VU VRIJE UNIVERSITEIT AMSTERDAM

High-Frequency Volatility Forecasting: A Comparative Study of Traditional, Machine-Learning, and Deep-Learning Models

Master Thesis

Abstract

This thesis evaluates daily volatility forecasting for eleven leading technology stocks using high-frequency intraday trade data. Realized variance is measured via the realized kernel estimator. We compare traditional econometric models (HAR, GARCH variants, GAS, Realized GARCH, stochastic volatility), machine-learning methods (LASSO, ridge, elastic net, random forest, XGBoost, support-vector regression), and deep-learning architectures (LSTM, GRU, encoder–decoder, Transformer) using daily sentiment derived from news headlines as a predictor. Forecasts at horizons of 1, 5, 10, and 20 days are evaluated using rolling-window RMSE/MAE, Diebold–Mariano tests, Model Confidence Sets, and Value-at-Risk backtests. Results show that realized GARCH models deliver superior short-term forecasts and stochastic volatility models excel at longer horizons. Incorporating news sentiment yields modest improvements, while machine-learning and deep-learning approaches yield performance gains that do not justify their adoption over established methods..

Instructor:
dr. Yicong Lin

Rick Teeuwissen

Master Financial Econometrics

July 13, 2025

Contents

1 Introduction

Financial market volatility is a central concern in finance due to its fundamental role in asset pricing, portfolio management and risk control. Reliable forecasts of volatility are crucial for investors seeking to optimize risk-adjusted returns and for financial institutions calculating capital requirements. Despite its importance, modeling and forecasting volatility remains one of the most challenging tasks in financial econometrics.

This paper narrows the focus to forecasting daily volatility for a selection of major technology companies. We construct a high-frequency dataset of intraday trades for eleven market leaders– Apple, Microsoft, Alphabet, Meta, Tesla, Nvidia, Tsmc, Asml, Amd, Ibm and Intel – spanning recent years. Using this rich intraday data, we compute the daily realized variance for each return series via the robust realized kernel estimator of Barndorff-Nielsen et al. [2009], which serves as the dependent variable in our forecasting exercise. We employ a wide range of forecasting models, starting with standard univariate time-series models. These include variants of HAR and GARCH models that use only each stock's own past volatility and returns to predict future volatility. Such models represent the traditional approach and set a baseline level of forecast accuracy.

The key innovation of our approach lies in expanding traditional volatility predictors with a sentiment factor extracted from daily news headlines related to each company. To extract these signals, we apply a state-of-the-art natural language processing (NLP) model that classifies headlines into sentiment categories such as positive, negative, or neutral. From these classified headlines we derive a daily sentiment score, relieving the overall tone of news coverage. By including such sentiment indicators as inputs, our models can utilize information beyond past returns and volatility, leveraging forward-looking information in news articles. The study employs a range of linear and non linear machine learning models to evaluate the usefulness of sentiment data.

In addition to classical and machine learning approaches, this study places particular emphasis on deep learning architectures due to their ability to capture nonlinear dependencies. We implement several recurrent neural networks (RNNs), including the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). Moreover, we deploy an encoder–attention-decoder framework and an auto-regressive Transformer model. Originally introduced by Vaswani et al. [2017], the Transformer architecture has become the dominant approach in NLP due to its self-attention mechanism and superior performance on long-range sequences. By applying this architecture to volatility forecasting, we aim to assess whether the strengths of Transformers translates into the realm of finance.

The results of this study offer valuable insights into the extent to which news sentiment carries predictive power relative to solely historical volatility-based forecasts. Since better volatility forecasts translate into better risk management and pricing, financial professionals could incorporate news based signals into volatility trading strategies, option valuation, or portfolio risk controls. Furthermore, focusing on the technology sector which is known for rapid innovation and significant media exposure highlights the practical relevance of our approach, as tech stocks often exhibit volatility that is news-driven (e.g., product launches). By providing a framework to anticipate such volatility using news data, this work has direct implications for traders and investors actively trading these stocks.

This thesis makes three key contributions. First, we empirically quantify the added predictive value of incorporating news-based sentiment scores into volatility models, offering new insights into the role of qualitative information in financial forecasting. Second, it explores the potential of combining sentiment scores with an auto-regressive Transformer model in financial forecasting and rigorously compares its performance to a robust set of over 50 traditional- and modern forecasting techniques. Lastly, the realized kernel estimator of Barndorff-Nielsen et al. [2009] is used to construct a robust daily volatility measure from high-frequency trade data, which is a more precise measure of the unobserved volatility.

The remainder of this thesis is organized as follows. Chapter 2 reviews the already existing research upon which this paper builds. In Chapter 3 we outline the data processing process of the high-frequency dataset. Chapter 4 reviews the theoretical background on realized volatility and kernel-based estimators. In Chapter 5 and 6 we discuss classical observation driven volatility models and parameter-driven models such as stochastic volatility, which will serve as benchmark approaches. Chapter 7 outlines the integration of market sentiment into our analysis and introduces the machine learning models and feature engineering steps. Chapter 8 focuses on deep learning models, detailing the architectures of the LSTM and Transformer networks used for forecasting. Chapter 9 discusses the out-of-sample evaluation procedure. Finally, in Chapter 10 we outline general results and Chapter 11/12 concludes with a summary of key findings and suggestions for future research in this area.

2 Literature Review

Volatility forecasting is one of the most important applications of financial econometric theory. The ability to produce accurate volatility forecasts is an essential component in the toolbox of financial institutions to asses their risk and decide upon their capital reserves in times of economic stress. Early surveys such as those by Poon and Granger [2003] show this. Better volatility forecasts can improve portfolio allocation and therefore enable institutions to manage their risk.

However, forecasting volatility is challenging. The true instantaneous volatility is even unobservable, forcing researches to use proxies which are computed using high-frequency intraday returns. One of the most popular proxies utilized in the literature is the realized variance (RV). It denotes the sum of squared intraday returns over some interval. Barndorff-Nielsen and Shephard [2002] showed this estimator to be an unbiased in most cases. However, at ultra-high sampling rates due to market-microstructure noise this is no longer the case. Barndorff-Nielsen et al. [2009] solved this problem by developing a realized kernel that fully makes use of the richness of high frequency data, yielding a better estimator of the underlying true volatility.

Over the past decades, a wide array of models have been developed to make volatility forecasting less challenging. Efforts started with classic parametric econometric approaches such as the ARCH model, introduced by Engle [1982] and later generalized by Bollerslev [1986a] to the GARCH. Since the birth of these models numerous variants such as the EGARCH and GJR-GARCH erose and became the industry standard in volatility modeling.

Building on these developments, Corsi [2009a] introduced the Heterogeneous Autoregressive (HAR) model, which treats realized volatility over daily, weekly, and monthly horizons as predictors. The HAR model is appealing because of its simplicity. Its ability to capture the long-term behavior of volatility with a few aggregated lags makes it difficult to beat, often even outperforming GARCH models. Furthermore, Hautsch and Hsieh [2012] extended the HAR model to depend on a realized kernel estimator of the volatility, rather than the RV and documented substantial out-of-sample gains. Extensions of the HAR such as the HAR-J for jumps or the HAR-Q which models the volatility of volatility also showed improvements in forecast accuracy [Bollerslev et al., 2016].

An increasingly popular class of volatility models is the Generalized Autoregressive Score (GAS) introduced by Creal et al. [2013]. In GAS models, volatility estimates are updated using the score of the conditional likelihood. Creal et al. [2013] demonstrated that GAS specifications can outperform conventional GARCH and HAR models on financial returns. Furthermore, Degiannakis et al. [2015] shows that GAS-based volatility forecasts compete strongly with HAR models.

Furthermore, the class of linear Realized GARCH models proposed by Hansen and Lunde [2011] improves the standard GARCH and HAR models by incorporating a daily realized measure to model the conditional variance with an out-of-sample RMSE reduction of approximately 5–10% on individual stocks. Tian and Hamori [2015] applies the Realized GARCH framework to short-term interest rates, showing that it beats conventional GARCH-type models. More recently, Xiao and Zhang [2023] shows the relevancy of this model by finding that Realized GARCH yields accuracy improvements on high volatility online assets such as Bitcoin.

Overall, traditional econometric models set a strong benchmark. This provides the motivation to further explore richer data sources and non-linear models. In recent years, machine learning (ML) techniques have gained popularity in financial volatility forecasting, thanks to their ability to capture complex non-linear patterns and integrate high-dimensional inputs. Christensen et al. [2023] provide a comprehensive comparison of ML algorithms for one-day-ahead prediction of stock volatility. They show that tree-based ensemble methods and other ML models can significantly improve predictive accuracy relative to HAR or GARCH benchmarks.

Similarly, Bucci [2020] demonstrated that feed-forward neural networks offer better out-of-sample forecasts of realized variance than traditional models in many cases. An extensive literature review by Gunnarsson et al. [2024] concludes that ML methods often yield forecast performance which is comparable or better than classic econometric models. Zhang et al. [2023] and Díaz et al. [2024] both find that ML approaches seem to be particularly competitive if combined with diverse features such as technical indicators, macro factors, or lagged variables. However, Gunnarsson et al. [2024] finds that no single approach dominates universally. While ML models can harness larger information sets and complex relations, simple econometric models remain competitive in certain settings. In general, the evidence suggests that machine learning expands the toolkit for volatility forecasting and can outperform traditional models, especially in dealing with nonlinear dynamics.

Within the ML domain, deep learning (DL) models have attracted significant attention for volatility forecasting. Especially Recurrent neural networks (RNNs), which are well-suited for sequential data, have been widely applied. In particular, the unique architecture

of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks can capture long temporal dependencies and complex nonlinearities. Empirical studies often find these architectures yield top-tier predictive performance. For instance, the survey by Gunnarsson et al. [2024] and Lei et al. [2021] report that LSTM- and GRU-based models consistently rank among the best-performing models.

These findings imply that these advanced architectures hold great potential. However, issues like data requirements and interpretability remain unsolved. DL/ML methods are "black box" algorithms, which means the specifics on how the model makes decisions between training and validation rounds are hidden, making it difficult to understand how the model formed its prediction. This lack of model transparency has been highlighted in the literature, and Hassani et al. [2024] mentions the growing interest in hybrid models which combine econometric specifications with the pattern recognizing capabilities of ML. However, very few studies have managed to develop DL models who have the same interpretability that comes natural to econometrical models [Gunnarsson et al., 2024].

Besides developing hybrid models, a rapid increasing area of research integrates textual data into volatility forecasting, based on the intuition that news drives the behavior of investors because it influences their volatility expectations [Engle and Ng, 1993]. Earlier work used news counts or simple sentiment indices as exogenous regressors in volatility models. For example, Rahimikia and Poon [2020],Audrino and Offner [2020] and Barbaglia et al. [2023] extended a HAR model with a sentiment index derived from news headlines and finds that during times of economic uncertainty news sentiment provided a statistically significant accuracy boost. Similarly, Lei et al. [2021] construct a sentiment index from millions of stockholders' forum comments. They report that an LSTM fitted with the sentiment factor achieved lower forecast errors than both an LSTM without sentiment. Bodilsen and Lunde [2025] focuses on news analytics for individual stocks, showing that macroeconomic news sentiment significantly improves volatility forecasts, particularly at longer horizons, while firm-specific news sentiment yields modest benefit.

Furthermore, due to their massive success in NLP, driving the current AI revolution, researchers have started experimenting with Transformer-based architectures, which are based on self-attention mechanisms instead of the recurrence in RNNs. Transformers have achieved state-of-the-art results in some time-series applications and promise to handle long sequences efficiently. Frank [2023] examined a Transformer model for S&P,500 stock returns and found it outperformed LSTM and random forest when trained on pooled cross-sectional data. However, evidence that this result translates to volatility forecasting is very limited. Ramos-Pérez et al. [2021] first applied a multi-transformer to one-day-ahead S&P,500 realized volatility and reported gains over traditional RNNs. Goel et al. [2024] extended this result by demonstrating that the Transformer can capture very long-horizon variance patterns. This evidence is however not universal. Qiu et al. [2025] for example found that LSTMs still outperform Transformers on a 30-year volatility dataset.

