000
	NG CGARCH	0.000	0.005	0.000	0.000	0.000	0.000	0.003	0.000	1.000	0.000	0.000	0.000
	LSTM CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.015
	TRANS CGARCH	0.060	0.000	0.728	0.025	0.091	0.642	0.007	0.008	0.000	0.000	1.000	0.000
	X-N-L-T CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.015	0.000	1.000
h-60	HAR	1.000	0.000	0.000	0.093	0.000	0.000	0.030	0.001	0.006	0.000	0.000	0.982
	CGARCH	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.132	0.000	0.000
	XgBoost	0.000	0.000	1.000	0.019	0.835	0.209	0.088	0.000	0.000	0.000	0.000	0.011
	NgBoost	0.093	0.000	0.019	1.000	0.005	0.000	0.384	0.009	0.005	0.000	0.000	0.361
	LSTM	0.000	0.000	0.835	0.005	1.000	0.056	0.157	0.000	0.000	0.000	0.000	0.002
	Transformer	0.000	0.000	0.209	0.000	0.056	1.000	0.022	0.000	0.000	0.000	0.000	0.000
	X-N-L-T	0.030	0.000	0.088	0.384	0.157	0.022	1.000	0.000	0.001	0.000	0.000	0.098
	XG CGARCH	0.001	0.000	0.000	0.009	0.000	0.000	0.000	1.000	0.008	0.000	0.008	0.003
	NG CGARCH	0.006	0.000	0.000	0.005	0.000	0.000	0.001	0.008	1.000	0.000	0.000	0.007
	LSTM CGARCH	0.000	0.132	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	TRANS CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.008	0.000	0.000	1.000	0.048
	X-N-L-T CGARCH	0.982	0.000	0.011	0.361	0.002	0.000	0.098	0.003	0.007	0.000	0.048	1.000

Table 5 shows us, for the S&P500 and for each horizon, the results of the pairwise different MGW predictive ability test operated for every pair combination of our models. Any given number shows us the p-value of the different predictive ability test between the model in column and the model in row. For each horizon, this table can be read as a triangular inferior (superior) matrix, since every value repeats itself. For example, the p-value of **0.06** observed at **h-5** between **NgBoost** and **XgBoost** suggests that for this asset at the **h-5** horizon, the predictive abilities of these models are **not statistically different** from each other at a threshold of 1%.

Table 6: **Pairwise Superior predictive ability : MGW test p-value results for the NASDAQ**

Horizon		HAR	CGARCH	XgBoost	**NgBoost**	LSTM	**Transformer**	X-N-L-T	XG CGARCH	NG CGARCH	LSTM CGARCH	TRANS CGARCH	XNLT CGARCH
h-5	HAR	1.000	0.232	0.002	0.217	0.008	0.660	0.534	0.001	0.009	0.317	0.010	0.000
	CGARCH	0.232	1.000	0.000	0.683	0.008	0.275	0.849	0.009	0.005	0.819	0.074	0.007
	XgBoost	0.002	0.000	1.000	0.000	0.314	0.025	0.025	0.000	0.000	0.002	0.001	0.000
	NgBoost	0.217	0.683	0.000	1.000	0.018	0.162	0.631	0.001	0.009	0.619	0.184	0.012
	LSTM	0.008	0.008	0.314	0.018	1.000	0.149	0.007	0.000	0.000	0.000	0.000	0.000
	Transformer	0.660	0.275	0.025	0.162	0.149	1.000	0.448	0.007	0.005	0.316	0.009	0.000
	X-N-L-T	0.534	0.849	0.025	0.631	0.007	0.448	1.000	0.000	0.000	0.929	0.000	0.000
	XG CGARCH	0.001	0.009	0.000	0.001	0.000	0.007	0.000	1.000	0.008	0.000	0.009	0.001
	NG CGARCH	0.009	0.005	0.000	0.009	0.000	0.005	0.000	0.000	1.000	0.008	0.002	0.003
	LSTM CGARCH	0.317	0.819	0.002	0.619	0.000	0.316	0.929	0.000	0.008	1.000	0.006	0.000
	TRANS CGARCH	0.010	0.074	0.001	0.184	0.000	0.009	0.000	0.009	0.002	0.006	1.000	0.001
	X-N-L-T CGARCH	0.000	0.007	0.000	0.012	0.000	0.000	0.000	0.001	0.003	0.000	0.001	1.000
h-10	HAR	1.000	0.000	0.034	0.567	0.008	0.063	0.381	0.000	0.000	0.000	0.000	0.001
	CGARCH	0.000	1.000	0.000	0.078	0.000	0.000	0.052	0.001	0.004	0.059	0.000	0.280
	XgBoost	0.034	0.000	1.000	0.003	0.281	0.305	0.039	0.000	0.000	0.000	0.000	0.000
	NgBoost	0.567	0.078	0.003	1.000	0.193	0.225	0.970	0.006	0.001	0.032	0.000	0.023
	LSTM	0.008	0.000	0.281	0.193	1.000	0.974	0.010	0.000	0.000	0.000	0.000	0.000
	Transformer	0.063	0.000	0.305	0.225	0.974	1.000	0.015	0.000	0.000	0.001	0.000	0.000
	X-N-L-T	0.381	0.052	0.039	0.970	0.010	0.015	1.000	0.000	0.009	0.041	0.001	0.000
	XG CGARCH	0.000	0.001	0.000	0.006	0.000	0.000	0.000	1.000	0.002	0.005	0.008	0.004
	NG CGARCH	0.000	0.004	0.000	0.001	0.000	0.000	0.009	0.002	1.000	0.005	0.001	0.005
	LSTM CGARCH	0.000	0.059	0.000	0.032	0.000	0.001	0.041	0.005	0.005	1.000	0.000	0.461
	TRANS CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.008	0.001	0.000	1.000	0.488
	X-N-L-T CGARCH	0.001	0.280	0.000	0.023	0.000	0.000	0.000	0.004	0.005	0.461	0.488	1.000
h-15	HAR	1.000	0.000	0.010	0.616	0.236	0.016	0.491	0.005	0.000	0.941	0.003	0.600
	CGARCH	0.000	1.000	0.000	0.000	0.002	0.000	0.067	0.000	0.007	0.000	0.209	0.007
	XgBoost	0.010	0.000	1.000	0.074	0.334	0.348	0.026	0.009	0.000	0.002	0.000	0.003
	NgBoost	0.616	0.000	0.074	1.000	0.665	**0.470**	0.345	0.000	0.000	0.670	0.001	0.399
	LSTM	0.236	0.002	0.334	0.665	1.000	0.601	0.033	0.005	0.000	0.514	0.003	0.241
	Transformer	0.016	0.000	0.348	**0.470**	0.601	1.000	0.014	0.008	0.000	0.301	0.001	0.099
	X-N-L-T	0.491	0.067	0.026	0.345	0.033	0.014	1.000	0.000	0.000	0.580	0.034	0.894
	XG CGARCH	0.005	0.000	0.009	0.000	0.005	0.008	0.000	1.000	0.000	0.000	0.000	0.000
	NG CGARCH	0.000	0.007	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.004	0.006	0.008
	LSTM CGARCH	0.941	0.000	0.002	0.670	0.514	0.301	0.580	0.000	0.004	1.000	0.000	0.459
	TRANS CGARCH	0.003	0.209	0.000	0.001	0.003	0.001	0.034	0.000	0.006	0.000	1.000	0.000
	X-N-L-T CGARCH	0.600	0.007	0.003	0.399	0.241	0.099	0.894	0.000	0.008	0.459	0.000	1.000
h-20	HAR	1.000	0.000	0.089	0.292	0.000	0.029	0.000	0.009	0.005	0.000	0.000	0.000
	CGARCH	0.000	1.000	0.000	0.006	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	XgBoost	0.089	0.000	1.000	0.013	0.957	0.743	0.000	0.009	0.006	0.000	0.000	0.000
	NgBoost	0.292	0.006	0.013	1.000	0.026	0.020	0.000	0.009	0.002	0.000	0.000	0.000
	LSTM	0.000	0.000	0.957	0.026	1.000	0.605	0.000	0.000	0.003	0.000	0.000	0.000
	Transformer	0.029	0.000	0.743	0.020	0.605	1.000	0.000	0.009	0.004	0.000	0.000	0.000
	X-N-L-T	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.079	0.093	0.000
	XG CGARCH	0.009	0.000	0.009	0.009	0.000	0.009	0.000	1.000	0.005	0.000	0.000	0.000
	NG CGARCH	0.005	0.000	0.006	0.003	0.005	0.004	0.000	0.002	1.000	0.000	0.000	0.000
	LSTM CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.079	0.000	0.000	1.000	0.000	0.000
	TRANS CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.093	0.000	0.000	0.000	1.000	0.000
	X-N-L-T CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
h-60	HAR	1.000	0.000	0.000	0.827	0.003	0.020	0.789	0.000	0.000	0.000	0.000	0.000
	CGARCH	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.103	0.829
	XgBoost	0.000	0.000	1.000	0.007	0.000	0.056	0.000	0.000	0.000	0.000	0.000	0.000
	NgBoost	0.827	0.000	0.007	1.000	0.534	0.136	0.921	0.000	0.000	0.000	0.000	0.000
	LSTM	0.003	0.000	0.000	0.534	1.000	0.000	0.197	0.000	0.000	0.000	0.000	0.000
	Transformer	0.020	0.000	0.056	0.136	0.000	1.000	0.022	0.000	0.000	0.000	0.000	0.000
	X-N-L-T	0.789	0.000	0.000	0.921	0.197	0.022	1.000	0.000	0.000	0.000	0.000	0.000
	XG CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.003	0.006
	NG CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.008	0.000
	LSTM CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	TRANS CGARCH	0.000	0.103	0.000	0.000	0.000	0.000	0.000	0.003	0.008	0.000	1.000	0.005
	X-N-L-T CGARCH	0.000	0.829	0.000	0.000	0.000	0.000	0.000	0.006	0.000	0.000	0.005	1.000

Table 6 shows us, for the NASDAQ and for each horizon, the results of the pairwise different MGW predictive ability test operated for every pair combination of our models. Any given number shows us the p-value of the different predictive ability test between the model in column and the model in row. For each horizon, this table can be read as a triangular inferior (superior) matrix, since every value repeats itself. For example, the p-value of **0.47** observed at **h-15** between the Transformer and NgBoost suggests that for this asset at the **h-15** horizon, the predictive abilities of these models are not statistically different from each other at a threshold of 1%.

Table 7: **Pairwise Superior predictive ability : MGW test p-value results for the RUSSEL 2000**

Horizon		HAR	CGARCH	XgBoost	NgBoost	**LSTM**	**Transformer**	X-N-L-T	XG CGARCH	NG CGARCH	LSTM CGARCH	TRANS CGARCH	XNLT CGARCH
h-5	HAR	1.000	0.986	0.173	0.007	0.650	0.625	0.778	0.007	0.003	0.395	0.035	0.478
	CGARCH	0.986	1.000	0.327	0.010	0.170	0.471	0.866	0.006	0.008	0.551	0.110	0.630
	XgBoost	0.173	0.327	1.000	0.000	0.171	0.435	0.248	0.000	0.007	0.398	0.814	0.360
	NgBoost	0.007	0.010	0.000	1.000	0.026	0.004	0.005	0.001	0.008	0.003	0.001	0.003
	LSTM	0.650	0.170	0.171	0.026	1.000	0.163	0.451	0.006	0.006	0.216	0.037	0.266
	Transformer	0.625	0.471	0.435	0.004	0.163	1.000	0.748	0.002	0.005	0.973	0.290	0.972
	X-N-L-T	0.778	0.866	0.248	0.005	0.451	0.748	1.000	0.000	0.004	0.249	0.011	0.056
	XG CGARCH	0.007	0.006	0.000	0.001	0.006	0.002	0.000	1.000	0.001	0.003	0.000	0.000
	NG CGARCH	0.003	0.008	0.007	0.008	0.006	0.005	0.004	0.001	1.000	0.004	0.001	0.003
	LSTM CGARCH	0.395	0.551	0.398	0.003	0.216	0.973	0.249	0.003	0.004	1.000	0.000	0.721
	TRANS CGARCH	0.035	0.110	0.814	0.001	0.037	0.290	0.011	0.000	0.001	0.000	1.000	0.008
	X-N-L-T CGARCH	0.478	0.630	0.360	0.003	0.266	0.972	0.056	0.000	0.003	0.721	0.008	1.000
h-10	HAR	1.000	0.691	0.700	0.542	0.682	0.322	0.003	0.007	0.006	0.000	0.316	0.001
	CGARCH	0.691	1.000	0.777	0.692	0.015	0.031	0.001	0.006	0.002	0.137	0.792	0.000
	XgBoost	0.700	0.777	1.000	0.953	0.624	0.550	0.316	0.001	0.000	0.787	0.857	0.000
	NgBoost	0.542	0.692	0.953	1.000	0.432	0.324	0.078	0.000	0.000	0.527	0.824	0.002
	LSTM	0.682	0.015	0.624	0.432	1.000	0.587	0.026	0.008	0.000	0.052	0.418	0.000
	Transformer	0.322	0.031	0.550	0.324	0.587	1.000	0.060	0.004	0.001	0.010	0.229	0.000
	X-N-L-T	0.003	0.001	0.316	0.078	0.026	0.060	1.000	0.000	0.001	0.000	0.005	0.000
	XG CGARCH	0.007	0.006	0.001	0.000	0.008	0.004	0.000	1.000	0.004	0.006	0.009	0.001
	NG CGARCH	0.006	0.002	0.000	0.000	0.000	0.001	0.001	0.004	1.000	0.000	0.008	0.002
	LSTM CGARCH	0.000	0.137	0.787	0.527	0.052	0.010	0.000	0.006	0.000	1.000	0.000	0.003
	TRANS CGARCH	0.316	0.792	0.857	0.824	0.418	0.229	0.005	0.009	0.008	0.000	1.000	0.001
	X-N-L-T CGARCH	0.001	0.000	0.000	0.002	0.000	0.000	0.000	0.001	0.002	0.003	0.001	1.000
h-15	HAR	1.000	0.031	0.492	0.061	0.185	0.052	0.008	0.009	0.009	0.000	0.326	0.071
	CGARCH	0.031	1.000	0.731	0.004	0.042	0.574	0.059	0.005	0.004	0.000	0.065	0.019
	XgBoost	0.492	0.731	1.000	0.047	0.979	0.814	0.428	0.002	0.000	0.001	0.612	0.000
	NgBoost	0.061	0.004	0.047	1.000	0.018	0.006	0.001	0.002	0.004	0.058	0.028	0.729
	LSTM	0.185	0.042	0.979	0.018	1.000	**0.082**	0.001	0.004	0.000	0.000	0.357	0.040
	Transformer	0.052	0.574	0.814	0.006	**0.082**	1.000	0.029	0.002	0.000	0.000	0.124	0.020
	X-N-L-T	0.008	0.059	0.428	0.001	0.001	0.029	1.000	0.000	0.008	0.000	0.018	0.006
	XG CGARCH	0.009	0.005	0.002	0.002	0.004	0.002	0.000	1.000	0.009	0.004	0.003	0.002
	NG CGARCH	0.009	0.004	0.000	0.004	0.000	0.000	0.008	0.009	1.000	0.000	0.002	0.005
	LSTM CGARCH	0.000	0.000	0.001	0.058	0.000	0.000	0.000	0.004	0.000	1.000	0.000	0.204
	TRANS CGARCH	0.326	0.065	0.612	0.028	0.357	0.124	0.018	0.003	0.002	0.000	1.000	0.047
	X-N-L-T CGARCH	0.071	0.019	0.000	0.729	0.040	0.020	0.006	0.002	0.005	0.204	0.047	1.000
h-20	HAR	1.000	0.008	0.055	0.109	0.112	0.827	0.327	0.001	0.000	0.000	0.000	0.104
	CGARCH	0.008	1.000	0.960	0.004	0.405	0.011	0.023	0.000	0.000	0.000	0.000	0.025
	XgBoost	0.055	0.960	1.000	0.001	0.737	0.214	0.011	0.000	0.000	0.000	0.000	0.001
	NgBoost	0.109	0.004	0.001	1.000	0.015	0.110	0.232	0.002	0.000	0.000	0.000	0.545
	LSTM	0.112	0.405	0.737	0.015	1.000	0.009	0.118	0.001	0.000	0.000	0.000	0.042
	Transformer	0.827	0.011	0.214	0.110	0.009	1.000	0.561	0.007	0.000	0.000	0.000	0.126
	X-N-L-T	0.327	0.023	0.011	0.232	0.118	0.561	1.000	0.001	0.000	0.000	0.000	0.158
	XG CGARCH	0.001	0.000	0.000	0.002	0.001	0.007	0.001	1.000	0.000	0.000	0.000	0.001
	NG CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.001	0.000	0.000
	LSTM CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	1.000	0.000	0.000
	TRANS CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
	X-N-L-T CGARCH	0.104	0.025	0.001	0.545	0.042	0.126	0.158	0.001	0.000	0.000	0.000	1.000
h-60	HAR	1.000	0.603	0.000	0.189	0.085	0.096	0.000	0.000	0.002	0.485	0.000	0.000
	CGARCH	0.603	1.000	0.000	0.153	0.116	0.109	0.000	0.000	0.000	0.233	0.000	0.000
	XgBoost	0.000	0.000	1.000	0.000	0.180	0.733	0.023	0.000	0.000	0.000	0.000	0.000
	NgBoost	0.189	0.153	0.000	1.000	0.023	0.013	0.000	0.000	0.005	0.273	0.169	0.000
	LSTM	0.085	0.116	0.180	0.023	1.000	0.424	0.014	0.000	0.001	0.082	0.000	0.000
	Transformer	0.096	0.109	0.733	0.013	0.424	1.000	0.255	0.000	0.005	0.063	0.000	0.000
	X-N-L-T	0.000	0.000	0.023	0.000	0.014	0.255	1.000	0.000	0.000	0.000	0.000	0.000
	XG CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.009
	NG CGARCH	0.002	0.000	0.000	0.005	0.001	0.005	0.000	0.000	1.000	0.007	0.000	0.000
	LSTM CGARCH	0.485	0.233	0.000	0.273	0.082	0.063	0.000	0.000	0.007	1.000	0.000	0.000
	TRANS CGARCH	0.000	0.000	0.000	0.169	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
	X-N-L-T CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.000	0.000	0.000	1.000