Despite these advances, the potential of Transformer-based volatility forecasting, particularly in combination with news sentiment data remains largely undiscovered. This thesis addresses this by systematically evaluating the performance of such a model, bench-marked against a robust ensemble of traditional and modern models.

3 Data

The empirical work utilizes the NYSE Trade and Quote (TAQ) database, which is accessible through Wharton Research Data Services (WRDS). We assemble a high-frequency dataset of trade records only. This is done in line with the recommendation in Barndorff-Nielsen et al. [2009] on the basis that trade-based information captures fundamental price shifts. Our sample spans from 2018-01-03 to 2025-02-27, offering more than seven years of high frequency observations.

3.1 Data Preparation

With the use of high frequency data, an aggressive data cleaning strategy is essential. High frequency prices are often contaminated by a variety of issues, such as recording errors, timestamp mismatches, and data feed delays. Since realized kernel estimators treat all observations equally, only a few of such faults are sufficient for the volatility estimates to become distorted. To prevent this, we apply the cleaning procedure proposed by Barndorff-Nielsen et al. [2009]. The first step is to apply some standard filters across all observations. We begin with the removal of obvious recording mistakes. This is done by locating any observations with a bid, ask or price equal to zero. Furthermore, any observation outside normal trading hours (09:30 - 16:00) is omitted from the sample. This step focuses the analysis on the continuous flow during the trading session and avoids overnight jumps. Next, we limit ourselves to a single exchange, the NASDAQ since it reduces trade/quotes mismatches between exchanges.

We continue with trade specific filters which are mainly concerned with keeping the sample consistent. The first step is primarily concerned with eliminating bookkeeping artifacts. The NYSE allows brokers to revise a previously reported trade. If this occurs the new recorded trade is marked with a non-zero "correction" tag. Leaving both the original and corrected trade in the sample would create an artificial price bump in opposite directions not based on any real new information [NYSE Market Data, 2011].

The second step excludes trades with so-called "special sale conditions". An example of this would be trades negotiated off exchange and often occur at prearranged prices or outside regular trading hours. Such trades are not part of the continuous intraday market dynamics and could exaggerate volatility if they occur far from the exchange price. Because high frequency trade information can yield dozens of observations on the same time stamp Barndorff-Nielsen et al. [2009] proposes that these should be collapsed in into a single observation through aggregation. When dealing with many observations on an infinitesimally small time scale, many bumps due to micro-bursts of algorithmic activity inflates the sample size and volatility. Therefore, we transform the data to a second by second price path by taking the median of intrasecond observations. In this case, the median is preferred over the mean or last price in that second, because of its robustness to outliers.

As a last step, we adjust for stock splits. This is essential because a split lowers the share price while increasing the share count, such that the firm's value is unchanged. If not accounted for, a stock split will result in a sudden price drop, inflating volatility and triggering false signals picked up by any realized kernel estimator.

4 Realized Measures of Volatility

In continuous time, the Quadratic Variation QV_t is the total population variance of a log-return process over a given horizon. Therefore, to make volatility forecasts, QV_t is the measure we aim to estimate. In the continuous time setting, we assume that the efficient log prices follow the following Itô jump-diffusion

$$\mathrm{d}\log P(t) \;=\; \mu(t)\,\mathrm{d}t \;+\; \sigma(t)\,\mathrm{d}W(t) \;+\; \mathrm{d}J(t), \qquad J(t) = \sum_{k=1}^{N_t} \kappa_k, \tag{1}$$

where $\mu(t)$ is the drift component, $\sigma(t)$ is the instantaneous volatility, and $W(t)$ is a standard Brownian motion. Abrupt shocks can cause price movements which are too violent to be captured by the continuous Brownian path. Such events get incorporated in the model through $J(t)$, which is a a compound Poisson process.

$$J(t) = \sum_{k=1}^{N_t} \kappa_k, \qquad N_t \sim \text{Poisson}(\lambda t),$$

where N_t counts the number of jumps that have arrived up to time t and the κ_k are i.i.d. random jump sizes. Furthermore, the parameter λ is the intensity. That means, on average λ jumps occur per unit of time. If $\lambda = 0$ (or all $\kappa_k = 0$), which would imply $J(t) = 0$ the model collapses back to pure diffusion. Under this assumption, let P_t be a realization from the price process $P(t)$. Then we can define $r_t = \Delta \log P_t$ as the continuous returns. Therefore, it is possible to write r_t as

$$r_t \;=\; \mu_t \;+\; \int_{t-1}^{t} \sigma(s)\,\mathrm{d}W(s), \tag{2}$$

with

$$\mu_t \;=\; \int_{t-1}^{t} \mu(s)\,\mathrm{d}s, \quad IV_t \;\equiv\; \int_{t-1}^{t} \sigma^2(s)\,\mathrm{d}s. \tag{3}$$

Conditional on mean μ_t and variance IV_t, we have

$$r_t \;\big|\; \mu_t,\, IV_t \;\sim\; \mathcal{N}\big(\mu_t,\, IV_t\big), \tag{4}$$

where it is implied that IV_t represents the Integrated Variation, capturing the population variance of r_t over $[t-1, t]$. In other words, IV_t is the population volatility part that models "normal" day-to-day noise without shocks. The Quadratic Variation QV_t, however, is the limit of the sum of squared increments as the sampling frequency within $[t-1, t]$ tends to infinity

$$QV_t \;=\; \text{plim}_{N\to\infty} \sum_{i=1}^{N} \big[r(t_i) \;-\; r(t_{i-1})\big]^2, \tag{5}$$

where $t_0 = t - 1 < t_1 < \cdots < t_N = t$ denotes a partition interval which approaches zero as $N \to \infty$. This means that under the assumption of no microstructure noise or price jumps ($J(t) = 0$, QV_t coincides with IV_t. In reality, however, price discontinuities and other market frictions typically make $J(t)$ non-zero such that QV_t becomes

$$QV_t \;=\; IV_t \;+\; \sum_{k=1}^{N_t} \kappa_k^2, \tag{6}$$

such that

$$IV_t \leq QV_t, \tag{7}$$

always holds. This makes QV_t the full realized variance of the price path, combining ordinary noise and unforeseen jumps. Equation (7) implies estimating QV_t is crucial for risk management and back-testing to prevent underestimating volatility.

4.1 Realized Volatility

The discrete time equivalent of equation 2 writes

$$r_t = 100 \times (\log P_t - \log P_{t-1}), \tag{8}$$

where the rescaling yields r_t to be in percentage terms. The realized volatility RV_t can be written as

$$RV_t = \sum_{i=1}^{m} r_{i,t}^2, \tag{9}$$

where $r_{i,t}$ is the i-th intradaily return and m is the number of intradaily returns used to compute RV_t. Andersen et al. [2003] show that

$$\sqrt{m}\left(RV_t - IV_t\right) \xrightarrow{d} \mathcal{N}\left(0,\; 2\int_{t-1}^{t} \sigma^4(s)\, ds\right). \tag{10}$$

This implies that $RV_t \to QV_t$ as $m \to \infty$ if and only if when $N_t = 0$ such that $J(t) = 0$. Koopman et al. [2005] identified 5-minute intervals as a suitable frequency where accuracy and minimization of microstructure effects are balanced. Since we restricted ourselves to a typical trading session from 9:30 AM to 4:00 PM this results in $m = 78$.

4.2 Realized Kernel Estimation of Quadratic Variation

To account for market microstructure noise and other frictions in high-frequency financial data, Barndorff-Nielsen et al. [2009] developed the Realized Kernel (RK) estimator. This statistic refines volatility measurement by mitigating the distortion introduced by problems such as asynchronous trading and recording errors. A key advantage of such an approach is its robustness to such noise, while preserving the efficiency of volatility estimation through dense sampling.

Let $\{X_{\tau_j}\}_{j=0}^{n}$ denote a sequence of log-prices observed at irregularly spaced time stamps $0 = \tau_0 < \tau_1 < \cdots < \tau_n = T$, over a single trading day. Then we can define high-frequency returns by $x_j = X_j - X_{j-1}$, for $j = 1, \ldots, n$, where the price series $\{X_j\}$ is constructed by local averaging at the boundaries to mitigate end-point effects. Specifically, Barndorff-Nielsen et al. [2009] advocate using

$$X_0 = \frac{1}{2}(X_{\tau_0} + X_{\tau_1}), \quad X_n = \frac{1}{2}(X_{\tau_{n-1}} + X_{\tau_n}), \tag{11}$$

because this approach reduces the bias introduced by extreme values in the first and last observations, ensuring that the noise component of the realized kernel estimator converges to zero as the sample size increases. Barndorff-Nielsen et al. [2009] also mention that this is often not necessary for actively traded assets. However, since the trading frequency differs across stocks and failure to correct for end points effects can result in significant bias, this step is taken as a precaution.

The Realized Kernel estimator for a log-price series X is defined as

$$K(X) = \sum_{h=-H}^{H} k\left(\frac{h}{H+1}\right)\gamma_h, \quad \gamma_h = \sum_{j=|h|+1}^{n} x_j x_{j-|h|}, \tag{12}$$

where $H \in \mathbb{N}$ is a bandwidth parameter, γ_h denotes the empirical autocovariances of returns, and $k(\cdot)$ is a symmetric kernel function. There exist several options for $k(\cdot)$, but the the Parzen kernel is often favored for its smoothness and positivity-preserving properties, which is useful in volatility applications

$$k(x) = \begin{cases} 1 - 6x^2 + 6x^3 & \text{if } 0 \le |x| \le \frac{1}{2}, \\ 2(1-|x|)^3 & \text{if } \frac{1}{2} < |x| \le 1, \\ 0 & \text{if } |x| > 1. \end{cases} \tag{13}$$

The bandwidth H determines the degree of smoothing. Selecting a larger bandwidth allows for greater autocorrelation in microstructure noise. However, it also yields greater variance of the kernel estimator. Barndorff-Nielsen et al. [2009] specify a data driven approach to select optimal H^*

$$H^* = \left\lceil c^* \, \hat{\xi}^{4/5} \, n^{3/5} \right\rceil, \tag{14}$$

where $c^* = 3.5134$ for the Parzen kernel and $\lceil \cdot \rceil$ represents the ceiling function. Furthermore, $\hat{\xi}^2$ is an estimate of the signal-to-noise ratio

$$\hat{\xi}^2 = \frac{\hat{\omega}^2}{\hat{IV}}. \tag{15}$$

The estimator $\hat{IV}$ serves as a proxy for the daily integrated variance. We compute it via a sparse subsampling procedure

$$\hat{IV} = RV_{\text{sparse}}, \tag{16}$$

where RV_{sparse} is the average realized variance of 20-minute returns, derived from 1-second log-price series $\{X_t\}$. This series is generated by interpolating the irregularly spaced data. That is, missing seconds are forward and backward filled with the observation from the previous or next seconds For 1200 overlapping 20-minute intervals, we define the return series as

$$x_j^{(i)} = X_{(j+1)\cdot 1200+i} - X_{j\cdot 1200+i}, \quad i = 0, \ldots, 1199, \tag{17}$$

and compute the corresponding realized variances

$$RV_{sparse}^{(i)} = \sum_j \left(x_j^{(i)}\right)^2, \tag{18}$$

such that,

$$RV_{\text{sparse}} = \frac{1}{1200} \sum_{i=0}^{1199} RV^{(i)}. \tag{19}$$

To reduce computational cost, we adopt a simplifying approximation by setting $RV_{\text{sparse}} \approx RV^{(1)}$. This is valid because RV_{sparse} is only used in (14) to compute H^*, which is rounded to the closest bigger integer. Therefore, small changes in RV_{sparse} will have a marginal effect on the computed value of the realized kernel, while the computational savings are significant.

The variable $\hat{\omega}^2$ is an estimate of the noise variance. It captures how much microstructure noise influences the observed prices. We apply a very similar procedure compared to the calculation of RV_{sparse}. In this case, we use the irregularly spaced log-prices $\{X_{\tau_j}\}$. If one just looks at returns computed over short intervals, those returns would include a lot of "true volatility", making it hard to disentangle the noise component. By taking every q_{th} observation, we construct a return series over larger gaps. This reduces the contribution of the true price process under the assumption that the signal is relatively smooth over those gaps. This makes the remaining variation in returns more likely to be a result of noise. To make this more robust, we repeat this q times, each with a different starting offset. For each starting offset $i = 1, \ldots, q$, we define the returns as

$$x_0^{(i)} = X_{\tau_q+(i-1)} - X_{\tau_0+(i-1)}, \tag{20a}$$

$$x_1^{(i)} = X_{\tau_{2q+(i-1)}} - X_{\tau_{q+(i-1)}}, \tag{20b}$$

$$\vdots$$

Each return series $\{x_j^{(i)}\}$ thus samples log-price differences at fixed lags of q observations. In accordance with Barndorff-Nielsen et al. [2009] we set $q = 25$. Then $RV_{dense}^{(i)}$ is calculated in the same manner as $RV_{sparse}^{(i)}$ in equation 18 such that we estimate $\hat{\omega}_{(i)}^2, \forall i \in \{1, \ldots, q\}$ as

$$\hat{\omega}_{(i)}^2 = \frac{RV_{\text{dense}}^{(i)}}{2n_{(i)}}, \tag{21}$$

where $n_{(i)}$ is the number of non-zero returns in the i_{th} subsample. The factor of 2 in the denominator arises because each noise component contributes to both x_j and x_{j+1}.
The final estimator of the noise variance is obtained by averaging over all q subsamples

$$\hat{\omega}^2 = \frac{1}{q} \sum_{i=1}^{q} \hat{\omega}_{(i)}^2. \tag{22}$$

Since the log-returns r_t are scaled by 100 in (8)), the kernel estimator is rescaled by a factor of $100^2 = 10{,}000$ (since it is a variance estimator). Despite both RV_t as defined in (9) and the realized kernel being both a consistent estimator of QV_t, the realized kernel has an advantage that comes directly from the utilization of higher-frequency observations. The rich data structure allows the realized kernel (RK_t) to better respond to and capture peaks. Therefore, we will use RK_t as the depended variable in further analysis.

5 Observation Driven Models

Observation–driven models form the class of traditional econometric approaches in which parameters are updated as functions of past observations and past parameter values. Examples include the GARCH and GAS specifications. As mentioned in the literature, these models are notoriously hard to outperform in volatility forecasting.

We assume r_t to have a Gaussian, Student-t, Exponential Generalized Beta distribution of the second kind (EGB2) and Asymmetric Student-t (AST) distribution. Let $\boldsymbol{\theta^{(d)}}$ denote a vector of distribution specific parameters. The full parameter vector including model parameters writes $\boldsymbol{\theta}$ such that $\boldsymbol{\theta^{(d)}} \subset \boldsymbol{\theta}$. The distribution of r_t conditional on past observation set $\mathcal{F}_{t-1}$ writes

$$r_t \mid \mathcal{F}_{t-1} \sim \mathcal{D}\big(\mu_t(\boldsymbol{\theta}, \mu_1),\ \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2), \boldsymbol{\theta^{(d)}}\big), \quad p\big(r_t \mid \mathcal{F}_{t-1}, \mu_t(\boldsymbol{\theta}, \mu_1), \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2),\ \boldsymbol{\theta^{(d)}}\big) \quad (23)$$

where $\mathcal{D}$ is one of the possible distributions and μ_1,σ_1 denote fixed but unknown initial values of the true time varying parameter. We estimate $\boldsymbol{\theta}$ by Maximum Likelihood Estimation (MLE) where $\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}$ denotes the MLE estimator. Appendix A and B describe the optimization problem and details for each distribution. We then make forecasts for horizons $h \in \mathcal{H} = \{1, 5, 10, 20\}$.

5.1 Auto-regressive Volatility Models

To model the temporal dependence in volatility, we consider auto-regressive models that capture the persistence observed in RK_t. These models express the current realized volatility as a linear function of its own past values. To apply this framework to realized-kernel estimates of quadratic variation, let $\mathcal{F}_{t-1} = \sigma\big(r_s,\ RK_s : s \leq t-1\big)$ such that $RK_t \mid \mathcal{F}_{t-1}$ follows exactly the same distribution as r_t in (23)

5.1.1 The Standard Auto-regressive Model

The auto-regressive model of order 1, or AR(1) model, originates from the work of Yule [1927] and is a foundational tool in time series analysis. The model is specified as

$$RK_t = \mu_t(\boldsymbol{\theta}) + \varepsilon_t, \quad (24a)$$
$$\mu_t(\boldsymbol{\theta}) = \mu + \phi\, RK_{t-1}, \quad (24b)$$

where $\varepsilon_t \sim \mathcal{D}(0, \sigma_\varepsilon^2, \boldsymbol{\theta})$, $\mu_t(\boldsymbol{\theta})$ is the conditional mean and $\sigma_t^2(\boldsymbol{\theta}) = \sigma_\varepsilon^2$ is the constant conditional variance. We let $\theta = (\mu, \phi, \sigma_\varepsilon^2, \boldsymbol{\theta}^{(d)})$ where $\mu \in \mathbb{R}, \sigma_\varepsilon^2 > 0$ and $\mid \phi \mid < 1$ ensuring stationarity. The AR(1) model is frequently included in studies due to its closed-form forecasting equations, and ease of estimation. However, its simplicity is also a limitation. It captures persistence at only a single lag and cannot accommodate the long-memory behavior often observed in financial volatility. Despite its limitations, the AR(1) model serves as a valuable baseline against which the performance of more flexible models can be evaluated. Appendix C.1 specifies h away forecasts.

5.1.2 The Heterogeneous Auto-regressive Model

The Heterogeneous Auto-regressive (HAR) model, introduced by Corsi [2009b], is a popular extension of the AR(1) framework designed to capture long-memory effects in financial volatility by aggregating information across multiple horizons. Specifically, it models the conditional mean $\mu_t(\boldsymbol{\theta})$ as a function of lagged daily, weekly, and monthly realized volatility

$$RK_t = \mu_t(\boldsymbol{\theta}) + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{D}\big(0, \sigma_\varepsilon^2, \boldsymbol{\theta}^{(d)}\big), \tag{25a}$$

$$\mu_t(\boldsymbol{\theta}) = \phi_0 + \phi_1 \, RK_{t-1} + \phi_2 \, RK_{t-1}^{(w)} + \phi_3 \, RK_{t-1}^{(m)}, \tag{25b}$$

where

$$RK_{t-1}^{(w)} = \tfrac{1}{5} \sum_{j=1}^{5} RK_{t-j}, \quad RK_{t-1}^{(m)} = \tfrac{1}{22} \sum_{j=1}^{22} RK_{t-j}. \tag{26}$$

Thanasoulas [2023] explains that this structure reflects how market participants may process information differently depending on their investment horizons. The recursive nature of the HAR model enables it to capture long-run volatility dynamics while remaining linear in parameters, which makes estimation and forecasting simple and intuitive. The h-away forecasts are computed in Appendix C.2

5.1.3 The Log-HAR Model

The log-HAR model, introduced by Corsi [2009b], applies the HAR framework to the logarithm of realized volatility. This transformation often improves the statistical properties of the model by stabilizing variance and reducing the impact of extreme values. Applying a logarithmic transformation often results in more normally distributed innovations, which might enhance the performance of the HAR model. The model equation writes

$$\log RK_t = \mu_t(\boldsymbol{\theta}) + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{D}\big(0, \sigma_\varepsilon^2, \boldsymbol{\theta}^{(d)}\big), \tag{27a}$$

$$\mu_t(\boldsymbol{\theta}) = \phi_0 + \phi_1 \, \log RK_{t-1}^{(d)} + \phi_2 \, \log RK_{t-1}^{(w)} + \phi_3 \, \log RK_{t-1}^{(m)}, \tag{27b}$$

where $\phi_0, \phi_1, \phi_2, \phi_3 \in \boldsymbol{\theta}$ and the lagged values are the same as in 26. The h away forecasts are defined in Appendix C.2.1

5.2 GARCH Models

While the AR(1), HAR, and log-HAR models have proven useful for forecasting realized measures of volatility, they are not originally developed to model volatility itself. Instead, they treat volatility as an observed variable and apply standard auto-regressive techniques without considering the underlying returns r_t. In contrast, the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models are specifically designed to capture volatility.

5.2.1 The GARCH Model

The standard GARCH(1,1) model, introduced by Bollerslev [1986b], is build on the assumption that asset-return volatility follows a simple auto-regressive process in squared shocks and past variance. Due to its simplicity and strong empirical performance it is one of the most popular models in modeling risk of financial assets.

The standard GARCH(1,1) model assumes that returns r_t follow

$$r_t = \sqrt{\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)}\varepsilon_t, \tag{28a}$$

$$\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2) = \omega + \alpha\, r_{t-1}^2 + \beta\sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2), \tag{28b}$$

where the condition mean $\mu_t(\boldsymbol{\theta}, \mu_1) = 0$ such that $\varepsilon_t \sim \mathcal{D}(0, 1, \boldsymbol{\theta^{(d)}})$ is an i.i.d random variable with mean 0 and variance 1. We impose $\omega > 0$, $\alpha \geq 0$, $\beta \geq 0$, and $\alpha + \beta < 1$ to ensure positive variance. These constraints also ensure that the model produces a stationarity and ergodic sequence. Furthermore, we estimate the unknown (fixed) starting value of the true filter σ_1^2 as the sample variance of the first 50 observations. For the GARCH, h step forecasts have a closed form given in Appendix C.3

5.2.2 The GARCH In Mean With Leverage Model

We now consider a nonlinear extension of the standard GARCH framework to account for leverage effects. The GARCH-in-mean with leverage model generalizes the conditional variance dynamics of the GARCH by including a nonlinear transformation of past returns and allows lagged variance to affect the mean. This model is particularly suited for capturing asymmetries and volatility feedback observed in financial returns.

The GARCH-ML model is specified as

$$r_t \;=\; \mu_t(\boldsymbol{\theta}) + \sqrt{\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)}\,\varepsilon_t \quad \varepsilon_t \;\sim\; \mathcal{D}\big(0, 1, \boldsymbol{\theta}^{(d)}\big), \tag{29a}$$

$$\mu_t(\boldsymbol{\theta}) \;=\; \mu + \lambda\,\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2), \tag{29b}$$

$$\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2) = \omega + (\alpha + \delta\tanh(-\gamma r_{t-1}))\left(\frac{r_{t-1} - \mu - \lambda\sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2)}{\sigma_{t-1}(\boldsymbol{\theta}, \sigma_1)}\right)^2 + \beta\sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2). \tag{29c}$$

The coefficient λ governs how the conditional variance feeds into the mean equation. A positive λ implies that a higher conditional variance raises the conditional mean of returns, capturing a risk premium as a result of the volatility. The parameters $\gamma \geq 0$, $\alpha > \mid \delta \mid$ control the response of volatility to shocks and allow for asymmetric effects through the nonlinear transformation $\tanh(-\gamma r_{t-1})$. The ability of capturing leverage effects directly comes from the fact that $\tanh(-\gamma r_{t-1})$ is an odd function, such that negative returns (a "bad" shock) yields a different multiplier than positive returns of the same size. The model allows us to tune the strength of the leverage effect through γ. Because of computational costs, this model is only evaluated under Normal and Student-t innovations. The h step away forecasts need to obtained through simulation where the details are given in Appendix C.4

5.2.3 The EGARCH Model

We now turn to the Exponential GARCH (EGARCH) model introduced by Nelson [1991], which extends the standard GARCH framework by modeling the logarithm of the conditional variance. The advantage of this specification is that it ensures positivity of the variance without imposing non-negativity constraints on the model parameters. The model equations write

$$r_t = \sqrt{\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)}\,\varepsilon_t, \quad \varepsilon \sim \mathcal{D}(0, 1, \boldsymbol{\theta}^{(d)}), \tag{30a}$$

$$\log(\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)) = \omega + \beta \log(\sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2)) + \alpha\big(|\varepsilon_{t-1}| - \mathbb{E}[|\varepsilon_{t-1}|]\big) + \gamma\,\varepsilon_{t-1}, . \tag{30b}$$

Because the variance is modeled in logs, $\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)$ is strictly positive without imposing non-negativity constraints on the parameters, we only require $|\beta| < 1$ to ensure the model produces stationary and ergodic sequences. In the updating equation we subtract $\mathbb{E}[|\varepsilon_{t-1}|]$ such that the asymmetric competent of the model has mean 0. That means α controls the magnitude of the standardized shock. That is, α controls how much big shocks regardless of their sign impact the volatility. The real leverage effect is captured by γ. If $\gamma > 0$, positive shocks have a larger impact on future volatility, and if $\gamma < 0$, negative shocks dominate. In contrast with the GARCH-ML model, in the EGARCH updating equation, leverage enters linearly through γ,ε_{t-1} and $\sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2)$ rather than through a bounded tanh transformation. Forecasts $h \geq 2$-step ahead are obtain through simulation and derivations of $\mathbb{E}[|\varepsilon_{t-1}|]$ are given in Appendix C.5.