Table 7 shows us, for the RUSSEL 2000 and for each horizon, the results of the pairwise different MGW predictive ability test operated for every pair combination of our models. Any given number shows us the p-value of the different predictive ability test between the model in column and the model in row. For each horizon, this table can be read as a triangular inferior (superior) matrix, since every value repeat itself. For example, the p-value of **0.082** observed at **h-15** between the **LSTM** and the **Transformer** suggest that for the RUSSEL at the **h-15** horizon, the predictive abilities of these models are **not statistically different** from each other at a threshold of 1%.

Table 8: **Pairwise Superior predictive ability : MGW test p-value results for GOLD**

Horizon		HAR	CGARCH	XgBoost	**NgBoost**	LSTM	Transformer	X-N-L-T	XG CGARCH	**NG CGARCH**	LSTM CGARCH	TRANS CGARCH	XNLT CGARCH
h-5	HAR	1.000	0.000	0.868	0.868	0.000	0.003	0.044	0.004	0.002	0.000	0.060	0.087
	CGARCH	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.364	0.000	0.000
	XgBoost	0.868	0.000	1.000	0.777	0.001	0.005	0.051	0.001	0.005	0.000	0.107	0.080
	NgBoost	0.868	0.000	0.777	1.000	0.019	0.021	0.086	0.006	0.003	0.000	0.211	0.088
	LSTM	0.000	0.000	0.001	0.019	1.000	0.664	0.360	0.000	0.007	0.000	0.037	0.548
	Transformer	0.003	0.000	0.005	0.021	0.664	1.000	0.159	0.000	0.006	0.000	0.043	0.278
	X-N-L-T	0.044	0.000	0.051	0.086	0.360	0.159	1.000	0.000	0.003	0.000	0.414	0.813
	XG CGARCH	0.004	0.000	0.001	0.006	0.000	0.000	0.000	1.000	0.000	0.000	0.001	0.000
	NG CGARCH	0.002	0.000	0.005	0.003	0.007	0.006	0.003	0.000	1.000	0.000	0.007	0.004
	LSTM CGARCH	0.000	0.364	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	TRANS CGARCH	0.060	0.000	0.107	0.211	0.037	0.043	0.414	0.001	0.007	0.000	1.000	0.475
	X-N-L-T CGARCH	0.087	0.000	0.080	0.088	0.548	0.278	0.813	0.000	0.004	0.000	0.475	1.000
h-10	HAR	1.000	0.000	0.316	0.001	0.001	0.001	0.096	0.006	0.008	0.293	0.303	0.000
	CGARCH	0.000	1.000	0.000	0.183	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.432
	XgBoost	0.316	0.000	1.000	0.000	0.011	0.002	0.189	0.001	0.003	0.474	0.476	0.000
	NgBoost	0.001	0.183	0.000	1.000	0.000	0.000	0.000	0.000	0.008	0.000	0.000	0.613
	LSTM	0.001	0.000	0.011	0.000	1.000	0.050	0.971	0.004	0.003	0.609	0.630	0.000
	Transformer	0.001	0.000	0.002	0.000	0.050	1.000	0.141	0.007	0.000	0.027	0.037	0.000
	X-N-L-T	0.096	0.000	0.189	0.000	0.971	0.141	1.000	0.008	0.000	0.407	0.400	0.000
	XG CGARCH	0.006	0.000	0.001	0.000	0.004	0.007	0.008	1.000	0.000	0.001	0.000	0.000
	NG CGARCH	0.008	0.001	0.003	0.008	0.003	0.000	0.000	0.000	1.000	0.000	0.000	0.000
	LSTM CGARCH	0.293	0.000	0.474	0.000	0.609	0.027	0.007	0.001	0.000	1.000	0.949	0.000
	TRANS CGARCH	0.303	0.000	0.476	0.000	0.630	0.037	0.400	0.000	0.000	0.949	1.000	0.000
	X-N-L-T CGARCH	0.000	0.432	0.000	0.613	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
h-15	HAR	1.000	0.000	0.114	0.006	0.000	0.000	0.031	0.006	0.000	0.012	0.963	0.001
	CGARCH	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.000	0.000	0.000
	XgBoost	0.114	0.000	1.000	0.000	0.003	0.019	0.094	0.005	0.000	0.040	0.605	0.309
	NgBoost	0.006	0.000	0.000	1.000	0.000	0.000	0.000	0.003	**0.009**	0.000	0.072	0.000
	LSTM	0.000	0.000	0.003	0.000	1.000	0.026	0.780	0.008	0.000	0.998	0.002	0.021
	Transformer	0.000	0.000	0.019	0.000	0.026	1.000	0.291	0.006	0.000	0.169	0.270	0.287
	X-N-L-T	0.031	0.000	0.094	0.000	0.780	0.291	1.000	0.000	0.000	0.517	0.000	0.206
	XG CGARCH	0.006	0.000	0.005	0.003	0.008	0.006	0.000	1.000	0.000	0.002	0.000	0.003
	NG CGARCH	0.000	0.007	0.000	**0.009**	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
	LSTM CGARCH	0.012	0.000	0.040	0.000	0.998	0.169	0.517	0.002	0.000	1.000	0.000	0.119
	TRANS CGARCH	0.963	0.000	0.605	0.072	0.002	0.270	0.000	0.000	0.000	0.000	1.000	0.384
	X-N-L-T CGARCH	0.001	0.000	0.309	0.000	0.021	0.287	0.206	0.003	0.000	0.119	0.384	1.000
h-20	HAR	1.000	0.000	0.169	0.019	0.000	0.000	0.000	0.004	0.009	0.777	0.825	0.005
	CGARCH	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	XgBoost	0.169	0.000	1.000	0.004	0.000	0.002	0.002	0.004	0.002	0.850	0.795	0.021
	NgBoost	0.019	0.000	0.004	1.000	0.000	0.000	0.000	0.004	0.003	0.037	0.044	0.000
	LSTM	0.000	0.000	0.000	0.000	1.000	0.011	0.140	0.000	0.003	0.000	0.000	0.097
	Transformer	0.000	0.000	0.002	0.000	0.011	1.000	0.037	0.008	0.009	0.248	0.220	0.268
	X-N-L-T	0.000	0.000	0.002	0.000	0.140	0.037	1.000	0.000	0.000	0.018	0.014	0.682
	XG CGARCH	0.004	0.000	0.004	0.004	0.000	0.008	0.000	1.000	0.007	0.000	0.000	0.000
	NG CGARCH	0.009	0.000	0.002	0.003	0.003	0.009	0.000	0.007	1.000	0.004	0.000	0.006
	LSTM CGARCH	0.777	0.000	0.850	0.037	0.000	0.248	0.018	0.000	0.004	1.000	0.391	0.005
	TRANS CGARCH	0.825	0.000	0.795	0.044	0.000	0.220	0.014	0.000	0.000	0.391	1.000	0.004
	X-N-L-T CGARCH	0.005	0.000	0.021	0.000	0.097	0.268	0.682	0.000	0.006	0.005	0.004	1.000
h-60	HAR	1.000	0.000	0.034	0.462	0.000	0.000	0.613	0.000	0.000	0.000	0.000	0.000
	CGARCH	0.000	1.000	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.016	0.000	0.000
	XgBoost	0.034	0.000	1.000	0.183	0.015	0.004	0.031	0.000	0.000	0.000	0.004	0.000
	NgBoost	0.462	0.002	0.183	1.000	0.019	0.020	0.540	0.007	0.000	0.000	0.007	0.000
	LSTM	0.000	0.000	0.015	0.019	1.000	0.851	0.000	0.000	0.000	0.000	0.282	0.000
	Transformer	0.000	0.000	0.004	0.020	0.851	1.000	0.000	0.000	0.000	0.000	0.201	0.000
	X-N-L-T	0.613	0.000	0.031	0.540	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
	XG CGARCH	0.000	0.000	0.000	0.007	0.000	0.000	0.000	1.000	**0.013**	0.000	0.000	0.000
	NG CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.013**	1.000	0.000	0.000	0.000
	LSTM CGARCH	0.000	0.016	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	TRANS CGARCH	0.000	0.000	0.004	0.007	0.282	0.201	0.000	0.000	0.000	0.000	1.000	0.000
	X-N-L-T CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000

Table 8 shows us, for GOLD and for each horizon, the results of the pairwise different MGW predictive ability test operated for every pair combination of our models. Any given number shows us the p-value of the different predictive ability test between the model in column and the model in row. For each horizon, this table can be read as a triangular inferior (superior) matrix, since every value repeat itself. For example, the p-value of **0.009** observed at **h-15** between the **NG CGARCH** and the **NgBoost** suggest that for gold at **h-15**, the predictive abilities of these models **are statistically different** from each other at a threshold of 1%. It is noteworthy that this is the first and one of only two assets for which the CGARCH boosting models are not differentiable from one another, as we can see from the **0.013** p-value at **h-60**

Table 9: **Pairwise Superior predictive ability : MGW test p-value results for OIL**