5.2.4 The Bayesian GARCH Model

We extend the standard GARCH model by adopting a Bayesian framework for parameter estimation. The model preserves the same volatility dynamics as the vanilla GARCH, but parameters are estimated via Markov Chain Monte Carlo (MCMC) sampling.

The Metropolis-Hastings algorithm is a MCMC method developed by Hastings [1970] and is used to generate samples from the posterior distribution of the model parameters. Rather than optimizing the likelihood function directly, we aim to approximate the full posterior $P(\boldsymbol{\theta})$ by constructing a Markov chain whose stationary distribution is the posterior distribution. The algorithm works by proposing new parameter values and accepting or rejecting them based on how likely they are relative to the current state.

We define the model in the same way as in (C.6) and (C.7). We first estimate the parameters via maximum likelihood under the same parameter restrictions to obtain a suitable starting point for the Metropolis-Hastings algorithm. To generate efficient proposals for the posterior, it is important that the proposal distribution $Q(\boldsymbol{\theta}, \boldsymbol{\Sigma})$ is wide enough to cover the posterior distribution. This means that an appropriate estimate of the covariance matrix $\boldsymbol{\Sigma}$ is essential. We use the inverse of the Hessian $\boldsymbol{H}$ evaluated at $\hat{\boldsymbol{\theta}}_T$

$$\hat{\boldsymbol{\Sigma}}_{MLE} = -\boldsymbol{H}(\hat{\boldsymbol{\theta}_T})^{-1}. \tag{31}$$

Algorithm 1 yields a posterior chain $\{\boldsymbol{\theta^{(i)}}\}_{i=1}^N$. At the very start the chain begins at $\hat{\boldsymbol{\theta}}_T$, which may be far from where most of the high-posterior-density mass lies. The sequence $\{\boldsymbol{\theta^{(i)}}\}_{i=1}^N$ needs time to converge to the stationary distribution. Therefore, we disregard

the first $B = 1000$ draws as burn in. We calculate the parameter estimates as the mean of the posterior chain

$$\hat{\boldsymbol{\theta}}^{(\boldsymbol{i})}_{\boldsymbol{MCMC}} = \frac{1}{N-B} \sum_{i=B+1}^{N} \boldsymbol{\theta}^{(i)}. \tag{32}$$

Algorithm 1 Independence–Chain Metropolis–Hastings

Input: $\boldsymbol{\theta}^{(0)} = \hat{\boldsymbol{\theta}}_{MLE}$,$P(\theta)$,$Q(\theta)$,$\hat{\boldsymbol{\Sigma}}_{MLE}$,$N$
Output: Draws $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^{N}$

for $i \leftarrow 1$ **to** N **do**
  Draw candidate $\tilde{\theta} \sim Q(\boldsymbol{\theta}_{i-1}, \hat{\boldsymbol{\Sigma}}_{MLE})$
  Compute

$$\alpha \leftarrow \min\left(1, \frac{P(\tilde{\boldsymbol{\theta}})/Q(\tilde{\boldsymbol{\theta}})}{P(\boldsymbol{\theta^{(i-1)}})/Q(\boldsymbol{\theta^{(i-1)}})}\right) = \min\left(1, \exp\left[\ln P(\tilde{\boldsymbol{\theta}}) - \ln P(\boldsymbol{\theta^{(i-1)}})\right] \frac{Q(\boldsymbol{\theta^{(i-1)}})}{Q(\tilde{\boldsymbol{\theta}})}\right).$$

  Draw $u \sim \mathcal{U}(0,1)$ **if** $u \leq \alpha$ **then**
    $\boldsymbol{\theta^{(i)}} \leftarrow \tilde{\boldsymbol{\theta}}$
  else
    $\boldsymbol{\theta^{(i)}} \leftarrow \boldsymbol{\theta^{(i-1)}}$
  end
end
return $\{\boldsymbol{\theta^{(i)}}\}_{i=1}^{N}$

Using the posterior mean estimates, the one-step-ahead conditional variance forecasts are made recursively using Equations C.6 and C.7. For computational reasons we only consider the Normal and Student-t densities as candidate distributions.

5.2.5 GARCH–MIDAS Model

In the GARCH–MIDAS framework of Engle et al. [2013] we extend the GARCH model by decomposing the daily conditional variance $\sigma_t^2(\boldsymbol{\theta}, g_1, \tau_1)$ of the log-returns r_t into a short-run GARCH(1,1) component $g_t(\boldsymbol{\theta}, g_1)$ and a long-run low-frequency component $\tau_t(\boldsymbol{\theta}, \boldsymbol{x}_t, \tau_1)$. Specifically,

$$r_t = \mu_t(\boldsymbol{\theta}) + \sqrt{\sigma_t^2(\boldsymbol{\theta}, g_1, \tau_1)}\,\varepsilon_t, \quad \varepsilon \sim \mathcal{D}(0, 1, \boldsymbol{\theta}^{(d)}), \tag{33a}$$

$$\sigma_t^2(\boldsymbol{\theta}, g_1, \tau_1) = \tau_t(\boldsymbol{\theta}, \boldsymbol{x}_t, \tau_1)\, g_t(\boldsymbol{\theta}, g_1), \tag{33b}$$

$$g_t(\boldsymbol{\theta}, g_1) = (1 - \alpha - \beta) + \alpha \frac{(r_{t-1} - \mu_t(\boldsymbol{\theta}))^2}{\tau_{t-1}(\boldsymbol{\theta}, \boldsymbol{x}_t, \tau_1)} + \beta g_{t-1}(\boldsymbol{\theta}, g_1), \tag{33c}$$

$$\tau_t(\boldsymbol{\theta}, \boldsymbol{x}_t, \tau_1) = m^2 + \lambda^2 \sum_{k=1}^{K-1} \phi_k(\omega)\, \boldsymbol{x}_{j(t)-k} = m^2 + \lambda^2\, \boldsymbol{\phi}(\omega)'\, \mathbf{x}_t, \tag{33d}$$

where $\boldsymbol{x_t}$ is a vector containing K lags of CPI inflation and the conditional mean $\mu_t(\boldsymbol{\theta}) = \mu$ is not time varying. Following van Driel [2019], the short-run volatility $g_t(\boldsymbol{\theta}, g_1)$ obeys the standard GARCH recursion under the usual parameter constraints. The short-run

component reacts immediately to shocks one day before. The power of the GARCH-MIDAS comes from that the short run GARCH component is complemented with the long-term component $\tau_t(\boldsymbol{\theta}, \boldsymbol{x}_t, \tau_1)$ that aggregates monthly CPI inflation via the Beta-lag weights

$$\phi_k(\omega) = \frac{(k/K)^{\omega-1}\,(1-k/K)^{\omega-1}}{\sum_{j=1}^{K-1}(j/K)^{\omega-1}(1-j/K)^{\omega-1}}, \quad k = 1, \ldots, K-1, \tag{34}$$

where large shocks in inflation receive greater weight. When inflation rises, this drives up the conditional volatility through the long-term component. When inflationary pressures decrease, $\tau_t(\boldsymbol{\theta}, \boldsymbol{x}_t, \tau_1)$ decreases $\sigma_t^2(\boldsymbol{\theta}, g_1, \tau_1)$, even in the presence of high volatility in the short-run. This joint modeling is what possibly leads to more accurate volatility forecasts. Details on how we construct $\boldsymbol{x}_t$ and make forecasts is presented in Appendix C.6.

5.3 Realized GARCH Models

Realized GARCH models enhance the standard GARCH framework by incorporating a measurement equation that links the conditional variance to the observed realized measure RK_t.

5.3.1 Linear Realized GARCH

We include the linear Realized GARCH specification of Hansen and Lunde [2011] as the simplest model in this category of realized models. The linear Realized GARCH model writes

$$r_t = \sqrt{\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)}\,\varepsilon_t, \quad \varepsilon \sim \mathcal{D}(0, 1, \boldsymbol{\theta^{(d)}}) \tag{35a}$$

$$\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2) = \omega + \beta\,\sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2) + \gamma\,RK_{t-1}, \tag{35b}$$

$$RK_t = \xi + \varphi\,\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2) + w_t, \tag{35c}$$

where $w_t = \tau(\varepsilon_t) + u_t$ with $\tau(\varepsilon_t) = \tau_1\,\varepsilon_t + \tau_2\,(\varepsilon_t^2 - 1)$ captures the leverage effect. Furthermore, innovations ε_t and $u_t \sim \mathcal{N}(0, \sigma_u^2)$ are independent. Since we always take u_t to be normal we include σ_u as a model parameter such that parameter vector writes $\boldsymbol{\theta} = (\omega, \beta, \gamma, \xi, \varphi, \tau_1, \tau_2, \sigma_u, \boldsymbol{\theta_0^{(d)}})$,. We need $\omega > 0$, $\beta \geq 0$, $\gamma \geq 0$, $\xi > 0$, $\varphi \geq 0$, $\sigma_u > 0$, and $\beta + \varphi\,\gamma < 1$ to ensure the process is positive and produces stationary and ergodic sequences. The additional measurement equation provides extra information about unconditional volatility shocks and observation noise, which may lead to improved out-of-sample volatility forecasts. Hansen and Lunde [2011] show the h away forecasts have a closed form by rewriting the model to a VARMA(1,1) and are given in Appendix C.7.1.

5.3.2 Log-Linear Realized GARCH

Hansen et al. [2011b] also define a log-linear extension where $\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)$ and RK_t get replaced with their logarithms

$$r_t = \sqrt{\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)}\varepsilon_t, \quad \varepsilon \sim \mathcal{D}(0, 1, \boldsymbol{\theta^{(d)}}), \tag{36a}$$

$$\log(\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)) = \omega + \beta\,\log(\sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2)) + \gamma\,\log(RK_{t-1}), \tag{36b}$$

$$\log(RK_t) = \xi + \varphi \log(\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)) + w_t. \tag{36c}$$

Similar to the difference between the HAR and log-HAR model,taking logs of $\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)$ and RK_t suppresses the effect of large outliers and automatically keeps the conditional variance forecasts positive. This allows ω and ξ to remain unrestricted. The h step-away-forecasts are defined in Appendix C.7.2.

5.3.3 Realized EGARCH

Building on the Realized GARCH framework, the Realized EGARCH model proposed again by Hansen et al. [2011b], extends this approach in combination with the EGARCH model from Nelson [1991] and is specified as follows

$$r_t = \sqrt{\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)}\varepsilon_t, \quad \varepsilon \sim \mathcal{D}(0, 1, \boldsymbol{\theta^{(d)}}), \tag{37a}$$

$$\log(\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)) = \omega + \beta \log(\sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2)) + w_t, \tag{37b}$$

$$\log(RK_t) = \xi + \log(\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)) + u_t, \tag{37c}$$

where $w_t = \tau(\varepsilon_{t-1}) + \delta u_{t-1}$ such that the leverage function appears directly into the volatility equation. Compared to the linear and log-linear GARCH specifications, we set $\varphi = 1$ in the measurement equation. This means that we are treating the realized variance $\log(RK_t)$ as an unbiased proxy for $\log \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)$. Instead of φ, we introduce δ in the updating equation which measures how much unexpected moves in RK_t (the innovation u_{t-1}) get carried into the conditional volatility one step later. Similar as before, we have $u_t \sim \mathcal{N}(0, \sigma_u^2)$ independent of ε_t. Using a similar strategy outlined in appendix C.5 we define the h step-away-forecasts using the VAR(1) companion form. That is, the inclusion of the measurement equation makes it such that we do not have to rely on Monte Carlo simulation as in the regular EGARCH model.