Horizon		HAR	CGARCH	**XgBoost**	**NgBoost**	LSTM	Transformer	X-N-L-T	XG CGARCH	NG CGARCH	LSTM CGARCH	TRANS CGARCH	XNLT CGARCH
h-5	HAR	1.000	0.000	0.578	0.159	0.305	0.838	0.000	0.005	0.303	0.145	0.021	0.000
	CGARCH	0.000	1.000	0.000	0.018	0.000	0.000	0.859	0.009	0.000	0.000	0.002	0.995
	XgBoost	0.578	0.000	1.000	0.226	0.280	0.696	0.000	0.004	0.004	0.565	0.110	0.000
	NgBoost	0.159	0.018	0.226	1.000	0.097	0.228	0.018	0.003	0.005	0.577	0.767	0.015
	LSTM	0.305	0.000	0.280	0.097	1.000	0.082	0.000	0.000	0.005	0.017	0.002	0.000
	Transformer	0.838	0.000	0.696	0.228	0.082	1.000	0.000	0.003	0.000	0.186	0.027	0.000
	X-N-L-T	0.000	0.859	0.000	0.018	0.000	0.000	1.000	0.003	0.000	0.000	0.000	0.571
	XG CGARCH	0.005	0.009	0.004	0.003	0.000	0.003	0.003	1.000	0.008	0.004	0.006	0.002
	NG CGARCH	0.003	0.000	0.004	0.005	0.005	0.000	0.000	0.008	1.000	0.002	0.005	0.000
	LSTM CGARCH	0.145	0.000	0.565	0.577	0.017	0.186	0.000	0.004	0.002	1.000	0.067	0.001
	TRANS CGARCH	0.021	0.002	0.110	0.767	0.002	0.027	0.000	0.006	0.005	0.067	1.000	0.000
	X-N-L-T CGARCH	0.000	0.995	0.000	0.015	0.000	0.000	0.571	0.002	0.000	0.001	0.000	1.000
h-10	HAR	1.000	0.000	0.622	0.070	0.439	0.555	0.000	0.004	0.000	0.048	0.143	0.000
	CGARCH	0.000	1.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.001
	XgBoost	0.622	0.000	1.000	0.016	0.453	0.474	0.036	0.009	0.006	0.105	0.224	0.010
	NgBoost	0.070	0.000	0.016	1.000	0.217	0.189	0.649	0.003	0.001	0.997	0.591	0.211
	LSTM	0.439	0.000	0.453	0.217	1.000	0.713	0.019	0.001	0.006	0.015	0.066	0.000
	Transformer	0.555	0.000	0.474	0.189	0.713	1.000	0.016	0.007	0.084	0.014	0.050	0.000
	X-N-L-T	0.000	0.001	0.036	0.649	0.019	0.016	1.000	0.004	0.000	0.599	0.199	0.086
	XG CGARCH	0.004	0.000	0.009	0.003	0.001	0.007	0.004	1.000	0.001	0.000	0.004	0.001
	NG CGARCH	0.000	0.000	0.006	0.001	0.006	0.004	0.000	0.001	1.000	0.006	0.000	0.000
	LSTM CGARCH	0.048	0.000	0.105	0.997	0.015	0.014	0.599	0.830	0.016	1.000	0.007	0.024
	TRANS CGARCH	0.143	0.000	0.224	0.591	0.066	0.050	0.199	0.004	0.000	0.007	1.000	0.001
	X-N-L-T CGARCH	0.000	0.001	0.010	0.211	0.000	0.000	0.086	0.001	0.000	0.024	0.001	1.000
h-15	HAR	1.000	0.000	0.338	0.011	0.755	0.580	0.001	0.000	0.008	0.662	0.578	0.005
	CGARCH	0.000	1.000	0.000	0.025	0.000	0.000	0.179	0.006	0.000	0.000	0.000	0.069
	XgBoost	0.338	0.000	1.000	0.001	0.356	0.628	0.008	0.000	0.007	0.338	0.571	0.019
	NgBoost	0.011	0.025	0.001	1.000	0.028	0.007	0.386	0.001	0.004	0.086	0.005	0.635
	LSTM	0.755	0.000	0.356	0.028	1.000	0.338	0.001	0.000	0.001	0.697	0.226	0.006
	Transformer	0.580	0.000	0.628	0.007	0.338	1.000	0.000	0.000	0.006	0.394	0.965	0.001
	X-N-L-T	0.001	0.179	0.008	0.386	0.001	0.000	1.000	0.009	0.007	0.008	0.001	0.002
	XG CGARCH	0.000	0.006	0.000	0.001	0.000	0.000	0.009	1.000	0.000	0.003	0.000	0.004
	NG CGARCH	0.008	0.000	0.007	0.004	0.001	0.006	0.007	0.000	1.000	0.008	0.007	0.004
	LSTM CGARCH	0.662	0.000	0.338	0.086	0.697	0.394	0.008	0.003	0.008	1.000	0.400	0.020
	TRANS CGARCH	0.578	0.000	0.571	0.005	0.226	0.965	0.001	0.000	0.007	0.400	1.000	0.003
	X-N-L-T CGARCH	0.005	0.069	0.019	0.635	0.006	0.001	0.002	0.004	0.004	0.020	0.003	1.000
h-20	HAR	1.000	0.001	0.298	0.140	0.968	0.091	0.693	0.001	0.004	0.002	0.405	0.103
	CGARCH	0.001	1.000	0.677	0.292	0.000	0.000	0.000	0.009	0.006	0.169	0.000	0.175
	XgBoost	0.298	0.677	1.000	**0.114**	0.315	0.236	0.265	0.005	0.006	0.791	0.463	0.672
	NgBoost	0.140	0.292	**0.114**	1.000	0.141	0.129	0.134	0.005	0.006	0.214	0.162	0.198
	LSTM	0.968	0.000	0.315	0.141	1.000	0.448	0.131	0.001	0.005	0.019	0.051	0.172
	Transformer	0.091	0.000	0.236	0.129	0.448	1.000	0.787	0.000	0.003	0.000	0.166	0.051
	X-N-L-T	0.693	0.000	0.265	0.134	0.131	0.787	1.000	0.002	0.007	0.009	0.016	0.089
	XG CGARCH	0.001	0.009	0.005	0.005	0.001	0.000	0.312	1.000	0.005	0.006	0.001	0.009
	NG CGARCH	0.004	0.006	0.006	0.006	0.005	0.003	0.007	0.005	1.000	0.000	0.004	0.001
	LSTM CGARCH	0.002	0.169	0.791	0.214	0.019	0.000	0.009	0.006	0.000	1.000	0.230	0.697
	TRANS CGARCH	0.405	0.000	0.463	0.162	0.051	0.166	0.016	0.001	0.004	0.230	1.000	0.491
	X-N-L-T CGARCH	0.103	0.175	0.672	0.198	0.172	0.051	0.089	0.009	0.001	0.697	0.491	1.000
h-60	HAR	1.000	0.047	0.218	0.256	0.002	0.013	0.365	0.009	0.000	0.284	0.042	0.000
	CGARCH	0.047	1.000	0.170	0.008	0.463	0.190	0.076	0.002	0.000	0.025	0.418	0.000
	XgBoost	0.218	0.170	1.000	0.009	0.232	0.744	0.655	0.001	0.000	0.050	0.117	0.000
	NgBoost	0.256	0.008	0.009	1.000	0.006	0.032	0.140	0.004	0.000	0.818	0.004	0.000
	LSTM	0.002	0.463	0.232	0.006	1.000	0.077	0.028	0.009	0.000	0.016	0.343	0.000
	Transformer	0.013	0.190	0.744	0.032	0.077	1.000	0.319	0.009	0.000	0.058	0.157	0.000
	X-N-L-T	0.365	0.076	0.655	0.140	0.028	0.319	1.000	0.001	0.000	0.164	0.056	0.000
	XG CGARCH	0.009	0.002	0.001	0.004	0.009	0.009	0.001	1.000	0.044	0.004	0.004	0.000
	NG CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.044	1.000	0.000	0.000	0.004
	LSTM CGARCH	0.284	0.025	0.050	0.818	0.016	0.058	0.164	0.004	0.000	1.000	0.015	0.000
	TRANS CGARCH	0.042	0.418	0.117	0.004	0.343	0.157	0.056	0.004	0.000	0.015	1.000	0.000
	X-N-L-T CGARCH	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.000	0.000	1.000

Table 9 shows us, for crude OIL and for each horizon, the results of the pairwise different MGW predictive ability test operated for every pair combination of our models. Any given number shows us the p-value of the different predictive ability test between the model in column and the model in row. For each horizon, this table can be read as a triangular inferior (superior) matrix, since every value repeat itself. For example, the p-value of **0.114** observed at **h-20** between the **XgBoost** and the **NgBoost** suggests that for OIL at **h-20**, the predictive abilities of these models are **not statistically different** from each other at a threshold of 1%.

Oil is the second asset for which the CGARCH boosting models are not differentiable from one another, as we can see from the **0.044** p-value at **h-60**

Table 10: **Model Confidence set best models at 99.9% confidence level**

	S&P500	**NASDAQ**	**RUSSEL 2000**	**GOLD**	**OIL**
h-5	HAR XgBoost NgBoost LSTM Transformer NG CGARCH	HAR XgBoost LSTM Transformer X-N-L-T NG CGARCH	CGARCH XgBoost Transformer X-N-L-T XG CGARCH NG CGARCH LSTM CGARCH TRANS CGARCH X-N-L-T CGARCH	LSTM Transformer X-N-L-T NG CGARCH TRANS CGARCH X-N-L-T CGARCH	HAR XgBoost NgBoost LSTM Transformer NG CGARCH LSTM CGARCH
h-10	XgBoost NgBoost LSTM Transformer X-N-L-T NG CGARCH	HAR XgBoost LSTM Transformer X-N-L-T NG CGARCH	HAR CGARCH LSTM Transformer X-N-L-T NG CGARCH	LSTM Transformer X-N-L-T XG CGARCH NG CGARCH	HAR XgBoost NgBoost LSTM Transformer XG CGARCH NG CGARCH TRANS CGARCH
h-15	HAR XgBoost LSTM Transformer XG CGARCH	XgBoost NgBoost LSTM Transformer XG CGARCH NG CGARCH	CGARCH XgBoost LSTM Transformer X-N-L-T NG CGARCH TRANS CGARCH	LSTM Transformer X-N-L-T NG CGARCH LSTM CGARCH	HAR XgBoost NgBoost LSTM Transformer NG CGARCH LSTM CGARCH TRANS CGARCH X-N-L-T CGARCH
h-20	HAR XgBoost NgBoost LSTM Transformer XG CGARCH NG CGARCH TRANS CGARCH	HAR XgBoost NgBoost LSTM Transformer XG CGARCH NG CGARCH	HAR CGARCH XgBoost LSTM Transformer X-N-L-T NG CGARCH	LSTM Transformer X-N-L-T NG CGARCH X-N-L-T CGARCH	HAR LSTM Transformer X-N-L-T NG CGARCH
h-60	XgBoost NgBoost LSTM Transformer X-N-L-T NG CGARCH	HAR XgBoost NgBoost Transformer X-N-L-T NG CGARCH	XgBoost LSTM Transformer X-N-L-T NG CGARCH	XgBoost LSTM Transformer TRANS CGARCH NG CGARCH	HAR CGARCH XgBoost LSTM Transformer X-N-L-T XG CGARCH NG CGARCH LSTM CGARCH TRANS CGARCH

Table 10 shows us, for every **[A-H]** problem ,the set of models for which the predictive abilities are not only the best, but also statistically indistinguishable from each other with a **99%** confidence level according to the MCS test. For example, the MCS results tells us that **for forecasting the 15 days ahead realized volatility of GOLD, the subset of best models with identical predictive ability is composed of the LSTM, the Transformer, the X-N-L-T, the NG GARCH, and the LSTM GARCH** with a 99% confidence level

Table 11: **Models Out of sample Forecasting Results Across Different Horizons for S&P500**

Horizon	Model	MAE	RMSE	Q-LIKE	OF
h-5	**HAR**	0.003	0.004	13.683	0.535
	CGARCH	0.004	0.004	33.011	0.782
	XgBoost	0.002	0.003	10.698	0.624
	NgBoost	0.003	0.004	16.470	0.545
	LSTM	0.003	0.004	11.834	0.485
	Transformer	0.003	0.004	11.723	0.475
	X-N-L-T	0.005	0.006	45.813	0.574
	XG CGARCH	0.010	0.011	51.515	0.000
	NG CGARCH	0.005	0.005	50.840	0.802
	LSTM CGARCH	0.015	0.016	216.194	0.980
	TRANS CGARCH	0.007	0.008	83.730	0.931
	X-N-L-T CGARCH	0.010	0.011	44.131	0.000
h-10	**HAR**	0.003	0.003	14.377	0.554
	CGARCH	0.004	0.004	37.633	0.802
	XgBoost	0.002	0.003	11.923	0.614
	NgBoost	0.002	0.003	13.046	0.574
	LSTM	0.002	0.003	11.145	0.465
	Transformer	0.002	0.003	14.181	0.624
	X-N-L-T	0.002	0.003	11.419	0.465
	XG CGARCH	0.004	0.006	45.488	0.356
	NG CGARCH	0.003	0.004	17.475	0.317
	LSTM CGARCH	0.003	0.004	14.093	0.426
	TRANS CGARCH	0.002	0.003	14.339	0.554
	X-N-L-T CGARCH	0.003	0.004	16.591	0.297
h-15	**HAR**	0.003	0.003	14.980	0.610
	CGARCH	0.004	0.005	43.224	0.840
	XgBoost	0.002	0.003	12.795	0.670
	NgBoost	0.003	0.005	21.864	0.660
	LSTM	0.002	0.003	12.761	0.520
	Transformer	0.002	0.003	14.505	0.670
	X-N-L-T	0.003	0.004	28.931	0.300
	XG CGARCH	0.002	0.004	13.840	0.400
	NG CGARCH	0.003	0.005	36.027	0.280
	LSTM CGARCH	0.012	0.014	56.262	0.030
	TRANS CGARCH	0.003	0.004	18.393	0.470
	X-N-L-T CGARCH	0.004	0.005	36.494	0.850
h-20	**HAR**	0.002	0.003	15.065	0.636
	CGARCH	0.004	0.005	45.857	0.869
	XgBoost	0.002	0.003	12.334	0.707
	NgBoost	0.002	0.003	16.951	0.636
	LSTM	0.002	0.003	12.905	0.535
	Transformer	0.002	0.003	11.677	0.545
	X-N-L-T	0.003	0.003	23.480	0.253
	XG CGARCH	0.002	0.003	16.608	0.788
	NG CGARCH	0.004	0.004	34.111	0.899
	LSTM CGARCH	0.007	0.008	68.569	0.869
	TRANS CGARCH	0.002	0.003	12.390	0.505
	X-N-L-T CGARCH	0.008	0.010	80.327	0.535
h-60	**HAR**	0.003	0.004	37.303	0.849
	CGARCH	0.007	0.008	110.415	0.968
	XgBoost	0.003	0.003	22.068	0.796
	NgBoost	0.003	0.004	30.694	0.742
	LSTM	0.002	0.003	22.781	0.645
	Transformer	0.002	0.003	19.432	0.677
	X-N-L-T	0.003	0.004	25.788	0.742
	XG CGARCH	0.004	0.005	54.237	0.634
	NG CGARCH	0.004	0.004	42.758	0.882
	LSTM CGARCH	0.007	0.007	83.981	0.946
	TRANS CGARCH	0.004	0.005	49.545	0.946
	X-N-L-T CGARCH	0.003	0.004	44.154	0.473

This table shows us, for the S&P500, the out of sample performance metrics scored by our models for every forecasting horizon. We made the choice to use the Mean-Absolute Error, the Root-Mean Square Error, the Quasi Likelihood, and the Overvaluation Frequency.

Table 12: **Models Out of sample Forecasting Results Across Different Horizons for the NASDAQ**

Horizon	Model	MAE	RMSE	Q-LIKE	OF
h-5	**HAR**	0.003	0.004	8.615	0.515
	CGARCH	0.003	0.004	11.587	0.663
	XgBoost	0.002	0.003	7.544	0.515
	NgBoost	0.003	0.004	10.945	0.545
	LSTM	0.003	0.003	7.266	0.436
	Transformer	0.003	0.004	7.896	0.495
	X-N-L-T	0.003	0.004	10.392	0.545
	XG CGARCH	0.003	0.004	13.008	0.584
	NG CGARCH	0.003	0.004	13.115	0.554
	LSTM CGARCH	0.003	0.004	10.182	0.564
	TRANS CGARCH	0.003	0.005	21.648	0.495
	X-N-L-T CGARCH	0.004	0.006	28.784	0.257
h-10	**HAR**	0.003	0.004	9.254	0.545
	CGARCH	0.003	0.004	13.477	0.693
	XgBoost	0.002	0.003	7.757	0.525
	NgBoost	0.003	0.004	10.960	0.495
	LSTM	0.003	0.004	8.323	0.485
	Transformer	0.003	0.004	7.867	0.475
	X-N-L-T	0.003	0.004	9.921	0.465
	XG CGARCH	0.004	0.004	15.521	0.673
	NG CGARCH	0.003	0.004	16.034	0.733
	LSTM CGARCH	0.003	0.004	14.529	0.713
	TRANS CGARCH	0.004	0.004	19.769	0.772
	X-N-L-T CGARCH	0.004	0.005	19.248	0.376
h-15	**HAR**	0.003	0.004	9.370	0.590
	CGARCH	0.003	0.004	14.890	0.740
	XgBoost	0.002	0.003	7.349	0.530
	NgBoost	0.003	0.003	8.671	0.560
	LSTM	0.003	0.004	7.862	0.400
	Transformer	0.002	0.004	7.751	0.470
	X-N-L-T	0.003	0.004	10.732	0.350
	XG CGARCH	0.002	0.003	6.942	0.550
	NG CGARCH	0.004	0.005	19.563	0.460
	LSTM CGARCH	0.003	0.003	9.987	0.710
	TRANS CGARCH	0.004	0.005	20.420	0.770
	X-N-L-T CGARCH	0.003	0.003	10.865	0.510
h-20	**HAR**	0.003	0.003	8.946	0.626
	CGARCH	0.003	0.004	15.720	0.788
	XgBoost	0.002	0.003	7.861	0.586
	NgBoost	0.003	0.004	10.564	0.545
	LSTM	0.002	0.003	7.808	0.616
	Transformer	0.002	0.003	7.073	0.495
	X-N-L-T	0.005	0.006	77.399	0.010
	XG CGARCH	0.002	0.003	7.673	0.596
	NG CGARCH	0.002	0.003	6.984	0.576
	LSTM CGARCH	0.006	0.007	41.049	0.949
	TRANS CGARCH	0.004	0.005	25.019	0.879
	X-N-L-T CGARCH	0.011	0.012	84.010	0.980
h-60	**HAR**	0.003	0.004	13.492	0.720
	CGARCH	0.005	0.006	35.024	0.914
	XgBoost	0.002	0.003	9.211	0.613
	NgBoost	0.003	0.004	14.043	0.581
	LSTM	0.003	0.004	14.472	0.688
	Transformer	0.003	0.003	10.450	0.559
	X-N-L-T	0.003	0.004	14.131	0.731
	XG CGARCH	0.005	0.005	30.935	0.892
	NG CGARCH	0.006	0.006	38.552	0.957
	LSTM CGARCH	0.011	0.011	82.745	0.989
	TRANS CGARCH	0.005	0.006	62.047	0.355
	X-N-L-T CGARCH	0.005	0.007	62.785	0.129

This table shows us, for the NASDAQ, the out of sample performance metrics scored by our models for every forecasting horizon. We made the choice to use the Mean-Absolute Error, the Root-Mean Square Error, the Quasi Likelihood, and the Overvaluation Frequency.