5.4 Generalized Autoregressive Score Models

Generalized Autoregressive Score (GAS) models, also known as score-driven models, update all time-varying parameters directly using the scaled score of the conditional likelihood. [Creal et al., 2013].

5.4.1 GAS Model

The first of such GAS models, is developed by Creal et al. [2013]. We set the unconditional mean $\mu_t(\boldsymbol{\theta}, \mu_1)$ equal to 0 such that the model specification writes

$$r_t \mid \mathcal{F}_{t-1} \sim \mathcal{D}(0, \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2), \boldsymbol{\theta^{(d)}}), \tag{38a}$$

$$\log \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2) = \omega + \beta \log \sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2) + \alpha\, s_{t-1}, \tag{38b}$$

$$s_t = S_t \cdot \nabla_t = \frac{\partial \log p\big(r_t \mid \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2), \mathcal{F}_{t-1}, \boldsymbol{\theta}\big)}{\partial \log \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)}, \tag{38c}$$

where s_t is the product of a scaling factor $S_t \equiv \mathbb{E}(\nabla_t^2)^{-1}$ and the score of the log-likelihood ∇_t evaluated at r_t. The main advantage of this model is that under heavy-tailed or skewed

$\mathcal{D}$ the update is naturally more robust to outliers or asymmetry. This property comes from the fact that if $\mathcal{D}$ has fat tails, the tails "flatten out" evenly such that the derivative of the log density will slowly degrade instead of explode. Therefore, ∇_t, and by extension, update s_t grow more slowly for large $|r_t|$, which makes the GAS update less sensitive to extreme observations.

Obviously, the score ∇_t depends on our choice of $\mathcal{D}$ and are derived in appendix C.8, as well as the h step away forecasts that needs to be obtained simulation.

5.5 Realized GAS Model

We enhance the standard GAS framework by adding the realized measure RK_t. Simarly as for the GARCH extensions, we add RK_t to the usual GAS update such that the model can utilize the actual volatility we just observed, not just past returns. Following the score-driven approach of Gorgi et al. [2019], we write the model as

$$RK_t \sim \Gamma\Big(\tfrac{\nu}{2}, \tfrac{2\sigma_t^2(\boldsymbol{\theta},\sigma_1^2)}{\nu}\Big). \tag{39}$$

This specification lets realized and return information jointly update the log-variance in a unified GAS framework. This is a clear illustration of the additional flexibility GAS models offer. We exploit this by employing a modified version of the Realized GAS model, where we replace the Gamma distribution with a Weibull distribution for RK_t. That is,

$$RK_t \sim \text{Weibell}\left(\frac{\nu}{2}, \frac{2\sigma_t^2(\boldsymbol{\theta},\sigma_1^2)}{\nu}\right). \tag{40}$$

The exact form of the updating equation and maximization problem are presented in appendix C.9

5.5.1 GAS–MIDAS Model

The GAS–MIDAS model of van Driel [2019] nests a low-frequency MIDAS component for the long-run variance into a standard score-driven (GAS) update for the short-run factor. As in Section 5, we set the unconditional mean to zero and write

$$r_t \mid \mathcal{F}_{t-1} \sim \mathcal{D}\big(0, \sigma_t^2(\boldsymbol{\theta}, \tau_1, g_1), \boldsymbol{\theta}^{(d)}\big), \tag{41a}$$

$$\sigma_t^2(\boldsymbol{\theta}, \tau_1, g_1) = \tau_t\big(\boldsymbol{\theta}, \boldsymbol{x}_t, \tau_1\big) g_t(\boldsymbol{\theta}, g_1), \tag{41b}$$

$$\nabla_t = \frac{\partial}{\partial \log g_t(\boldsymbol{\theta}, g_1)} \log p\big(r_t \mid \ 0, \tau_t\big(\boldsymbol{\theta}, \boldsymbol{x}_t, \tau_1\big) g_t(\boldsymbol{\theta}, g_1), \mathcal{F}_{t-1}, \boldsymbol{\theta}\big) \tag{41c}$$

$$S_t = \mathbb{E}\big[\nabla_t^2 \mid \mathcal{F}_{t-1}\big]^{-1}, \tag{41d}$$

,

$$s_t = S_t \nabla_t \tag{41e}$$

$$\log g_t(\boldsymbol{\theta}, g_1) = \omega + \beta \log g_{t-1}(\boldsymbol{\theta}, g_1) + \alpha s_{t-1}. \tag{41f}$$

Compared to a standard GAS model, GAS–MIDAS incorporates low-frequency macro signals, such that it adapts not only to yesterday's return but also to broader economic shifts in macro-indicators such as inflation. As explained, the update s_t in the GAS-MIDAS automatically suppresses extreme returns under heavy tails or skewness. We derive S_t, ∇_t, $\log g_t(\boldsymbol{\theta}, g_1)$ and h step forecasts in appendix C.10.

6 Parameter Driven Models

Observation-driven models such as GARCH and GAS treat volatility as a deterministic function of past data. That is, the parameters are deterministic functions of lagged dependent variables and contemporaneous variables. Parameters evolve randomly over time but are perfectly predictable one-step-ahead given past information. In contrast, in parameter-driven models, parameters vary over time as dynamic processes with their own innovations. We include this class of models to establish which is more effective in forecasting realized volatility.

6.1 Stochastic Volatility Model

The Stochastic Volatility (SV) model, as introduced by Durbin and Koopman [2012] has a solid foundation which originates from theoretical models for asset price processes that are used for many popular option pricing models like the Black-Scholes model of Black and Scholes [1973]. More specifically, we begin with the continuous-time log-price model of Equation (1) without jumps $J(t)$ and assume $\log \sigma_t^2$ follows an Ornstein-Uhlenbeck process

$$\log \sigma(t)^2 = \xi + H(t), \quad dH(t) = -\lambda\, H(t)\, dt + \sigma_\eta\, dB(t), \tag{42}$$

where ξ is constant, $\lambda \in (0, 1)$ and B(t) a standard Brownian motion independent of W(t) in (1). Furthermore, $\sigma_n > 0$ is a coefficient which can be thought of representing the "volatility of volatility". Applying the Euler-Maruyama discretisation method will lead to the SV model specified as

$$r_t = \mu + \sigma \exp\left(\frac{\tilde{s}_t}{2}\right) u_t, \tag{43a}$$

$$\tilde{s}_{t+1} = \phi \tilde{s}_t + \eta_t, \tag{43b}$$

where $u_t \sim \mathcal{NID}(0, 1)$ and $\eta_t \sim \mathcal{NID}(0, \sigma_\eta^2)$, $0 < \phi < 1$ to maintain stationarity in the state equation, $\sigma \in \mathbb{R}$ and $\mu \in \mathbb{R}$. Here, $\tilde{s}_t$ represents the state process which evolves as a stationary autoregressive process. The equation of r_t and $\tilde{s}_t$ are referred to as the observation equation and state equation respectively.

Because the observation equation in (43) is non-linear in the state $\tilde{s}_t$ the exact likelihood involves high-dimensional integrals and cannot be evaluated in closed form. Consequently, the usual Kalman Filter (KF) and Kalman Filter Smoother (KFS) techniques cannot be used to derive filtered estimates of $\tilde{s}_t$ which are also needed to deploy MLE for parameter estimation. This forces us to perform Quasi-Maximum Likelihood Estimation.

6.1.1 Kalman Filter Quasi-Maximum Likelihood

We introduce the following transformation of r_t

$$z_t = \log\left((r_t - \mu)^2\right), \tag{44}$$

to construct a new observation equation

$$z_t = \kappa + \tilde{s}_t + e_t, \quad \text{with} \quad e_t \sim p(e_t), \tag{45}$$

where the transformed observation equation is now linear in the state $\tilde{s}_t$. Since $u_t \sim \mathcal{N}(0,1)$ implies $u_t^2 \sim \chi^2(1)$, we have the error term $e_t = \log u_t^2 \sim \log \chi^2(1)$ which has mean -1.27 and variance $\pi^2/2$. To address the non-Gaussianity of e_t, we use the quasi maximum likelihood (QML) method introduced by Harvey et al. [1994]. This method works by wrongfully specifying e_t as Gaussian. The transformed model writes

$$z_t = -1.27 + s_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \pi^2/2), \tag{46a}$$

$$s_{t+1} = \psi + \phi s_t + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma_\eta^2). \tag{46b}$$

The state equation in (46) now incorporates a mean adjustment to align with the Gaussian assumption for ε_t and note that $s_t = \kappa + \tilde{s}_t$ and $\psi = \kappa(1-\phi)$. Together (46) forms a linear Gaussian state space model, which allows us to apply the KF to obtain the filtered states and use MLE to estimate the parameter set $\boldsymbol{\theta} = \{\psi, \phi, \sigma_\eta\}$. The KF algorithm and closed form forecast equations are given in Appendix C.11

6.1.2 Bootstrap Particle Filter

While the data transformation and Gaussian assumption in the previous section allows for the use of the Kalman Filter, this is not the exact specification of the model. The Bootstrap Particle Filter overcomes this limitation by employing a simulation-based approach. This allows for a more accurate estimation of the latent states. This method is a Sequential Monte Carlo (SMC) approach designed to handle state-space models where the observation equation is nonlinear, such as the SV model in (43).

The QML estimates for ϕ, ψ and σ_η^2, are obtained as in the previous section and μ is given by $\mu = \frac{\psi}{1-\phi}$. The Bootstrap Particle Filter then estimates the latent states as described in Algorithm 2.

This method uses $M = 1000$ particles over the entire time period. The first state at $t = 1$ is initialized by drawing from the unconditional distribution using (47). Then, for each t, M particles $\tilde{s}_t^{(i)}$ are drawn from the state transition equation (48). These sampled states serve as a Monte Carlo approximation to the true distribution of the latent state. The simulation randomness in this step is what the Bootstrap Particle Filter allows to capture the full range of possible states. Under the observation equation r_t follows (49). One might notice that σ^2 and the distribution specific parameters $\boldsymbol{\theta}^{(d)}$ are assumed known and estimated from r_t. The importance weights are then given by the likelihood values of observing r_t under this distribution. In (50) the weights are normalized to sum to 1.

The estimate of the filtered state at time t is subsequently obtained as the weighted sum of the particles in (51).

Lastly, to avoid particle degeneracy the algorithm executes systematic resampling. This means M new independent particles $s_t^{(i)}$ are drawn from the set $\{\tilde{s}_t^{(1)}, \ldots, \tilde{s}_t^{(M)}\}$ with replacement, using the probabilities $\{w_t^{(1)}, \ldots, w_t^{(M)}\}$. This is necessary such that high-weight particles are retained and low-weight particles are replaced.

Algorithm 2 Bootstrap Particle Filter for Stochastic Volatility

Input: QML estimates $\hat{\phi}$, $\hat{\sigma}_\eta^2$, $\hat{\mu}$, known σ^2, # particles M, series length T, data $\{r_t\}_{t=1}^T$.
Output: Filtered state estimates $\{\hat{s}_t\}_{t=1}^T$.

for $i \leftarrow 1$ **to** M **do**
  Initialize particle i**:**

$$s_1^{(i)} \sim \mathcal{N}\Big(0, \frac{\hat{\sigma}_\eta^2}{1-\hat{\phi}^2}\Big). \tag{47}$$

end
for $t \leftarrow 2$ **to** T **do**
  for $i \leftarrow 1$ **to** M **do**
    Propagate particle i**:**

$$\tilde{s}_t^{(i)} \sim \mathcal{N}\big(\hat{\phi}\, s_{t-1}^{(i)}, \hat{\sigma}_\eta^2\big). \tag{48}$$

  end
  for $i \leftarrow 1$ **to** M **do**
    Weight particle i**:**

$$\tilde{w}_t^{(i)} \sim \mathcal{D}\Big(\mu, \sigma^2 \exp\big(\frac{\tilde{s}_t}{2}\big), \boldsymbol{\theta^{(d)}}\Big) \tag{49}$$

  end
  Normalize weights:

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^M \tilde{w}_t^{(j)}}, \quad i = 1, \ldots, M. \tag{50}$$

  Estimate filtered state:

$$\hat{s}_t = \sum_{i=1}^M w_t^{(i)}\, \tilde{s}_t^{(i)}. \tag{51}$$

  Resample particles: Draw $s_t^{(i)}$ with replacement from $\{\tilde{s}_t^{(j)}\}_{j=1}^M$ using weights $\{w_t^{(j)}\}_{j=1}^M$.
end
return $\{\hat{s}_t\}_{t=1}^T$

The main advantage of the bootstrap particle filter is that it works directly with the true SV observation equation. By utilizing simulation techwniques, it captures nonlinear features that the Kalman-filter QML is not able to model. In practice this yields more reliable state estimates and a better fit when the returns are far from linear and Gaussian.

7 Machine Learning

Up to now, we have focused observation and parametric approaches. These methods offer clear economic interpretation and rely on explicit likelihood formulations. To better

capture complex potentially nonlinear relationships about future volatility, we transition into the realm of supervised learning algorithms. These methods are more data-driven, focusing on flexible pattern recognition as the driver of predictive performance.

Similarly as before, we define $\boldsymbol{\theta}$ as the parameter vector. We define a general learning algorithm as $f_{\boldsymbol{\theta}} : \mathbb{R}^P \to \mathbb{R}^{\mathcal{H}}$, where we often omit subscript $\boldsymbol{\theta}$. Because such machine learning models generally do not make any assumptions on the distribution of the data, the distribution parameter specific vector $\boldsymbol{\theta}^{(d)}$ is replaced by a vector of hyperparameters $\boldsymbol{\theta}^{(hp)} \subset \boldsymbol{\theta}$.

7.1 Market Sentiment Extraction

To reach our goal of obtaining of obtaining accurate volatility forecasts, we incorporate market sentiment by utilizing information that is not contained in past price movements. We use news headlines to model shifts in the mood and risk appetite of investors which can drive abrupt changes in realized volatility.

To scrape news articles we utilize the news repository of Business Insider [2025]. For each stock, we collect all articles between 2018-01-03 and 2025-02-27 and extract the title, as well as the publication time. Next, we use the Financial BERT NLP model, developed by Araci [2019] to assess whether the article is positive, negative or neutral, solely based on the headline.

Financial BERT is an adapted variant of the original Bidirectional Encoder Representations from Transformers (BERT) model specifically tuned for financial text. The architecture of BERT consists of 12 Transformer encoder layers and 12 self-attention heads with a hidden size $d = 768$. FinBERT undergoes two phases of additional training. First, the base BERT weights are further pretrained on a large collection of financial documents common in finance. The pretrained Financial BERT is then fine-tuned on the Financial PhraseBank of Malo et al. [2014],, where each headline is prepended with the special token $[CLS]$, appended with $[SEP]$, and fed through the 12 encoder layers. The final hidden state at the $[CLS]$ position is passed through a linear classification layer plus softmax to produce probabilities for the three sentiment classes. In later sections, we discuss the architecture of Transformers and multi-head-attention in more detail. Figure 1 shows a graphical illustration of the second training phase of the model.

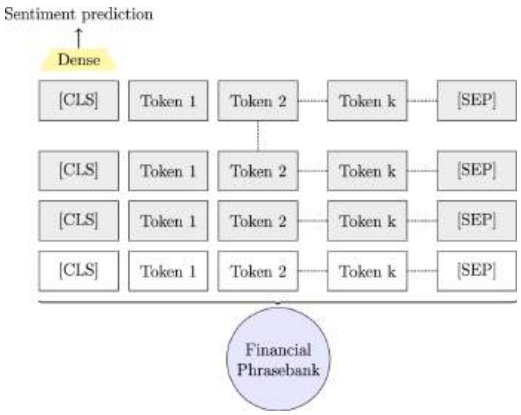


Figure 1: Illustration of Financial BERT architecture from Araci [2019].

To obtain a daily sentiment classification for each stock, we apply a majority voting rule. Headlines are grouped by calendar date and if one label appears more than the others it wins such that that day is labeled, for example, "positive". If there is a tie, we pick the headline withe highest average confidence score. Finally, labels are mapped to to $\{0, 1, 2\}$ to create a categorical variable, which can be utilized as a feature in the machine learning models. Because Business Insider [2025] did not contain an article for each stock for each day and not all machine learning models can handle missing values, we have to perform imputation of the missing categorical data. We start by initialization of leading missing values by the empirical marginal distribution. Next, we exploit the temporal dependence of the series via a first-order Markov-chain technique, which has been used earlier in statistical analysis of time series with missing data by Zhu and Galbraith [2010b] and Zucchini et al. [2016] and explained in Appendix D.0.1.

7.2 Feature Creation

To capture both the magnitude and persistence of market moves, we begin by transforming the raw log-return r_t and realized kernel RK_t. At each time t we compute the absolute return, squared return, and log-kernel

$$|r_t|, \quad r_t^2, \quad \log\big(RK_t\big). \tag{52}$$

These transformations highlight volatility spikes and nonlinear effects. Next, we include up to L lagged values of each series to enable models to "see" time dependent effects. In particular, for $i = 1, \ldots, L$ we add

$$|r_{t-i}|, \quad r_{t-i}^2, \quad RK_{t-i}. \tag{53}$$

Finally, to capture longer-term trends, we compute rolling statistics over a set of windows $\mathcal{W} = \{5, 20, 60\}$. For each window $w \in \mathcal{W}$, we calculate the w-day moving average and standard deviation of the same series. For example, for the log-return

$$\overline{r}_t^{(w)} = \frac{1}{w}\sum_{i=0}^{w-1} r_{t-i}, \quad \sigma_r^{(w)}(t) = \sqrt{\frac{1}{w}\sum_{i=0}^{w-1}\big(r_{t-i} - \overline{r}_t^{(w)}\big)^2}, \tag{54}$$

which we repeat for $|r_t|$, r_t^2, and $\log(RK_t)$. In addition, we normalize the current realized kernel by its 20-day average

$$\frac{RK_t}{\overline{RK}_t^{(20)}}, \tag{55}$$

providing a scale-invariant measure of recent volatility.

Since our sentiment series $MS_t \in \{0, 1, 2\}$, we first convert it into three binary indicators via one-hot encoding

$$s_{j,t} = \mathbf{1}\{s_t = j\}, \quad j \in \{0, 1, 2\}. \tag{56}$$

To allow for time dependencies in sentiment we include up to L lags, as in (53) and consider sentiment over recent history $p_{c,t}^{(w)}$ in the same manner as in (54). Finally, for each window we define the Shannon entropy

$$H_t^{(w)} = -\sum_{c=0}^{2} p_{c,t}^{(w)} \log\big(p_{c,t}^{(w)}\big), \tag{57}$$

developed by Shannon [1948] which quantifies how many times sentiment switches or remains constant per rolling window.

We drop any rows with missing values from lags or rolling windows. We denote the training sample size as $N > \max(L, \max\{\mathcal{W}\})$. Let $\mathbf{X} \in \mathbb{R}^{N \times P}$ denote the matrix of the P features. We define the target vector as $\boldsymbol{Y} \in \mathbb{R}^{N \times |\mathcal{H}|}$ such that $\boldsymbol{Y}$ has one column per horizon

$$\mathbf{Y}^{(\mathbf{j})} = \big[\, RK_{t+h_j} \big]_{t=1,\ldots,N} \quad \forall j \in \{1, \ldots \max|\mathcal{H}|\}, \tag{58}$$

such that

$$\mathbf{Y} = \big[Y^{(1)}, Y^{(2)}, \ldots, Y^{(|\mathcal{H}|)}\big]. \tag{59}$$

As a last step, we standardize $\boldsymbol{X}$ and $\boldsymbol{Y}$ to have zero mean and unit variance.

7.2.1 Hyperparameter Optimization

Let $\boldsymbol{\theta}^{(\mathrm{hp})} = (\theta_1, \theta_2, \ldots, \theta_m)'$ denote the vector of m hyperparameters for a given learning algorithm. Each θ_i takes values in a possibly unbounded domain Θ_i, so the full hyperparameter space is

$$\Theta^{\mathrm{hp}} = \Theta_1 \times \Theta_2 \times \cdots \times \Theta_m, \tag{60}$$

which may also be unbounded. With grid searching, we choose finite candidate sets $\Theta_i^* \subset \Theta_i$ for each hyperparameter. This allows us to form the grid

$$\mathcal{G} = \Theta_1^* \times \Theta_2^* \times \cdots \times \Theta_m^*. \tag{61}$$

We begin by selecting an initial training set $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$ on which we apply Algorithm 3

Algorithm 3 Time Series Cross-Validation

Input: Length initial training data N_{train}, number of splits K.
. **Output:** Sequence of (train_idx, test_idx) pairs.
$n_{\text{test}} \leftarrow \left\lfloor \frac{N_{train}}{K+1} \right\rfloor$
for $k \leftarrow 1$ **to** K **do**
    train_end $\leftarrow k \times n_{\text{test}}$
    test_start $\leftarrow$ train_end $+ 1$
    test_end $\leftarrow (k+1) \times n_{\text{test}}$
    train_idx $\leftarrow [1, \ldots,$ train_end$]$
    test_idx $\leftarrow [$test_start$, \ldots,$ test_end$]$
end
return (train_idx, test_idx)

which splits the training sample as $\{(\mathbf{X}_{\text{train}}^{(k)}, \mathbf{Y}_{\text{train}}^{(k)}), (\mathbf{X}_{\text{val}}^{(k)}, \mathbf{Y}_{\text{val}}^{(k)})\}_{k=1}^{K}\}$, with n_k observations per validation split such that $\mathbf{X}_{\text{val}}^{(k)} \in \mathbb{R}^{n_k \times P}$ and $\mathbf{Y}_{\text{val}}^{(k)} \in \mathbb{R}^{n_k \times |\mathcal{H}|}$. Then for every

$\boldsymbol{\theta^{(hp)}} \in \mathcal{G}$, we find $\hat{f}^{(k)}_{\boldsymbol{\theta}^{(\text{hp})}}$, which is the fitted model on $(\mathbf{X}^{(k)}_{\text{train}}, \mathbf{Y}^{(k)}_{\text{train}})$ using $\boldsymbol{\theta^{(hp)}}$. This allows us to compute predictions on the validation features using $\boldsymbol{\theta}^{(hp)} \subset \boldsymbol{\theta}$

$$\hat{\mathbf{Y}}^{(k)}_{\text{val}} = \hat{f}^{(k)}\big(\mathbf{X}^{(k)}_{\text{val}}, \boldsymbol{\theta}\big), \tag{62}$$

such that the validation loss on fold k writes

$$\ell_k\big(\boldsymbol{\theta}\big) = \frac{1}{n_k}\big\|\mathbf{Y}^{(k)}_{\text{val}} - \hat{\mathbf{Y}}^{(k)}_{\text{val}}\big\|^2_F, \tag{63}$$

where $\|\cdot\|_F$ denotes the Frobenius norm such that $\ell_k\big(\boldsymbol{\theta}\big)$ represents the square loss across all validation observations and horizons $h \in \mathcal{H}$. We let $\hat{\boldsymbol{\theta}}^{(\text{hp})} \in \mathcal{G}$ be the hyper parameter vector that minimizes the squared loss on average across all folds

$$\hat{\boldsymbol{\theta}}^{(\text{hp})} = \arg\min_{\boldsymbol{\theta}^{(\text{hp})} \in \mathcal{G}} \frac{1}{K}\sum_{k=1}^{K} \ell_k\big(\boldsymbol{\theta}^{(\text{hp})}\big). \tag{64}$$

The grid search suffers from the curse of dimensionality, as $|\mathcal{G}|$ grows exponentially in the number of hyperparameters. Therefore, to put constraints on the computational costs, the grid search optimizes the average validation loss across all forecast horizons at once. Despite the heavy computational burden grid searching selects values for the hyper parameters which are data-driven [Pedregosa et al., 2011].