Table 13: **Models Out of sample Forecasting Results Across Different Horizons for the RUSSEL 2000**

Horizon	Model	MAE	RMSE	Q-LIKE	OF
h-5	**HAR**	0.004	0.006	10.366	0.614
	CGARCH	0.004	0.006	9.222	0.376
	XgBoost	0.004	0.005	8.303	0.644
	NgBoost	0.005	0.007	14.652	0.574
	LSTM	0.004	0.006	9.731	0.386
	Transformer	0.004	0.006	8.702	0.406
	X-N-L-T	0.004	0.006	9.442	0.574
	XG CGARCH	0.004	0.006	8.589	0.535
	NG CGARCH	0.004	0.006	12.268	0.723
	LSTM CGARCH	0.004	0.006	9.235	0.545
	TRANS CGARCH	0.004	0.006	8.827	0.525
	X-N-L-T CGARCH	0.004	0.006	9.207	0.554
h-10	**HAR**	0.004	0.006	10.100	0.584
	CGARCH	0.004	0.006	9.535	0.366
	XgBoost	0.004	0.009	12.020	0.614
	NgBoost	0.004	0.006	12.379	0.485
	LSTM	0.004	0.006	8.637	0.376
	Transformer	0.004	0.006	8.762	0.347
	X-N-L-T	0.004	0.006	7.665	0.505
	XG CGARCH	0.005	0.008	15.456	0.663
	NG CGARCH	0.004	0.006	11.802	0.663
	LSTM CGARCH	0.004	0.006	12.943	0.673
	TRANS CGARCH	0.004	0.006	11.002	0.693
	X-N-L-T CGARCH	0.008	0.015	52.045	0.129
h-15	**HAR**	0.004	0.006	10.124	0.600
	CGARCH	0.004	0.006	8.418	0.420
	XgBoost	0.004	0.006	9.615	0.630
	NgBoost	0.005	0.006	13.878	0.580
	LSTM	0.004	0.006	8.576	0.420
	Transformer	0.004	0.006	8.191	0.420
	X-N-L-T	0.004	0.006	7.305	0.440
	XG CGARCH	0.004	0.005	11.241	0.770
	NG CGARCH	0.004	0.006	9.285	0.600
	LSTM CGARCH	0.005	0.008	28.953	0.230
	TRANS CGARCH	0.004	0.006	9.886	0.630
	X-N-L-T CGARCH	0.005	0.006	13.847	0.790
h-20	**HAR**	0.004	0.005	9.122	0.636
	CGARCH	0.003	0.005	7.323	0.475
	XgBoost	0.003	0.005	8.238	0.667
	NgBoost	0.004	0.006	13.212	0.586
	LSTM	0.003	0.005	7.264	0.414
	Transformer	0.004	0.006	8.428	0.333
	X-N-L-T	0.004	0.005	10.265	0.758
	XG CGARCH	0.005	0.006	18.856	0.838
	NG CGARCH	0.007	0.008	27.514	0.889
	LSTM CGARCH	0.007	0.008	27.382	0.879
	TRANS CGARCH	0.008	0.008	31.631	0.909
	X-N-L-T CGARCH	0.004	0.006	15.469	0.657
h-60	**HAR**	0.004	0.005	14.831	0.753
	CGARCH	0.004	0.005	13.787	0.731
	XgBoost	0.003	0.004	8.463	0.720
	NgBoost	0.005	0.007	18.106	0.602
	LSTM	0.004	0.005	10.656	0.505
	Transformer	0.003	0.005	7.574	0.452
	X-N-L-T	0.003	0.004	7.117	0.645
	XG CGARCH	0.007	0.007	28.240	0.935
	NG CGARCH	0.004	0.005	13.187	0.742
	LSTM CGARCH	0.004	0.005	13.787	0.849
	TRANS CGARCH	0.005	0.006	18.649	0.925
	X-N-L-T CGARCH	0.007	0.008	28.713	0.925

This table shows us, for the RUSSEL 2000, the out of sample performance metrics scored by our models for every forecasting horizon. We made the choice to use the Mean-Absolute Error, the Root-Mean Square Error, the Quasi Likelihood, and the Overvaluation Frequency.

Table 14: **Models Out of sample Forecasting Results Across Different Horizons for GOLD**

Horizon	Model	MAE	RMSE	Q-LIKE	OF
h-5	**HAR**	0.002	0.003	10.772	0.747
	CGARCH	0.004	0.004	28.748	0.905
	XgBoost	0.002	0.003	10.475	0.705
	NgBoost	0.002	0.003	11.075	0.653
	LSTM	0.002	0.003	7.012	0.589
	Transformer	0.002	0.003	6.114	0.432
	X-N-L-T	0.002	0.003	7.302	0.505
	XG CGARCH	0.002	0.004	16.496	0.211
	NG CGARCH	0.002	0.003	8.767	0.558
	LSTM CGARCH	0.003	0.004	45.181	0.074
	TRANS CGARCH	0.002	0.003	8.826	0.653
	X-N-L-T CGARCH	0.002	0.003	7.752	0.453
h-10	**HAR**	0.002	0.003	11.627	0.734
	CGARCH	0.003	0.004	25.533	0.926
	XgBoost	0.002	0.003	10.558	0.745
	NgBoost	0.003	0.004	20.210	0.670
	LSTM	0.002	0.003	8.239	0.638
	Transformer	0.002	0.003	6.710	0.489
	X-N-L-T	0.002	0.003	8.300	0.394
	XG CGARCH	0.002	0.003	6.617	0.340
	NG CGARCH	0.002	0.004	18.366	0.191
	LSTM CGARCH	0.002	0.003	8.294	0.255
	TRANS CGARCH	0.002	0.003	8.221	0.277
	X-N-L-T CGARCH	0.003	0.004	39.720	0.074
h-15	**HAR**	0.002	0.003	12.433	0.787
	CGARCH	0.004	0.004	26.958	0.936
	XgBoost	0.002	0.003	11.249	0.755
	NgBoost	0.003	0.004	16.946	0.713
	LSTM	0.002	0.003	7.192	0.553
	Transformer	0.002	0.003	10.027	0.723
	X-N-L-T	0.002	0.003	7.310	0.330
	XG CGARCH	0.002	0.003	8.955	0.213
	NG CGARCH	0.003	0.004	29.706	0.000
	LSTM CGARCH	0.002	0.003	6.804	0.394
	TRANS CGARCH	0.002	0.004	11.418	0.106
	X-N-L-T CGARCH	0.002	0.003	10.276	0.723
h-20	**HAR**	0.002	0.003	12.771	0.817
	CGARCH	0.004	0.004	28.642	0.935
	XgBoost	0.002	0.003	11.166	0.710
	NgBoost	0.003	0.004	19.566	0.710
	LSTM	0.002	0.003	6.386	0.548
	Transformer	0.002	0.003	9.479	0.710
	X-N-L-T	0.002	0.003	7.367	0.613
	XG CGARCH	0.002	0.004	15.152	0.054
	NG CGARCH	0.002	0.003	10.164	0.677
	LSTM CGARCH	0.002	0.004	10.817	0.140
	TRANS CGARCH	0.002	0.004	11.087	0.183
	X-N-L-T CGARCH	0.002	0.003	7.160	0.516
h-60	**HAR**	0.003	0.003	18.023	0.885
	CGARCH	0.006	0.006	49.740	0.954
	XgBoost	0.002	0.003	13.055	0.770
	NgBoost	0.003	0.007	16.983	0.724
	LSTM	0.002	0.003	9.638	0.655
	Transformer	0.002	0.003	9.382	0.747
	X-N-L-T	0.003	0.003	19.807	0.782
	XG CGARCH	0.004	0.005	33.717	0.828
	NG CGARCH	0.010	0.011	105.317	0.931
	LSTM CGARCH	0.007	0.008	64.579	0.931
	TRANS CGARCH	0.002	0.003	6.793	0.402
	X-N-L-T CGARCH	0.014	0.018	168.036	0.920

This table shows us, for Gold, the out of sample performance metrics scored by our models for every forecasting horizon. We made the choice to use the Mean-Absolute Error, the Root-Mean Square Error, the Quasi Likelihood, and the Overvaluation Frequency.

Table 15: Models Out of sample Forecasting Results Across Different Horizons for OIL

Horizon	Model	MAE	RMSE	Q-LIKE	OF
h-5	**HAR**	0.004	0.006	5.878	0.636
	CGARCH	0.006	0.009	13.559	0.068
	XgBoost	0.004	0.006	5.735	0.659
	NgBoost	0.005	0.007	7.879	0.591
	LSTM	0.004	0.007	5.377	0.489
	Transformer	0.004	0.007	5.760	0.534
	X-N-L-T	0.006	0.009	25.957	0.398
	XG CGARCH	0.005	0.008	11.443	0.545
	NG CGARCH	0.004	0.007	6.369	0.523
	LSTM CGARCH	0.004	0.007	6.839	0.614
	TRANS CGARCH	0.005	0.007	8.001	0.534
	X-N-L-T CGARCH	0.006	0.009	26.479	0.455
h-10	**HAR**	0.004	0.006	5.811	0.580
	CGARCH	0.006	0.009	14.127	0.057
	XgBoost	0.004	0.006	5.268	0.602
	NgBoost	0.005	0.007	7.385	0.625
	LSTM	0.004	0.007	5.934	0.409
	Transformer	0.004	0.007	5.697	0.443
	X-N-L-T	0.005	0.007	8.201	0.636
	XG CGARCH	0.005	0.007	6.944	0.443
	NG CGARCH	0.004	0.007	6.113	0.420
	LSTM CGARCH	0.005	0.007	6.572	0.352
	TRANS CGARCH	0.004	0.007	6.096	0.352
	X-N-L-T CGARCH	0.005	0.008	9.970	0.443
h-15	**HAR**	0.004	0.007	6.098	0.575
	CGARCH	0.006	0.009	12.076	0.115
	XgBoost	0.004	0.006	5.664	0.644
	NgBoost	0.005	0.007	7.654	0.563
	LSTM	0.004	0.007	5.916	0.437
	Transformer	0.004	0.007	5.777	0.517
	X-N-L-T	0.005	0.008	13.667	0.552
	XG CGARCH	0.006	0.008	12.278	0.690
	NG CGARCH	0.004	0.007	6.465	0.563
	LSTM CGARCH	0.004	0.007	6.281	0.333
	TRANS CGARCH	0.004	0.007	5.718	0.483
	X-N-L-T CGARCH	0.005	0.008	12.571	0.494
h-20	**HAR**	0.004	0.007	6.321	0.616
	CGARCH	0.005	0.008	10.241	0.116
	XgBoost	0.005	0.011	10.017	0.593
	NgBoost	0.009	0.030	24.940	0.651
	LSTM	0.004	0.007	5.843	0.419
	Transformer	0.004	0.007	5.915	0.570
	X-N-L-T	0.004	0.007	5.601	0.430
	XG CGARCH	0.005	0.007	7.608	0.535
	NG CGARCH	0.004	0.007	7.236	0.337
	LSTM CGARCH	0.005	0.007	8.168	0.570
	TRANS CGARCH	0.004	0.007	6.539	0.372
	X-N-L-T CGARCH	0.005	0.007	8.042	0.570
h-60	**HAR**	0.004	0.005	7.560	0.775
	CGARCH	0.003	0.004	4.332	0.425
	XgBoost	0.004	0.004	5.701	0.637
	NgBoost	0.004	0.006	8.446	0.600
	LSTM	0.003	0.004	5.257	0.600
	Transformer	0.004	0.004	6.018	0.688
	X-N-L-T	0.004	0.005	7.166	0.650
	XG CGARCH	0.004	0.005	8.244	0.450
	NG CGARCH	0.009	0.010	23.868	0.950
	LSTM CGARCH	0.004	0.005	8.591	0.713
	TRANS CGARCH	0.003	0.004	4.035	0.450
	X-N-L-T CGARCH	0.008	0.010	23.741	0.750

This table shows us, for Crude Oil, the out of sample performance metrics scored by our models for every forecasting horizon. We made the choice to use the Mean-Absolute Error, the Root-Mean Square Error, the Quasi Likelihood, and the Overvaluation Frequency.

Table 16: **Models In sample Forecasting Results Across Different Horizons for the S&P500**

Horizon	Model	MAE	RMSE	OF
h-5	**HAR**	0.004	0.006	0.646
	CGARCH	0.007	0.009	0.886
	XgBoost	0.003	0.005	0.668
	NgBoost	0.003	0.005	0.607
	LSTM	0.003	0.006	0.493
	Transformer	0.003	0.006	0.525
	X-N-L-T	0.008	0.017	0.446
	XG CGARCH	0.009	0.014	0.340
	NG CGARCH	0.007	0.012	0.609
	LSTM CGARCH	0.020	0.031	0.630
	TRANS CGARCH	0.010	0.016	0.616
	X-N-L-T CGARCH	0.012	0.018	0.379
h-10	**HAR**	0.004	0.007	0.659
	CGARCH	0.008	0.010	0.879
	XgBoost	0.003	0.005	0.677
	NgBoost	0.004	0.006	0.625
	LSTM	0.003	0.007	0.505
	Transformer	0.004	0.008	0.482
	X-N-L-T	0.004	0.007	0.484
	XG CGARCH	0.007	0.013	0.573
	NG CGARCH	0.004	0.008	0.518
	LSTM CGARCH	0.004	0.007	0.613
	TRANS CGARCH	0.004	0.007	0.646
	X-N-L-T CGARCH	0.004	0.008	0.491
h-15	**HAR**	0.004	0.008	0.664
	CGARCH	0.008	0.011	0.900
	XgBoost	0.003	0.005	0.670
	NgBoost	0.003	0.005	0.637
	LSTM	0.004	0.008	0.572
	Transformer	0.004	0.008	0.483
	X-N-L-T	0.005	0.008	0.174
	XG CGARCH	0.006	0.011	0.247
	NG CGARCH	0.008	0.013	0.583
	LSTM CGARCH	0.015	0.021	0.372
	TRANS CGARCH	0.009	0.013	0.679
	X-N-L-T CGARCH	0.010	0.014	0.436
h-20	**HAR**	0.004	0.008	0.668
	CGARCH	0.008	0.011	0.886
	XgBoost	0.003	0.005	0.672
	NgBoost	0.004	0.006	0.622
	LSTM	0.004	0.008	0.563
	Transformer	0.004	0.008	0.518
	X-N-L-T	0.006	0.009	0.137
	XG CGARCH	0.015	0.023	0.328
	NG CGARCH	0.021	0.032	0.339
	LSTM CGARCH	0.037	0.054	0.341
	TRANS CGARCH	0.021	0.032	0.269
	X-N-L-T CGARCH	0.017	0.023	0.419
h-60	**HAR**	0.005	0.009	0.645
	CGARCH	0.010	0.012	0.838
	XgBoost	0.004	0.006	0.660
	NgBoost	0.004	0.007	0.592
	LSTM	0.005	0.009	0.476
	Transformer	0.005	0.009	0.502
	X-N-L-T	0.008	0.013	0.291
	XG CGARCH	0.014	0.020	0.494
	NG CGARCH	0.006	0.009	0.444
	LSTM CGARCH	0.017	0.023	0.449
	TRANS CGARCH	0.009	0.012	0.432
	X-N-L-T CGARCH	0.015	0.020	0.524

This table shows us, for the S&P500, the in sample performance metrics scored by our models for every forecasting horizon.