7.3 Shrinkage Methods

Shrinkage estimators belong to the broad field of statistical learning, which seeks a compromise between purely parametric approaches and highly flexible algorithmic methods. Shrinkage methods represent a framework similar to Ordinary Least Squares (OLS) by being linear in parameters and computationally efficient, but with the extra flexibility to down-weight or eliminate irrelevant covariates. The penalized OLS objective writes

$$(\beta^*_{0,j},\, \boldsymbol{\beta}^*_j) \;=\; \arg\min_{\beta_{0,j}, \boldsymbol{\beta}_j} \left\{ \frac{1}{N}\left\|\mathbf{Y}^{(j)} - \beta_{0,j}\,\mathbf{1} - \mathbf{X}\,\boldsymbol{\beta}_j\right\|^2_2 \;+\; \lambda\mathcal{P}(\boldsymbol{\beta}_j)\right\} \quad j = 1, \ldots, |\mathcal{H}|, \tag{65}$$

such that $\boldsymbol{\theta}_j = (\beta_{0,j},\, \boldsymbol{\beta}_j)$ where $\lambda \in \boldsymbol{\theta}^{(hp)}$ is a hyperparameter and $\beta_{1,j}, \ldots, \beta_{P,j})'$ are the slope coefficients for horizon j.The fitted model predicts the future response as a linear combination of the current features, with coefficients shrunk or zeroed out according to the chosen penalty $\mathcal{P}(\boldsymbol{\beta}^*_j)$.

7.3.1 LASSO (L1) Regression

The Least Absolute Shrinkage and Selection Operator (LASSO) developed by Tibshirani [1996] applies an L_1 penalty to the slope coefficients $\boldsymbol{\beta}_j$ for each horizon j

$$\mathcal{P}(\boldsymbol{\beta}_j) = \sum_{i=1}^{P}\big|\beta_{i,j}\big|, \tag{66}$$

where λ controls the strength of the penalty. The absolute-value penalty is convex but non-differentiable at zero, which causes it to shrink some coefficients to exactly zero. This

variable selection capability is most useful in high-dimensional settings where $P \gg N$ but is also potentially useful in removing highly correlated variables such moving-average predictors and reduce overfitting. However, it might not handle extremely high collinearity well. In such cases, other techniques like Ridge regression or Elastic Net might be more appropriate [IBM, 2024].

7.3.2 Ridge (L2) Regression

Ridge regression developed by [Hoerl and Kennard, 1970] is similar to the LASSO but augments the least-squares criterion with a slightly different L_2 penalty on the slope coefficients. For each forecast horizon j

$$\mathcal{P}(\boldsymbol{\beta}_j) = \sum_{i=1}^{P} \beta_{i,j}^2 \tag{67}$$

is strictly convex and differentiable, which ensures a unique analytic solution of (65) given by

$$\boldsymbol{\beta}_j^* = (\mathbf{X}'\mathbf{X} + \lambda\,\mathbf{I})^{-1}\mathbf{X}'\boldsymbol{Y}^{(j)}. \tag{68}$$

Unlike the LASSO, Ridge does not perform variable selection. It shrinks all coefficients continuously toward zero but never exactly to zero. This continuous shrinkage can yield better forecasts when predictors are highly correlated. When the features in $\mathbf{X}$ are highly correlated, $\mathbf{X}'\mathbf{X}$ becomes ill-conditioned. The hyperparameter $\lambda \geq 0$ in (65) serves as the constant shifting the diagonals of the design matrix. Effectively it shifts the eigenvalues of $\mathbf{X}^\top\mathbf{X}$ away from zero which results in a stable system with a unique solution of the minimization problem. Ridge regression is especially effective when many predictors have small but nonzero effects.

7.3.3 Elastic Net

The Elastic Net of [Zou and Hastie, 2005] combines the strengths of both LASSO and Ridge by introducing a convex combination of the L_1 and L_2 penalties. For each forecast horizon j the penalty writes

$$\mathcal{P}_{\text{enet}}(\boldsymbol{\beta}_j) = \alpha \sum_{i=1}^{P} \left|\beta_{i,j}\right| \;+\; (1-\alpha) \sum_{i=1}^{P} \beta_{i,j}^2. \tag{69}$$

where $\alpha \in [0, 1]$ is an additional hyperparameter. It is particularly useful when predictors are highly correlated where the L_2 component stabilizes the solution, while the L_1 component simultaneously removes features to reduce multicolinearity issues.

7.4 Random Forest

We seek further improvements in forecasting accuracy compared to shrinkage methods by considering Random Forest (RF). RF is a highly non linear ensemble learning method introduced by Breiman [2001] that constructs a large collection of decision trees and aggregates their predictions to produce a single forecast. A decision tree is a nonparametric model that partitions the predictor space into disjoint regions which are called terminal

nodes or leaves. It assigns to each region the average of the training targets within that region. Formally, for each training index i we have a feature vector $\mathbf{X}_i \in \mathbb{R}^P$ and for a fixed forecast horizon j, target y_{i+h_j}. A single tree T_b partitions $\mathbb{R}^P$ into regions $\{R_{b,j,m}\}_{m=1}^{M_b}$. The prediction of the tree at a new point $\mathbf{X}_t$ writes

$$\hat{f}_j^{(b)}(\mathbf{X}_t) \;=\; \frac{1}{|R_{b,j}(t)|} \sum_{i \in R_{b,j}(t)} y_{i+h_j}, \tag{70}$$

where

$$R_{b,j}(t) = \big\{\, i : \mathbf{X}_i \text{ and } \mathbf{X}_t \text{ fall in the same terminal node of } T_b \big\}. \tag{71}$$

The tree is grown by choosing splits on predictors and split-points to minimize the sum of squared errors in each child node. The tree stops until a stopping criterion is met such as minimum node size or maximum depth. These function as the hyperparameters collected in $\boldsymbol{\theta}^{(hp)}$.
Although decision trees are intuitive and can capture complex nonlinearities, they have high variance when grown to full depth and may suffer from overfitting.

Bagging is a powerful technique designed to reduce variance and improve model stability. In bagging, we generate B different datasets by randomly sampling with replacement from the original training data. For each of these bootstrapped datasets, a separate tree is trained. This means that in total B trees get formed. Random Forest aggregates their predictions by simple averaging. For horizon j and a new feature vector $\mathbf{X}_t$, the ensemble forecast is

$$\hat{y}_{t+h_j} \;=\; \frac{1}{B}\sum_{b=1}^{B} \hat{f}_j^{(b)}(\mathbf{X}_t) = \frac{1}{B}\sum_{b=1}^{B} \left(\frac{1}{|R_{b,j}(t)|} \sum_{i \in R_{b,j}(t)} y_{i+h_j} \right) \tag{72}$$

where $\hat{f}^{(b)}(x)$ is the prediction from the b-th tree trained on its respective bootstrap sample as in (70).

Besides bagging Random Forest introduces an additional layer of randomness. At each split in tree b it randomly selects a subset of size $\lfloor \sqrt{P} \rfloor$ features and finds the best split only among those features. This subsampling prevents dominant predictors being chosen in every tree and prevents that the trees in the forest become too correlated.

Let σ^2 denote the variance of a single full-grown tree and the forest as $T = \sum_{b=1}^{B} T_b$. Let ρ be the average pairwise correlation between trees at the same feature vector $\mathbf{X}_t$. The variance of the forest then writes

$$\mathrm{Var}(T) = \frac{1}{B^2}\sum_{i=1}^{B}\sum_{j=1}^{B} \mathrm{Cov}\big(T_i, T_j\big) = \rho\,\sigma^2 + \frac{1-\rho}{B}\,\sigma^2. \tag{73}$$

When the number of bootstrapped datasets $B \to \infty$ the second term will tend to 0. Furthermore, when trees are grown on different feature subsets, we will have $\rho < 1$ such that the ensemble variance of the entire forest is lower than σ^2 of a single tree, yielding more stable and accurate forecasts without increasing bias.

7.5 Extreme Gradient Boosting

Like Random Forest, Extreme Gradient Boosting (XGBoost), developed by Chen and Guestrin [2016], builds an ensemble of trees. Instead of training them independently on bootstrapped samples, it adds them sequentially to correct the squared residuals of the existing model

$$\ell\big(y_{i+h_j},\, \hat{y}^{(b-1)}_{i+h_j}\big) = \left(y_{i+h_j} - \left(\hat{y}^{(b-1)}_{i+h_j} + \nu\, f^{(b)}_j(\mathbf{X}_i)\right)\right)^2. \tag{74}$$

Each tree $f^{(b)}_j$ partitions $\mathbb{R}^P$ into M_{leaf} leafs $\{R_{b,j,m}\}_{m=1}^{T}$ and assigns to each leaf m a weight w_m. Furthermore, the hyperparameter ν denotes the learning rate. It controls the shrinkage of each new tree's contribution to the ensemble. This means that for small values of ν each tree makes a small correction, such that the model needs more trees B. XGBoost minimizes a regularized objective

$$\hat{f}^{(b)}_j = \arg\min_f \sum_{i=1}^{N} \ell\big(y_{i+h_j},\, \hat{y}^{(b-1)}_{i+h_j} + \nu\, f(\mathbf{X}_i)\big) + \Omega(f), \tag{75}$$

where $\Omega(f)$ is a L_2 regularization term

$$\Omega(f) = \gamma\, M_{leafs} + \tfrac{1}{2}\lambda \sum_{m=1}^{M_{leaf}} w_m^2. \tag{76}$$

The hyperparameter γ controls the complexity of the model by imposing a penalty on the number of leafs in the tree. The hyperparameters $\boldsymbol{\theta}^{(hp)}$ therefore include $\{\nu,\, \gamma,\, \lambda, B\}$. Because the minimization nproblem in (75) has no closed form solution, XGBoost uses a second-order Taylor approximation around the prediction of the previous $b-1$ trees

$$Q^{(b)}(f) = \sum_{i=1}^{N} \big[g_i\, f(\mathbf{X}_i) \;+\; \tfrac{1}{2}\, h_i\, f(\mathbf{X}_i)^2\big] \;+\; \Omega(f), \tag{77}$$

where g_i and h_i are the first and second derivatives of ℓ in (74) at $\hat{y}^{(b-1)}_{i+h_j}$. Setting $\partial Q/\partial w_m = 0$ yields the optimal leaf weights

$$w^*_m = -\frac{\sum_{i\in R_{b,j,m}} g_i}{\sum_{i\in R_{b,j,m}} h_i + \lambda}\,, \tag{78}$$

and hence to the fitted tree

$$\hat{f}^{(b)}_j(\mathbf{X}_t) = \sum_{m=1}^{T} w^*_{b,j,m}\, \mathbf{1}\{\mathbf{X}_t \in R_{b,j,m}\}. \tag{79}$$

The forecast is simply the weight $w*$ of the leaf that $\boldsymbol{X}_t$ falls into. Next, we update the ensemble prediction by a shrinkage step

$$\hat{y}^{(b)}_{t+h_j} = \hat{y}^{(b-1)}_{t+h_j} + \nu\, \hat{f}^{(b)}_j(\mathbf{X}_t). \tag{80}$$

Thus after B iterations the forecast writes

$$\hat{y}_{t+h_j}^{\text{XGB}} = \hat{y}_{t+h_j}^{(0)} + \sum_{b=1}^{B} \nu \, \hat{f}_j^{(b)}(\mathbf{X}_t) \tag{81}$$

where we initialize $\hat{y}_{t+h_j}^{(0)} = \frac{1}{N}\sum_{i=1}^{N} y_{i+h_j}$.

In essence, XGBoost builds an additive model by fitting each new tree to the residuals of the current ensemble and then shrinking its contribution. XGBoost's ability to iteratively improve model performance by focusing on errors from previous iterations may make it well-suited for forecasting high volatility periods because it potentially captures non-linear dynamics of the realized variance more effectively, addressing our key goal of enhancing prediction accuracy.