Table 17: **Models In sample Forecasting Results Across Different Horizons for the NASDAQ**

Horizon	Model	MAE	RMSE	OF
h-5	**HAR**	0.004	0.008	0.612
	CGARCH	0.006	0.009	0.768
	XgBoost	0.004	0.006	0.586
	NgBoost	0.004	0.007	0.537
	LSTM	0.004	0.008	0.521
	Transformer	0.004	0.008	0.498
	X-N-L-T	0.005	0.009	0.451
	XG CGARCH	0.006	0.012	0.486
	NG CGARCH	0.006	0.009	0.684
	LSTM CGARCH	0.005	0.009	0.635
	TRANS CGARCH	0.006	0.010	0.618
	X-N-L-T CGARCH	0.005	0.010	0.498
h-10	**HAR**	0.005	0.008	0.614
	CGARCH	0.007	0.010	0.786
	XgBoost	0.004	0.007	0.575
	NgBoost	0.004	0.007	0.532
	LSTM	0.005	0.009	0.571
	Transformer	0.005	0.009	0.463
	X-N-L-T	0.006	0.011	0.370
	XG CGARCH	0.006	0.011	0.650
	NG CGARCH	0.006	0.010	0.725
	LSTM CGARCH	0.007	0.010	0.793
	TRANS CGARCH	0.008	0.011	0.834
	X-N-L-T CGARCH	0.006	0.011	0.473
h-15	**HAR**	0.005	0.009	0.615
	CGARCH	0.007	0.011	0.788
	XgBoost	0.004	0.007	0.570
	NgBoost	0.004	0.007	0.523
	LSTM	0.005	0.009	0.381
	Transformer	0.005	0.009	0.490
	X-N-L-T	0.006	0.011	0.214
	XG CGARCH	0.011	0.016	0.278
	NG CGARCH	0.012	0.019	0.633
	LSTM CGARCH	0.012	0.018	0.309
	TRANS CGARCH	0.019	0.028	0.328
	X-N-L-T CGARCH	0.011	0.016	0.294
h-20	**HAR**	0.005	0.009	0.605
	CGARCH	0.008	0.011	0.792
	XgBoost	0.004	0.007	0.570
	NgBoost	0.005	0.008	0.539
	LSTM	0.005	0.009	0.524
	Transformer	0.005	0.010	0.437
	X-N-L-T	0.011	0.016	0.074
	XG CGARCH	0.014	0.020	0.286
	NG CGARCH	0.012	0.017	0.273
	LSTM CGARCH	0.030	0.044	0.349
	TRANS CGARCH	0.017	0.024	0.365
	X-N-L-T CGARCH	0.015	0.020	0.583
h-60	**HAR**	0.005	0.010	0.536
	CGARCH	0.008	0.012	0.761
	XgBoost	0.004	0.008	0.519
	NgBoost	0.005	0.009	0.511
	LSTM	0.006	0.010	0.530
	Transformer	0.006	0.010	0.393
	X-N-L-T	0.006	0.012	0.395
	XG CGARCH	0.007	0.012	0.667
	NG CGARCH	0.006	0.009	0.750
	LSTM CGARCH	0.018	0.021	0.511
	TRANS CGARCH	0.018	0.024	0.502
	X-N-L-T CGARCH	0.021	0.025	0.498

This table shows us, for the NASDAQ, the in sample performance metrics scored by our models for every forecasting horizon.

Table 18: **Models In sample Forecasting Results Across Different Horizons for the RUSSEL**

Horizon	Model	MAE	RMSE	OF
h-5	**HAR**	0.008	0.021	0.702
	CGARCH	0.007	0.021	0.660
	XgBoost	0.007	0.021	0.670
	NgBoost	0.008	0.021	0.637
	LSTM	0.007	0.021	0.530
	Transformer	0.008	0.023	0.504
	X-N-L-T	0.007	0.021	0.619
	XG CGARCH	0.007	0.021	0.575
	NG CGARCH	0.009	0.021	0.782
	LSTM CGARCH	0.007	0.021	0.718
	TRANS CGARCH	0.007	0.021	0.723
	X-N-L-T CGARCH	0.007	0.021	0.658
h-10	**HAR**	0.008	0.021	0.698
	CGARCH	0.008	0.021	0.675
	XgBoost	0.007	0.020	0.666
	NgBoost	0.008	0.021	0.604
	LSTM	0.008	0.022	0.552
	Transformer	0.008	0.022	0.480
	X-N-L-T	0.008	0.022	0.452
	XG CGARCH	0.010	0.022	0.773
	NG CGARCH	0.009	0.021	0.804
	LSTM CGARCH	0.011	0.022	0.843
	TRANS CGARCH	0.008	0.021	0.680
	X-N-L-T CGARCH	0.010	0.024	0.480
h-15	**HAR**	0.008	0.022	0.710
	CGARCH	0.008	0.022	0.691
	XgBoost	0.007	0.021	0.672
	NgBoost	0.009	0.026	0.588
	LSTM	0.008	0.022	0.564
	Transformer	0.008	0.022	0.436
	X-N-L-T	0.009	0.023	0.437
	XG CGARCH	0.011	0.024	0.428
	NG CGARCH	0.011	0.023	0.779
	LSTM CGARCH	0.023	0.037	0.650
	TRANS CGARCH	0.010	0.022	0.775
	X-N-L-T CGARCH	0.011	0.023	0.610
h-20	**HAR**	0.009	0.022	0.716
	CGARCH	0.009	0.022	0.694
	XgBoost	0.007	0.021	0.664
	NgBoost	0.008	0.021	0.600
	LSTM	0.008	0.022	0.544
	Transformer	0.008	0.023	0.430
	X-N-L-T	0.009	0.023	0.577
	XG CGARCH	0.027	0.043	0.347
	NG CGARCH	0.031	0.049	0.362
	LSTM CGARCH	0.037	0.057	0.349
	TRANS CGARCH	0.025	0.039	0.406
	X-N-L-T CGARCH	0.026	0.042	0.354
h-60	**HAR**	0.009	0.024	0.628
	CGARCH	0.011	0.024	0.692
	XgBoost	0.008	0.022	0.645
	NgBoost	0.009	0.023	0.588
	LSTM	0.009	0.024	0.434
	Transformer	0.010	0.025	0.429
	X-N-L-T	0.009	0.024	0.476
	XG CGARCH	0.013	0.026	0.496
	NG CGARCH	0.010	0.024	0.688
	LSTM CGARCH	0.009	0.023	0.628
	TRANS CGARCH	0.011	0.025	0.470
	X-N-L-T CGARCH	0.016	0.028	0.434

This table shows us, for the RUSSEL 2000, the in sample performance metrics scored by our models for every forecasting horizon.

Table 19: **Models In sample Forecasting Results Across Different Horizons for GOLD**

Horizon	Model	MAE	RMSE	OF
h-5	**HAR**	0.004	0.012	0.742
	CGARCH	0.008	0.015	0.884
	XgBoost	0.004	0.012	0.712
	NgBoost	0.004	0.012	0.658
	LSTM	0.004	0.013	0.486
	Transformer	0.003	0.012	0.499
	X-N-L-T	0.005	0.014	0.185
	XG CGARCH	0.005	0.013	0.387
	NG CGARCH	0.004	0.013	0.475
	LSTM CGARCH	0.006	0.014	0.314
	TRANS CGARCH	0.005	0.013	0.714
	X-N-L-T CGARCH	0.006	0.014	0.135
h-10	**HAR**	0.004	0.013	0.743
	CGARCH	0.009	0.015	0.882
	XgBoost	0.004	0.012	0.705
	NgBoost	0.004	0.012	0.650
	LSTM	0.004	0.013	0.650
	Transformer	0.004	0.013	0.510
	X-N-L-T	0.005	0.014	0.259
	XG CGARCH	0.005	0.013	0.146
	NG CGARCH	0.009	0.017	0.076
	LSTM CGARCH	0.004	0.013	0.439
	TRANS CGARCH	0.004	0.013	0.137
	X-N-L-T CGARCH	0.007	0.015	0.160
h-15	**HAR**	0.004	0.013	0.742
	CGARCH	0.009	0.016	0.890
	XgBoost	0.004	0.012	0.700
	NgBoost	0.004	0.012	0.641
	LSTM	0.004	0.013	0.587
	Transformer	0.004	0.013	0.492
	X-N-L-T	0.005	0.014	0.186
	XG CGARCH	0.005	0.014	0.095
	NG CGARCH	0.005	0.014	0.012
	LSTM CGARCH	0.005	0.013	0.233
	TRANS CGARCH	0.004	0.013	0.105
	X-N-L-T CGARCH	0.005	0.013	0.589
h-20	**HAR**	0.004	0.013	0.714
	CGARCH	0.009	0.016	0.890
	XgBoost	0.003	0.012	0.694
	NgBoost	0.004	0.012	0.604
	LSTM	0.004	0.013	0.460
	Transformer	0.004	0.013	0.493
	X-N-L-T	0.004	0.013	0.440
	XG CGARCH	0.005	0.014	0.249
	NG CGARCH	0.004	0.013	0.448
	LSTM CGARCH	0.005	0.014	0.103
	TRANS CGARCH	0.007	0.014	0.059
	X-N-L-T CGARCH	0.005	0.014	0.590
h-60	**HAR**	0.004	0.007	0.670
	CGARCH	0.010	0.013	0.864
	XgBoost	0.003	0.007	0.711
	NgBoost	0.004	0.008	0.633
	LSTM	0.004	0.008	0.501
	Transformer	0.003	0.008	0.471
	X-N-L-T	0.004	0.008	0.483
	XG CGARCH	0.015	0.022	0.388
	NG CGARCH	0.028	0.042	0.446
	LSTM CGARCH	0.020	0.030	0.432
	TRANS CGARCH	0.006	0.010	0.148
	X-N-L-T CGARCH	0.039	0.054	0.487

This table shows us, for Gold, the in sample performance metrics scored by our models for every forecasting horizon.

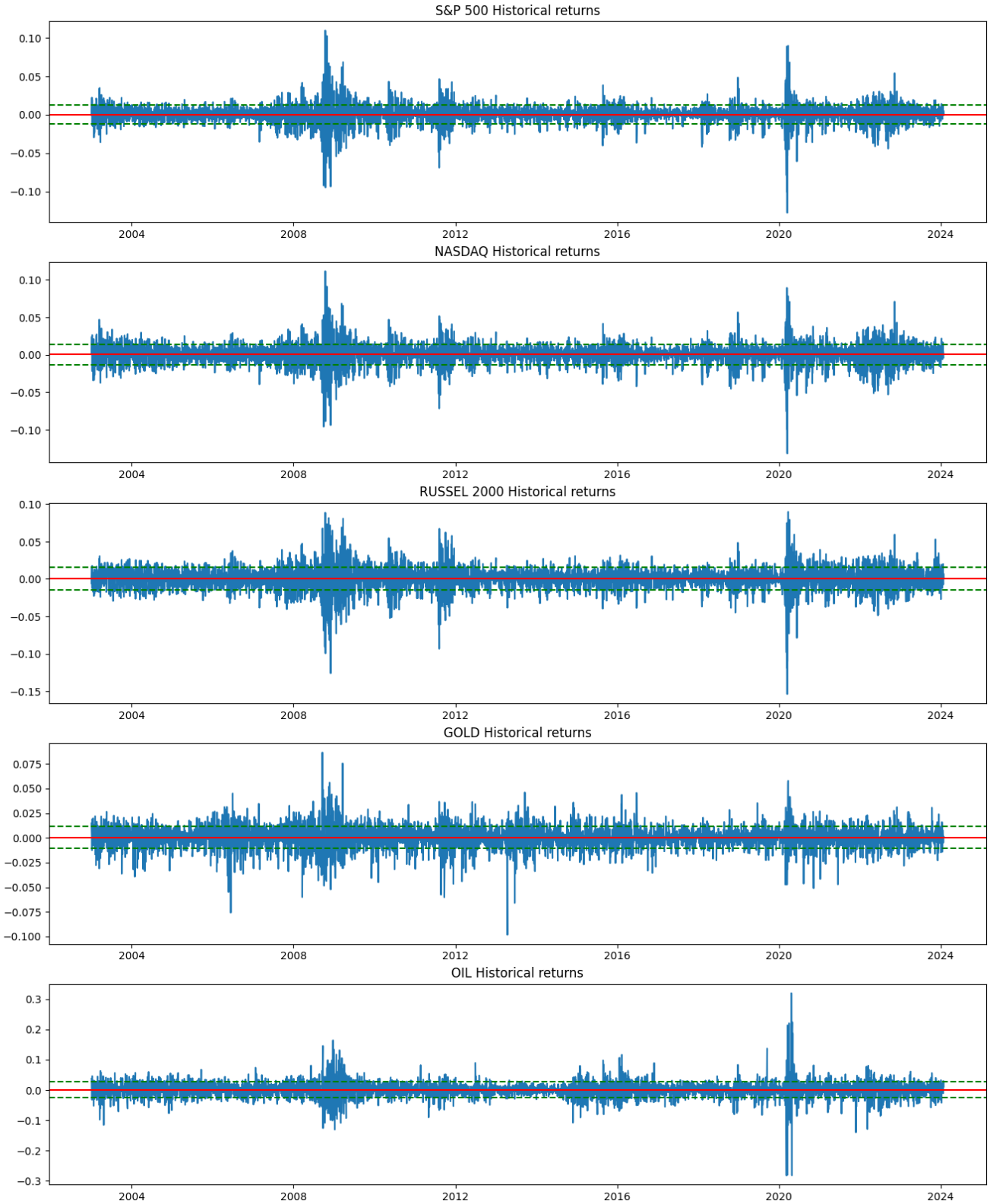
Table 20: **Models In sample Forecasting Results Across Different Horizons for OIL**

Horizon	Model	MAE	RMSE	OF
h-5	**HAR**	0.005	0.009	0.610
	CGARCH	0.006	0.009	0.346
	XgBoost	0.005	0.010	0.622
	NgBoost	0.005	0.013	0.581
	LSTM	0.005	0.009	0.487
	Transformer	0.005	0.009	0.509
	X-N-L-T	0.010	0.018	0.288
	XG CGARCH	0.006	0.010	0.648
	NG CGARCH	0.008	0.016	0.308
	LSTM CGARCH	0.006	0.010	0.728
	TRANS CGARCH	0.005	0.010	0.559
	X-N-L-T CGARCH	0.013	0.022	0.278
h-10	**HAR**	0.005	0.009	0.622
	CGARCH	0.006	0.010	0.345
	XgBoost	0.005	0.009	0.610
	NgBoost	0.005	0.009	0.589
	LSTM	0.005	0.010	0.522
	Transformer	0.005	0.011	0.470
	X-N-L-T	0.006	0.012	0.581
	XG CGARCH	0.005	0.009	0.507
	NG CGARCH	0.005	0.011	0.374
	LSTM CGARCH	0.006	0.011	0.333
	TRANS CGARCH	0.005	0.009	0.458
	X-N-L-T CGARCH	0.009	0.013	0.735
h-15	**HAR**	0.005	0.009	0.619
	CGARCH	0.006	0.010	0.379
	XgBoost	0.005	0.009	0.609
	NgBoost	0.006	0.017	0.592
	LSTM	0.005	0.010	0.519
	Transformer	0.006	0.010	0.548
	X-N-L-T	0.008	0.014	0.393
	XG CGARCH	0.008	0.014	0.525
	NG CGARCH	0.011	0.020	0.282
	LSTM CGARCH	0.009	0.016	0.199
	TRANS CGARCH	0.009	0.016	0.266
	X-N-L-T CGARCH	0.007	0.013	0.427
h-20	**HAR**	0.005	0.010	0.623
	CGARCH	0.006	0.010	0.390
	XgBoost	0.005	0.010	0.623
	NgBoost	0.006	0.013	0.580
	LSTM	0.005	0.010	0.486
	Transformer	0.005	0.011	0.467
	X-N-L-T	0.005	0.011	0.416
	XG CGARCH	0.013	0.022	0.220
	NG CGARCH	0.011	0.017	0.126
	LSTM CGARCH	0.014	0.024	0.247
	TRANS CGARCH	0.011	0.018	0.145
	X-N-L-T CGARCH	0.010	0.018	0.294
h-60	**HAR**	0.006	0.011	0.585
	CGARCH	0.008	0.014	0.552
	XgBoost	0.005	0.010	0.638
	NgBoost	0.006	0.011	0.628
	LSTM	0.007	0.012	0.473
	Transformer	0.007	0.013	0.519
	X-N-L-T	0.008	0.014	0.413
	XG CGARCH	0.011	0.018	0.549
	NG CGARCH	0.023	0.036	0.322
	LSTM CGARCH	0.016	0.025	0.248
	TRANS CGARCH	0.008	0.013	0.134
	X-N-L-T CGARCH	0.032	0.050	0.337

This table shows us, for Crude oil, the in sample performance metrics scored by our models for every forecasting horizon.

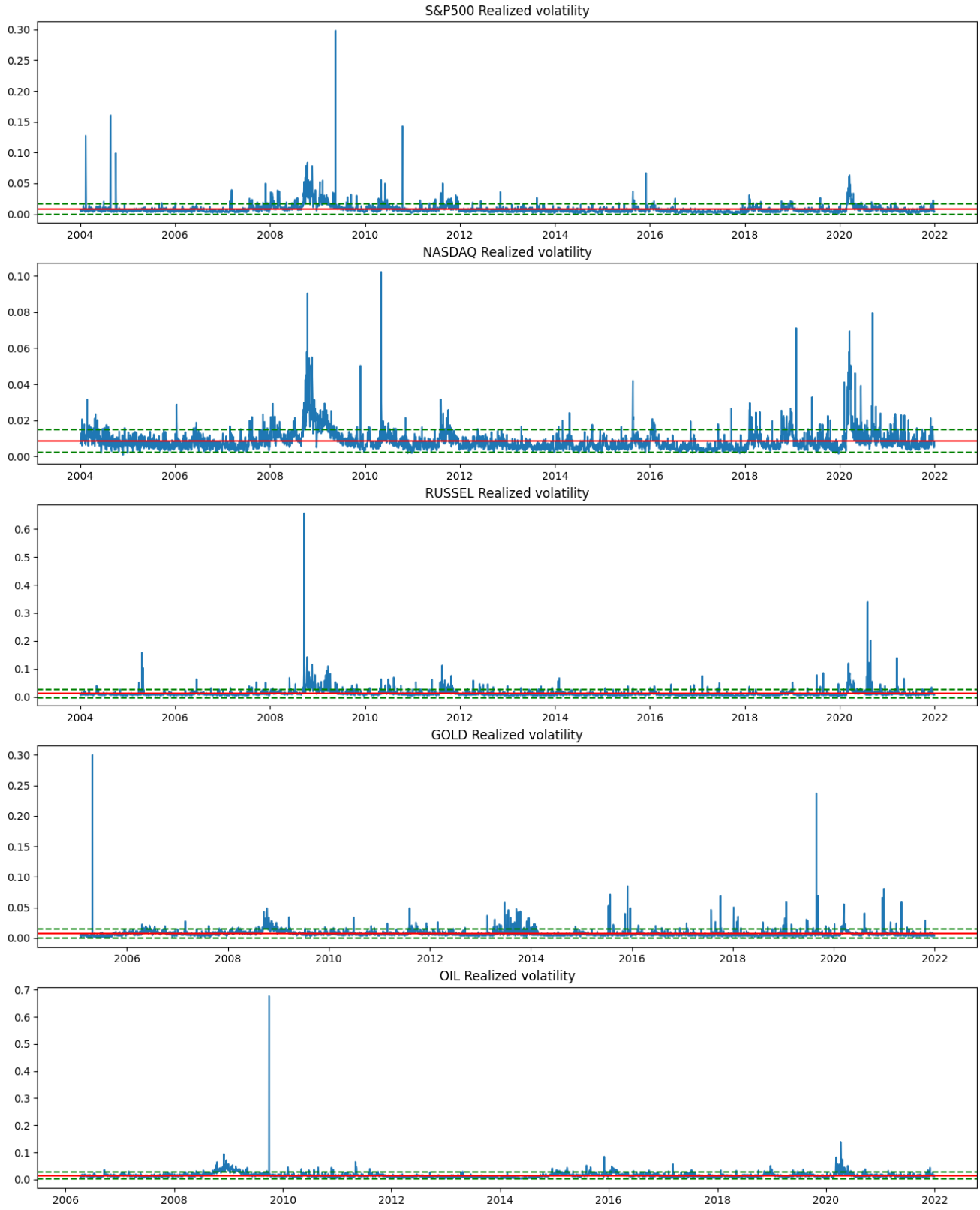
List of Figures

Figure 1: Historical daily returns



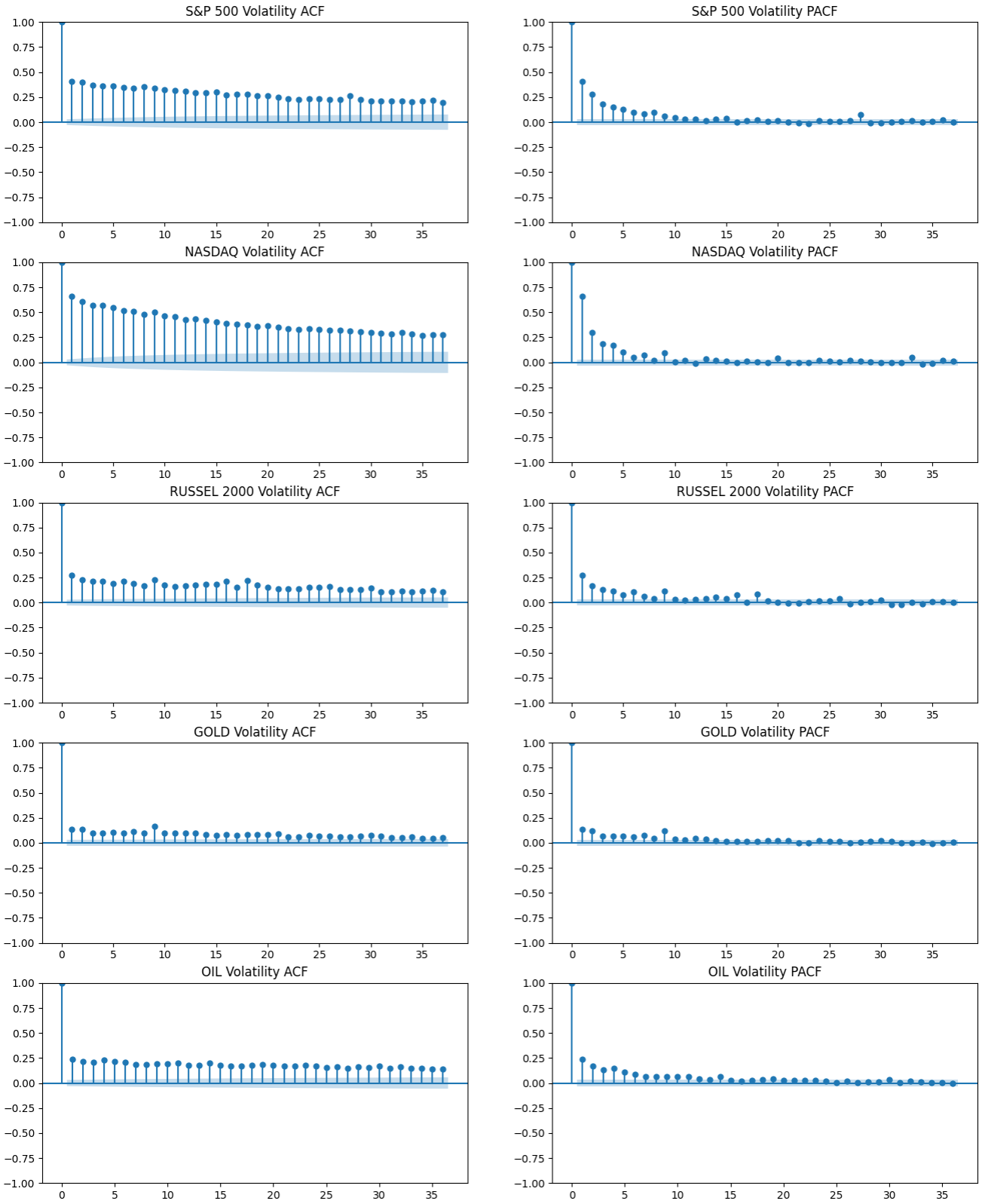
This figure shows us the historical daily returns for all our assets. As illustrated by the mean (red) and 1 standard deviation interval lines (dotted green), our returns series appear stationary and are on average slightly above 0 as per our sampling period.We can see that for every asset the largest spikes in returns happen approximately at the same periods (2008-2009 and 2020), which respectively coincide with the subprime and the COVID-19 crisis

Figure 2: Historical realized volatility of daily returns



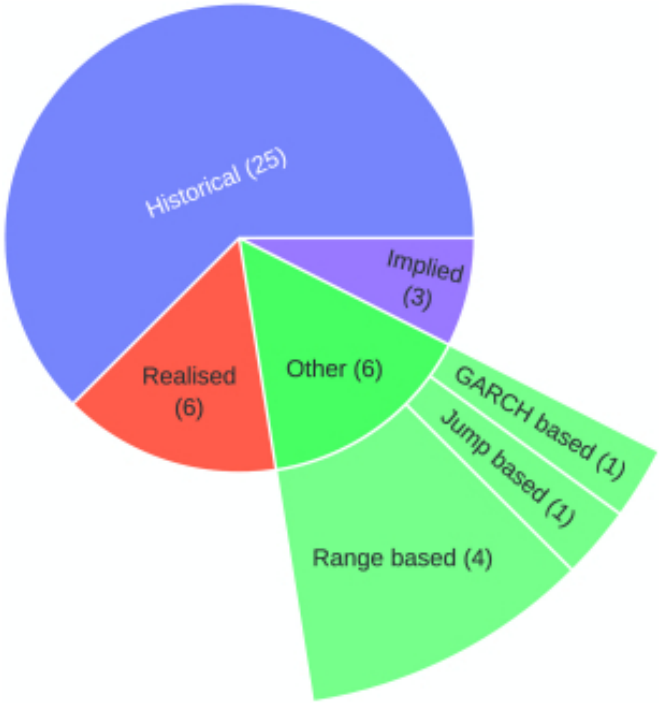
This figure plots the realized volatility time series for every asset.It is easy to see that on average, it seems visually that our oil proxy displays more ranging volatility than the other assets as well as the largest jump (around 2009), opposite to gold, which appears to be the less volatile asset as per our sampling period. The NASDAQ seems to display the most volatility jumps on our sampled window, which is in line with what we would expect from the technology sector. We can discern some coinciding volatility clusters across our assets, corresponding to systemic market-wise shocks (financial crisis) as well as idiosyncratic jumps or patterns associated with each assets' returns drivers.

Figure 3: Realized Volatility Auto-correlation and Partial Auto-correlation functions



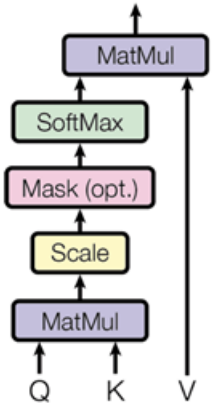
The panels respectively show us the Auto-correlation function and the Partial Auto-correlation function for each asset's volatility. For most assets, partial auto-correlation disappears after lag 10, which seems to suggest that our assets' long memory property is nonexistent after two calendar weeks.

Figure 4: Image from Ge et al. [1] showing the types of volatility definitions used by the papers in their literature review



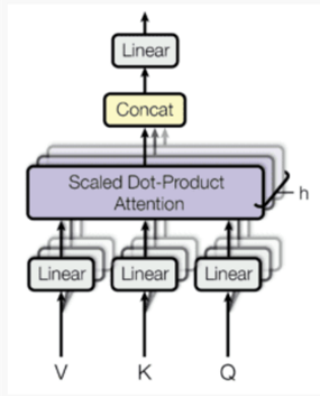
This figure from Ge et al. [1] informs us on the different volatility proxies used in the forecasting literature they reviewed. As illustrated, the ease of access to Historical volatility (simply computed from standard deviation of daily returns) makes it the most popular choice among volatility forecasting academics.

Figure 5: Self-attention mechanism as presented by Vaswani and al.



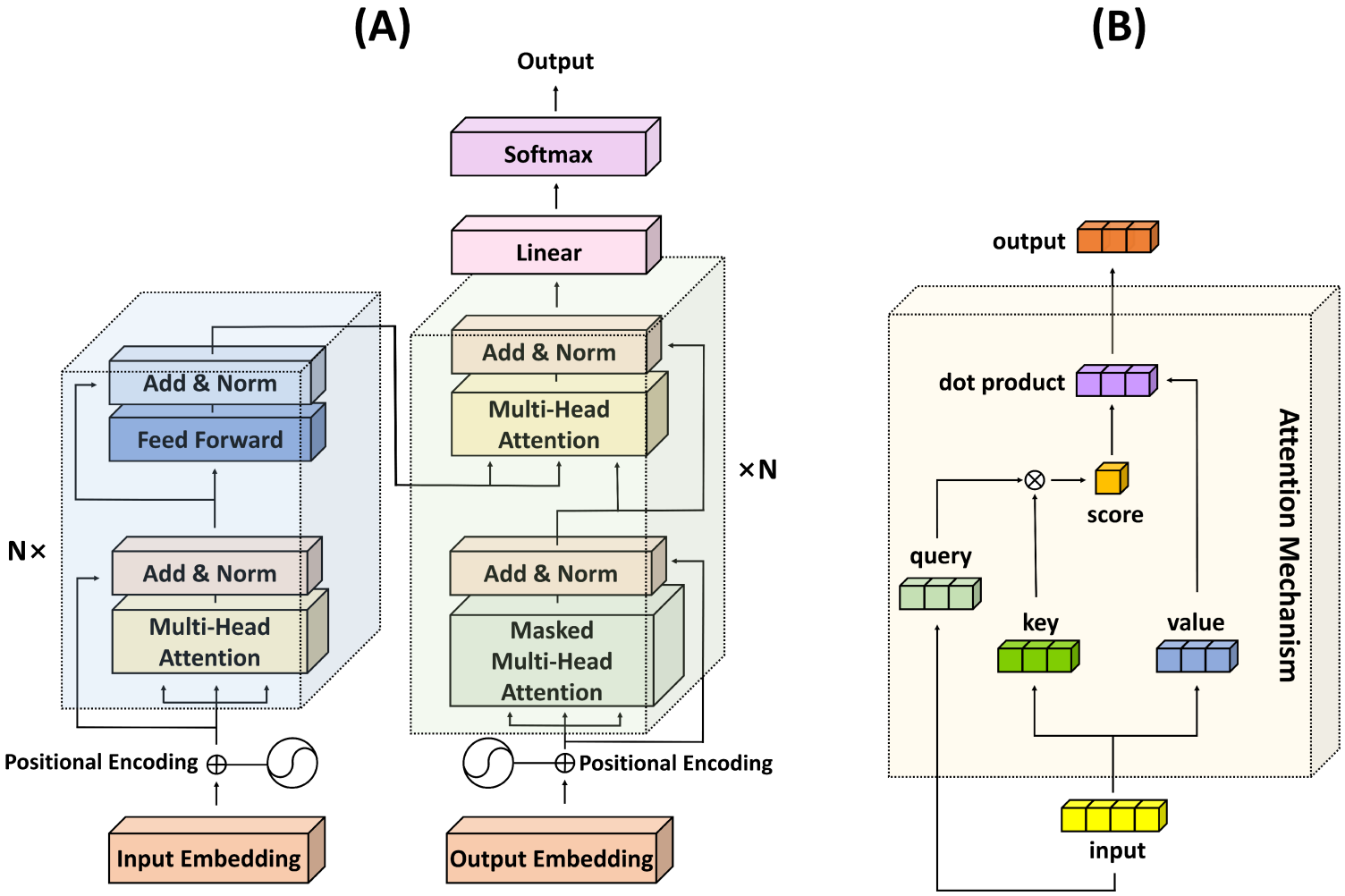
The mechanism takes as input three representations of the sequence, which are typically referred to as queries (Q), keys (K), and values (V). These representations are produced through matrix multiplication of the input embeddings with trainable weights. In practice, Q, K, and V might be identical in self-attention, as they come from the same previous layer output.The first step in the self-attention mechanism is to calculate the dot product of the queries with the keys. This operation measures the compatibility or similarity between different positions in the input sequence. The result is a weight matrix where each element represents the attention score from one element to another.The attention scores are then scaled down by dividing them by the square root of the dimension of the keys. This scaling helps in stabilizing the gradients during training, as it prevents the dot products from becoming too large and leading to very small gradients.An optional masking step can be applied, which is typically used in contexts where certain positions should not be attended to, such as padded elements or, in the case of the decoder, to prevent attending to future tokens (for causality) ; The scaled attention scores are passed through a softmax function, which converts them into probabilities. The softmax operation is performed on each row of the matrix, so each row sums up to 1. This step ensures that the attention weights across the sequence add up to 1, forming a probability distribution ; The output of the softmax function is then used to weight the values (V). The attention probabilities are multiplied with the values to produce a weighted sum, which represents the output of the attention mechanism for each position ; Finally, the results of the weighted sum are combined (typically through another matrix multiplication) to produce the final output of the self-attention layer, which then goes to subsequent parts of the Transformer model.

Figure 6: Multi-Head Self-Attention as presented by Vaswani and al.



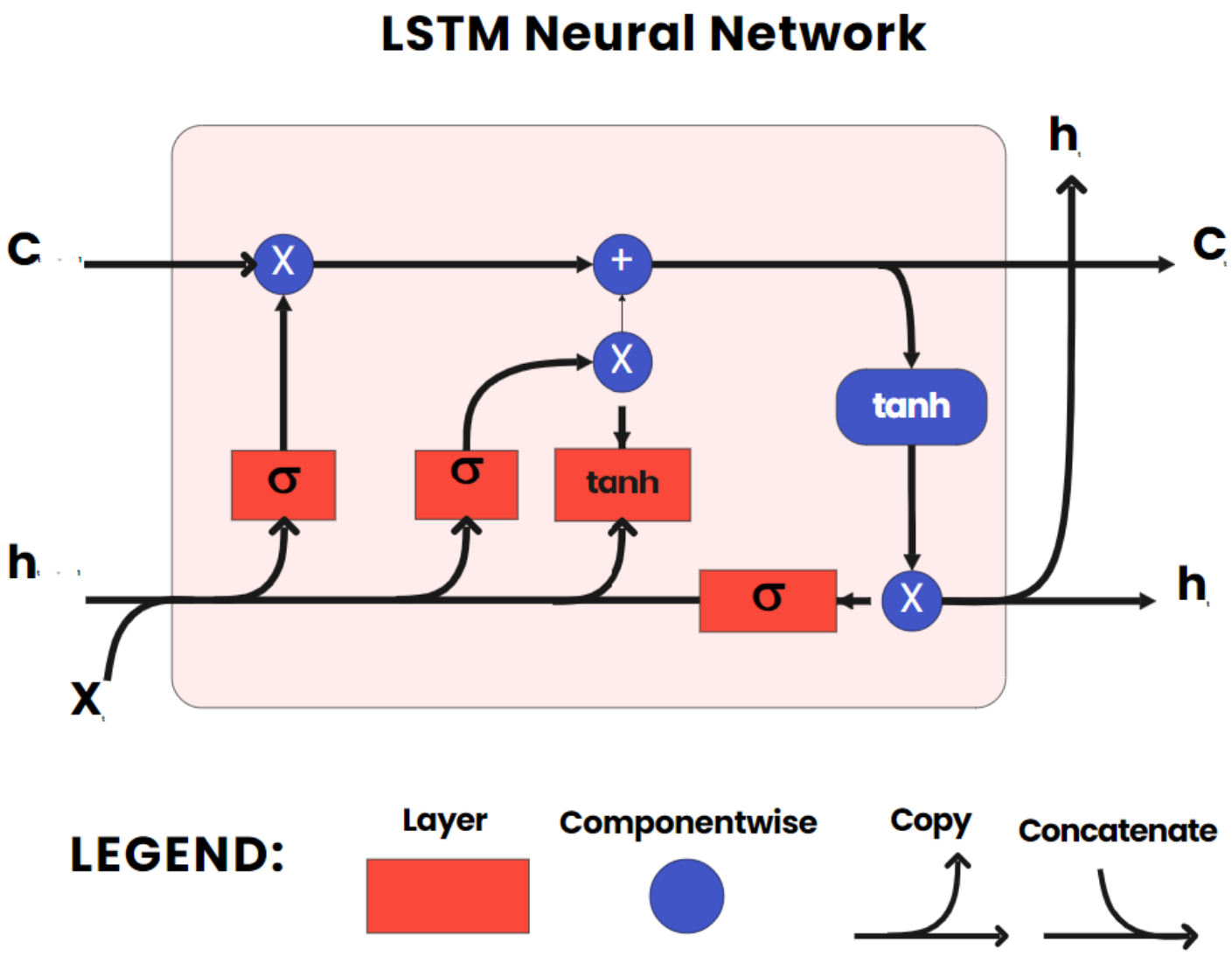
This figure depicts us the Multi-Head Self-Attention. Initially, the input vectors are linearly transformed into queries (Q), keys (K), and values (V) for each attention head ; For each head, scaled dot-product attention is computed just as in the single-head case (as illustrated in the previous image). The queries are dotted with the keys, scaled, and then a softmax function is applied to determine the attention weights. These weights are then used to compute a weighted sum of the values ; The output from each head is then concatenated, effectively combining the different learned representations ; Finally, this concatenated output is again linearly transformed. This step integrates information from all the heads to produce the final output for the multi-head attention layer. By comparing it to the previous single self-attention mechanism, multi-head attention can be thought of as running several instances of self-attention in parallel (each 'head' being an instance), with the concatenated result capturing a multi-faceted representation of the input sequence. This allows the model to pay attention to different parts of the sequence in varied ways simultaneously, offering a more complex and nuanced understanding than the single-head self-attention

Figure 7: Original transformer and self-attention architecture as proposed by Vaswani et al. [2]



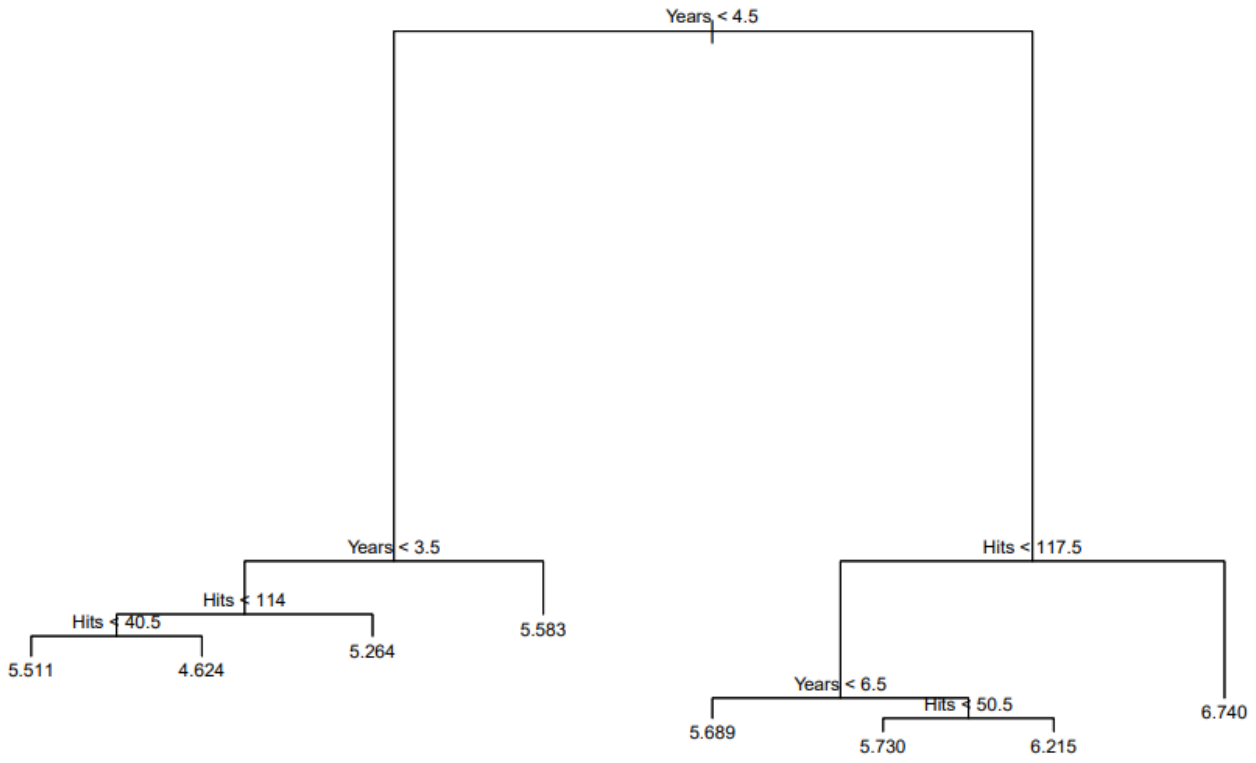
The diagram depicts the Transformer architecture, consisting of an encoder-decoder structure (A) and an attention mechanism (B). The encoder maps an input sequence to a sequence of continuous representations. For each input token, embeddings are summed with positional encodings to inject sequence order information. The encoder consists of N identical layers, each with two sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward network, both followed by an add & norm step which applies residual connections and normalization. The decoder also has N identical layers, with an additional masked multi-head attention layer to prevent positions from attending to subsequent positions. The final output is generated through a linear transformation followed by a softmax layer. Part (B) details the attention mechanism where queries, keys, and values are used in a scaled dot-product attention process to produce an output sequence.The attention scores are computed using dot products of queries with keys and are used to weight the values.

Figure 8: LSTM Cell showing computation flow



Schematic of an LSTM cell: The forget gate utilizes a sigmoid (σ) activation to regulate the retention of information from the previous cell state **c**. Simultaneously, the input gate determines potential new information to be added via a sigmoid activation and a tanh-generated vector of candidate values. The cell state is then updated by an element-wise multiplication of the forget gate's output with the old state and an addition of the input gate's scaled new candidate values **x**, allowing the cell to maintain and modify its memory. Lastly, the output gate, through another sigmoid function, filters the updated cell state via a tanh function to produce the next hidden state, which captures the relevant temporal information for the sequence being processed. Both the new hidden state **h** and the updated cell state **c** are forwarded to the next time step in the sequence.

Figure 9: Regression tree for predicting log salary from hits and years played.



At each internal node, we ask the associated question, and go to the left child if the answer is "yes", to the right child if the answer is "no".

Figure 10: NgBoost learning loop as illustrated by Duan and al.

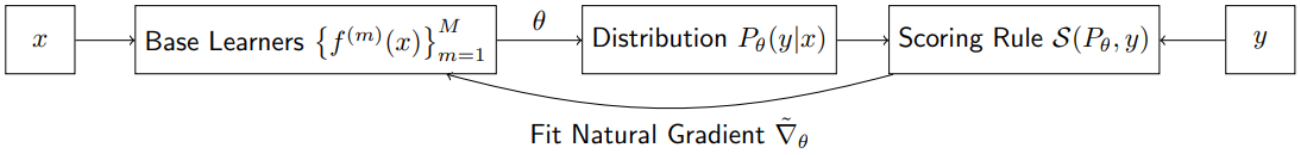


Figure 11: Ensemble model combining XgBoost, NgBoost,LSTM and Transformer(X-N-L-T)

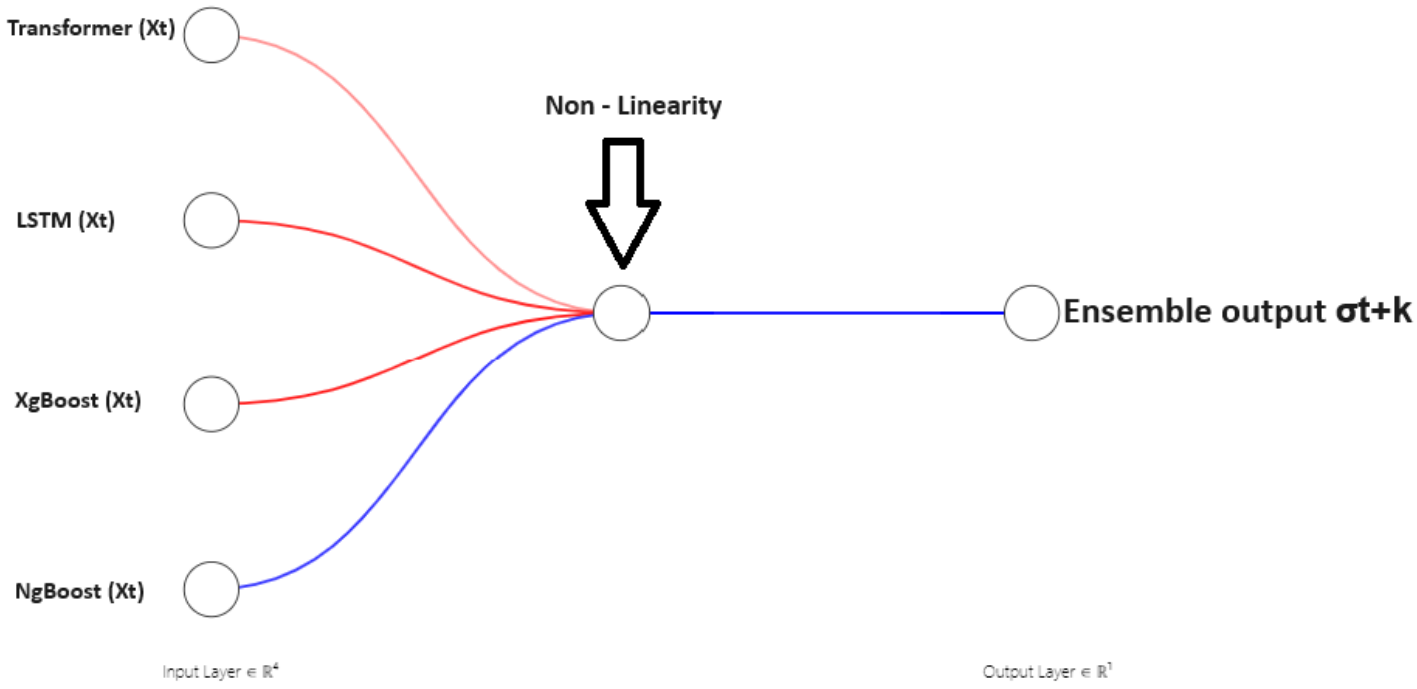


Illustration of the Single layer Feed Forward network used to create and ensemble X-N-L-T model from the outputs of the Transformer, LSTM, XgBoost and NgBoost models.

Figure 12: CGARCH enhanced Transformer

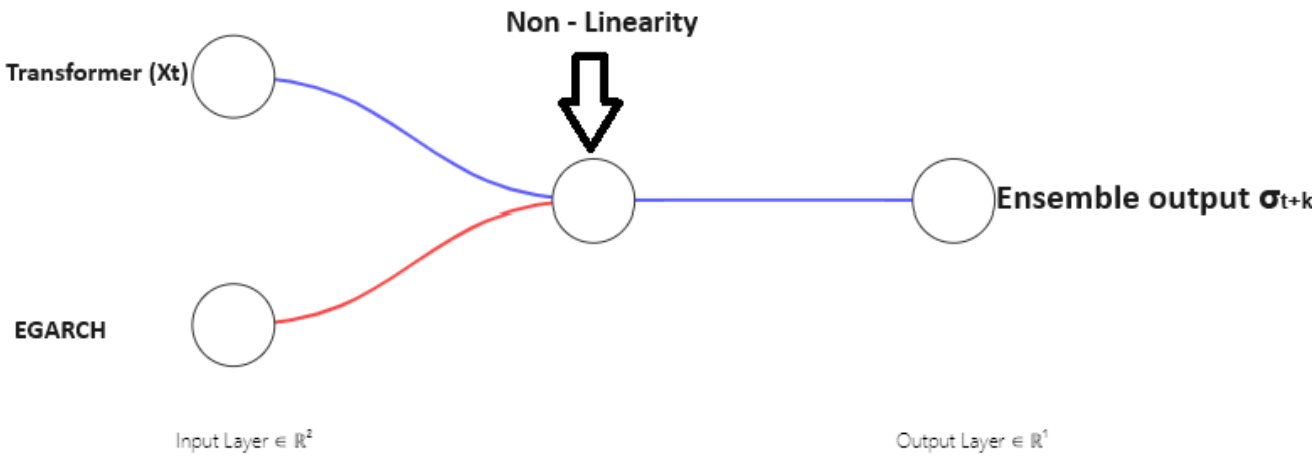


Illustration of the Single layer Feed Forward network used to create and ensemble CGARCH enhanced Transformer model from the outputs of the Transformer and CGARCH models.

Figure 13: CGARCH enhanced LSTM

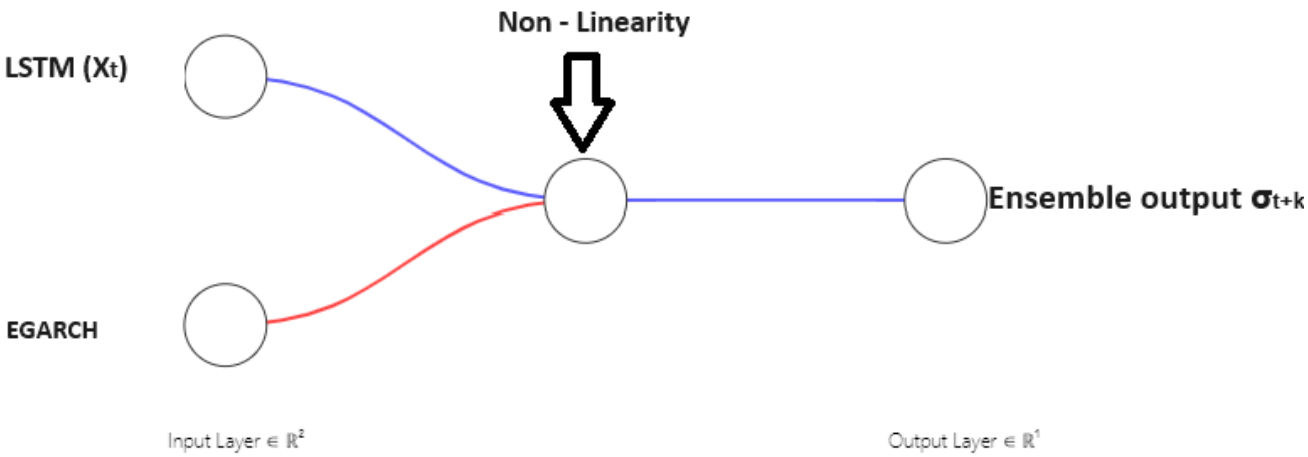


Illustration of the Single layer Feed Forward network used to create and ensemble CGARCH enhanced LSTM model from the outputs of the LSTM and CGARCH models.

Figure 14: CGARCH enhanced XgBoost

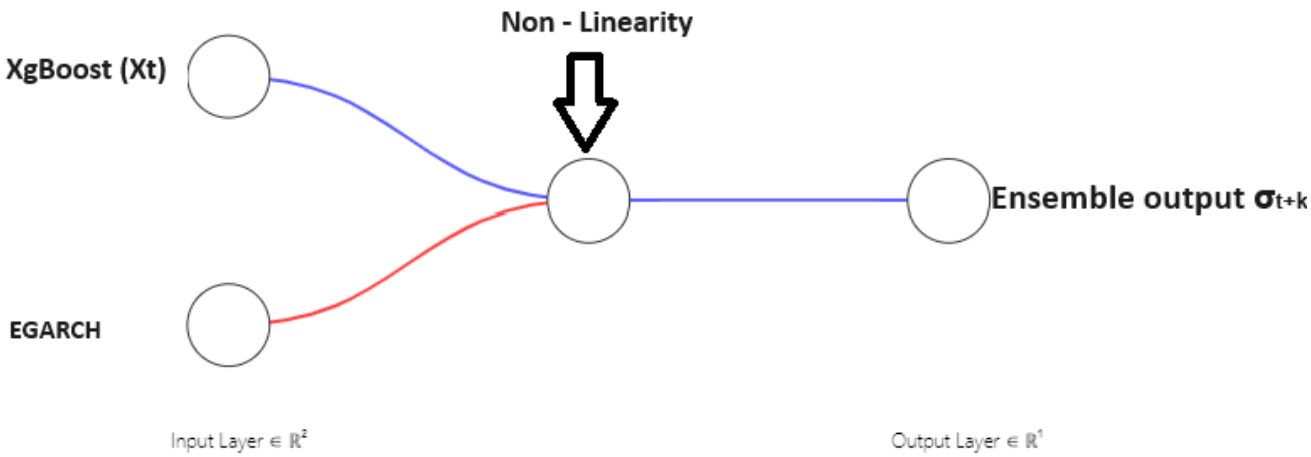


Illustration of the Single layer Feed Forward network used to create and ensemble CGARCH enhanced XgBoost model from the outputs of the XgBoost and CGARCH models.

Figure 15: CGARCH enhanced NgBoost

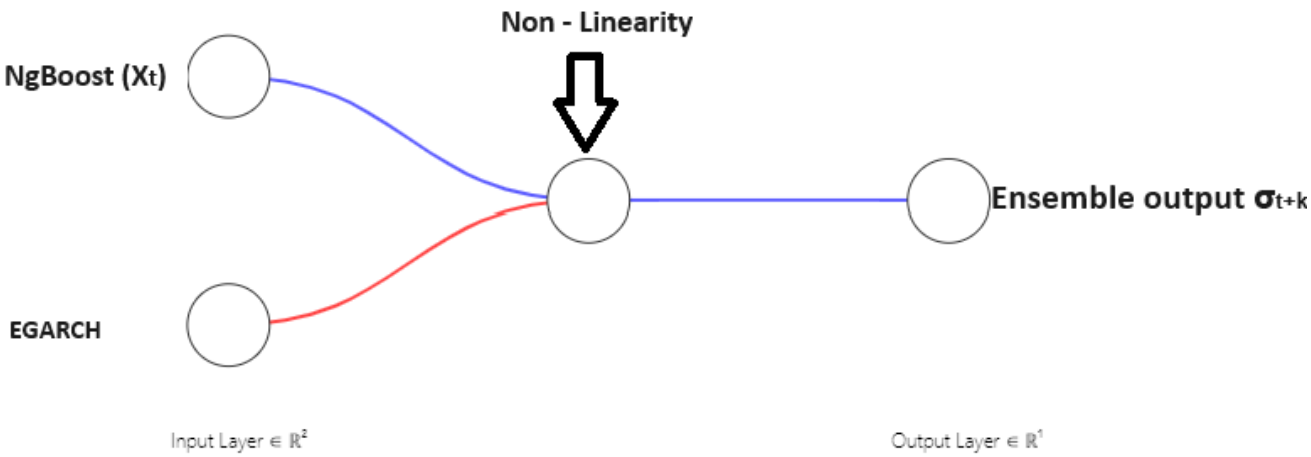


Illustration of the Single layer Feed Forward network used to create and ensemble CGARCH enhanced NgBoost model from the outputs of the NgBoost and CGARCH models.

Figure 16: CGARCH enhanced X-N-L-T

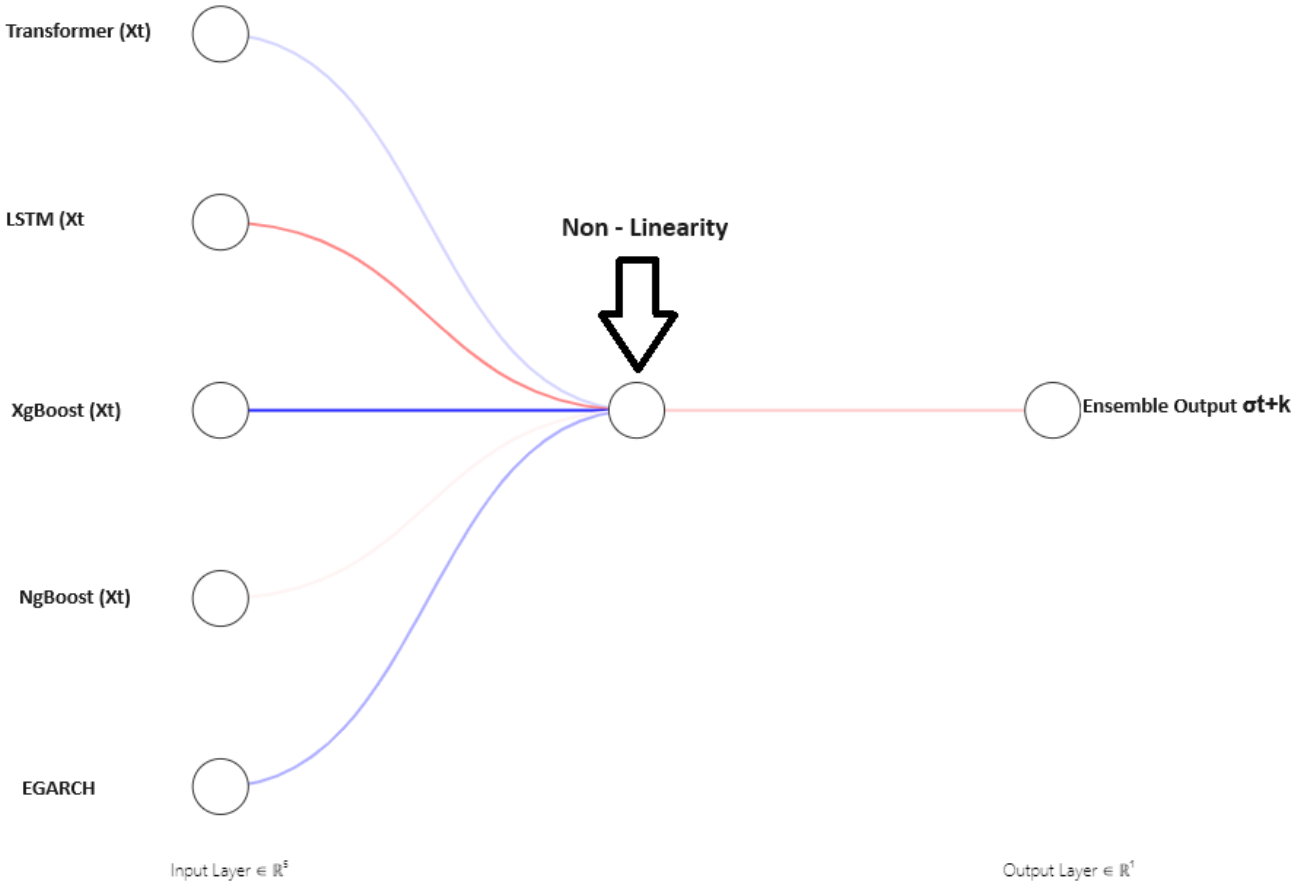


Illustration of the Single layer Feed Forward network used to create a CGARCH enhanced ensemble model from all the single models considered in this study, namely the Transformer, the LSTM, the XgBoost and the CGARCH. This is the largest model experimented with in this study in terms of size, parameters, and theoritical flexibility.

Figure 17: Rolling Window procedure