7.6 Support Vector Regression

Support Vector Regression (SVR) adapts the Support Vector Machine framework proposed by Vapnik [1995], to a regression setting. We incorporate it in our study, because before the widespread adoption of deep learning algorithms, SVR was a very popular method for predicting financial metrics [Gupta et al., 2019]. It gained popularity due to its robustness in handling non-stationary and noisy data.

This robustness comes from the fact that SVR fits a function that remains within an ε-insensitive "tube" around the training targets. SVR estimates a linear regression function $f_j(\mathbf{X}_i)$ that approximates the targets y_{i+h_j} up to a tolerance ε. In practice, this means that any prediction error $|y_{i+h_j} - \hat{f}_j(\mathbf{X}_i)| \leq \varepsilon$ is treated as zero loss. The advantage of this approach is that the model does not try to fit noise or small fluctuations. Geometrically, one can view this as fitting a "tube" of radius ε around the regression function. Points $\boldsymbol{X}_i$ for which $|y_{i+h_j} - \hat{f}_j(\mathbf{X}_i)| > \varepsilon$ lie outside this tube and receive a penalty via slack variables.

For each horizon j, SVR solves

$$\min_{\boldsymbol{w}_j, b_j, \{\xi_i, \xi_i^*\}} \frac{1}{2}\|\boldsymbol{w}_j\|_2 \; + \; C\sum_{i=1}^{N}(\xi_i + \xi_i^*), \tag{82}$$

under the constraints

$$y_{i+h_j} - (w_j^\top \mathbf{X}_i + b_j) \leq \varepsilon + \xi_i, \tag{83a}$$

$$(\boldsymbol{w}_j^\top \mathbf{X}_i + b_j) - y_{i+h_j} \leq \varepsilon + \xi_i^*, \tag{83b}$$

$$\xi_i, \; \xi_i^* \geq 0, \tag{83c}$$

for $i = 1, \ldots, N$ where $w_j \in \mathbb{R}^P$ and $b_j \in \mathbb{R}$ define the linear prediction $f_j(\mathbf{X}) = w_j^\top \mathbf{X} + b_j$, ε sets the width of the "ε-tube," and slack variables ξ_i, ξ_i^* penalize deviations beyond ε. The regularization constant $C > 0$ is inversely proportional to the L_2 regularization strength trading off tube width against penalty for errors.

Using Lagrange multipliers α_i, α_i^* and applying the kernel trick $K(\mathbf{X}, \mathbf{X}') = \langle \zeta(\mathbf{X}), \zeta(\mathbf{X}') \rangle$ yields the dual form and the predictor

$$\hat{f}_j(\mathbf{X}_t) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*)\, K(\mathbf{X}_t, \mathbf{X}_i) \; + \; b_j. \tag{84}$$

Observations i with α_i or $\alpha_i^* > 0$ are the support vectors of the ε tube. These points are the only observations that contribute to the predictor $\hat{f}_j(\mathbf{X}_t)$ in (84), which implies that the complexity of the model depends on the number of support vectors. The kernel trick gives SVR the power to operate as if it were fitting a linear model in a very high or even infinite dimensional feature space $\zeta(\mathbf{X})$ without ever computing the mapping ζ explicitly. We use the radial basis function defined as

$$K\big(\mathbf{X}_t, \mathbf{X}_i\big) = \exp\big(-\gamma \, \|\mathbf{X}_t - \mathbf{X}_i\|^2\big), \tag{85}$$

with $\gamma > 0$ which controls how quickly the similarity between two points decreases with distance. That is, for large values of γ the model focuses tightly around each support vector. A small γ makes points remain similar over larger distances, which makes the model more general [Hastie et al., 2023]. The SVR hyperparameters are $\boldsymbol{\theta}^{(hp)} = \{C, \; \varepsilon, \gamma\}$

One of the strengths of SVR is its strong theoretical foundation. The optimization problem in (83)) is convex, which ensures a unique solution. However, kernel SVR scales as $\mathcal{O}(N^3)$ in training time and $\mathcal{O}(n^2)$ in memory for n support vectors, making it costly for large samples [Hastie et al., 2009]. Unlike LASSO or Elastic Net, SVR does not have feature selection capabilities, and its performance is very sensitive to the scale of the features, as well as on the hyperparameters.

8 Deep Learning

Deep learning refers to a family of models that learn complex non-linear relationships of the input data through multiple layers of linear and nonlinear transformations. In the context of our multi-horizon forecasting framework, the definition of $f_{\boldsymbol{\theta}}$ in section 7 remains unchanged. Now $\boldsymbol{\theta}$ collects both the model parameters such as the weights and biases of each layer and the hyperparameters, such as number of hidden layers and learning rate. Such hyperparameters are set manually without grid searching due to the heavy computational costs of deep learning models.

The shapes of our inputs changes slightly as described in section 7. We collect the inputs into a third-order tensor $\mathbf{X} \in \mathbb{R}^{N \times L \times P}$,where each slice $\mathbf{X}_i \in \mathbb{R}^{L \times P}$ contains the P features over the past L days for sample i. The corresponding multi-horizon targets are $\mathbf{Y} = \big(\mathbf{y}_i\big)_{i=1}^{N} \in \mathbb{R}^{N \times |\mathcal{H}|}$ where $\mathbf{y}_i = \big(y_{i+h_1}, \ldots, y_{i+h_H}\big)'$. The pair $(\boldsymbol{X_i}, \boldsymbol{y_i})$ is used to fit the model. Similar as before let t denote an out-of-sample index such that we compare $\hat{\boldsymbol{y}}$ with $\boldsymbol{y_t}$. A detailed explanation on activation functions, dropout regularization and backpropagation is given in Appendix D.0.1

8.1 Feedforward Network

Feedforward neural networks, or multilayer perceptrons (MLPs), are simple neural networks. The information moves forward to the output nodes with no cycles or loops present [Masini et al., 2023].

We define for each sample i the flattened input vector $\mathbf{z}_i = \mathrm{vec}(\mathbf{X}_i) \in \mathbb{R}^{LP}$, The MLP implements the mapping $f_{\boldsymbol{\theta}}$ with D hidden layers of affine transforms and activations

$$\mathbf{h}_i^{(0)} = \mathbf{z}_i, \tag{86a}$$
$$\mathbf{a}_i^{(d)} = W^{(d)}\,\mathbf{h}_i^{(d-1)} + \mathbf{b}^{(d)}, \quad d = 1, \ldots, D, \tag{86b}$$
$$\mathbf{h}_i^{(d)} = ReLU\big(\mathbf{a}_i^{(d)}\big), \tag{86c}$$

where each $W^{(d)}, \mathbf{b}^{(d)} \in \boldsymbol{\theta}^{(m)}$ are learned weights and biases and $\hat{\mathbf{y}}_i$ is the forecast vector

$$\hat{\mathbf{y}}_i = W^{(D+1)}\,\mathbf{h}_i^{(D)} + \mathbf{b}^{(D+1)}. \tag{87}$$

In our experiments we set $D = 2$ hidden layers with widths $h_1 = 48$ and $h_2 = 24$, and apply dropout with rate $p = 0.2$ after each activation.

8.1.1 Long Short-Term Memory Network

Unlike feedforward networks, which process each input independently, recurrent neural networks (RNNs) have a hidden state $\mathbf{h}_t$ that is updated sequentially and allows information to flow across time steps. This temporal structure makes RNNs very suitable in time series applications. However, simple RNNs suffer from vanishing or exploding gradients when dependencies become too long [Masini et al., 2023].

The Long Short-Term Memory (LSTM) network, first developed by Hochreiter and Schmidhuber [1997], is a type of RNN designed to capture short-run dependencies in $\mathbf{h}_{i,t}$ as well as long-range dependencies in $\mathbf{c}_t$. At time t, an LSTM cell is defined as

$$\mathbf{f}_t = \sigma\big(W_f\,\mathbf{x}_t + U_f\,\mathbf{h}_{t-1} + \mathbf{b}_f\big), \tag{88a}$$
$$\mathbf{i}_t = \sigma\big(W_i\,\mathbf{x}_t + U_I\,\mathbf{h}_{t-1} + \mathbf{b}_i\big), \tag{88b}$$
$$\mathbf{p}_t = \tanh\big(W_c\,\mathbf{x}_t + U_c\,\mathbf{h}_{t-1} + \mathbf{b}_c\big), \tag{88c}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} \;+\; \mathbf{I}_t \odot \mathbf{p}_t, \tag{88d}$$
$$\mathbf{o}_t = \sigma\big(W_o\,\mathbf{x}_t + U_o\,\mathbf{h}_{t-1} + \mathbf{b}_o\big), \tag{88e}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\big(\mathbf{c}_t\big). \tag{88f}$$

The forget gate $\mathbf{f}_t$ controls how much of the previous cell state $\mathbf{c}_{t-1}$ is carried forward. By passing the affine combination of the new input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ through a sigmoid activation, each element of $\mathbf{f}_t \in (0, 1)$ acts as a soft "erase" coefficient on the corresponding component of $\mathbf{c}_{t-1}$. Values near 0 correspond to forgetting irrelevant information, while values near 1 imply information is kept in the model. This is important since it prevents outdated or noisy signals from living in the cell state over many time steps.

The input gate $\mathbf{i}_t$ and the candidate update $\mathbf{p}_t$ work together to add new information to the cell state. The candidate $\mathbf{p}_t \in [-1, 1]$ represents potential new information as an affine combination of the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{i,t-1}$. The input gate $\mathbf{I}_t \in (0, 1)$ decides how much of this new information should be added to the cell state

$\mathbf{c}_{i,t}$. This is done by multiplying $\mathbf{I}_t$ element-wise with $\mathbf{p}_t$ such that only the most relevant new features enter the cell. The new cell state $\mathbf{c}_t$ is then the sum of what should not be forgotten ($\mathbf{f}_t \odot \mathbf{c}_{t-1}$) and what should not be left out ($\mathbf{I}_t \odot \mathbf{p}_t$).

After the cell state $\mathbf{c}_t$ has been updated, the output gate $\mathbf{o}_t$ regulates what part of this internal long-term memory is exposed to the new hidden state $\mathbf{h}_t$, which represents the short-term memory. The output gate does this by combining the input $\mathbf{x}_t$ and previous hidden state $\mathbf{h}_{t-1}$ through the sigmoid function such that $\mathbf{o}_t \in (0, 1)$.

The final hidden state $\mathbf{h}_t$ provides a filtered, nonlinear transformation of the long-run memory of the network [Masini et al., 2023]. A detailed description of the LSTM's backward pass is given in Appendix D.0.1.

8.1.2 Gated Recurrent Unit

Gated Recurrent Units (GRUs),first conceived by [Cho et al., 2014], simplify the LSTM architecture by combining the forget and input gates into a single update gate and merging cell and hidden states. Like LSTMs, GRUs address the vanishing-gradient problem in more standard RNNs while still being able to capture long-range dependencies

$$\mathbf{z}_t = \sigma\big(W_z\,\mathbf{x}_t + U_z\,\mathbf{h}_{t-1} + \mathbf{b}_z\big), \tag{89a}$$
$$\mathbf{r}_t = \sigma\big(W_r\,\mathbf{x}_t + U_r\,\mathbf{h}_{t-1} + \mathbf{b}_r\big), \tag{89b}$$
$$\tilde{\mathbf{h}}_t = \tanh\big(W_h\,\mathbf{x}_t + U_h\,(\mathbf{r}_{i,t} \odot \mathbf{h}_{t-1}) + \mathbf{b}_h\big), \tag{89c}$$
$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} \;+\; \mathbf{z}_t \odot \tilde{\mathbf{h}}_t. \tag{89d}$$

Here, the update gate $\mathbf{z}_t$ blends the previous hidden state $\mathbf{h}_{t-1}$ with the candidate $\tilde{\mathbf{h}}_t$, determining how much new information to incorporate. Furthermore, the reset gate $\mathbf{r}_t$ controls how much of $\mathbf{h}_{t-1}$ to use when computing $\tilde{\mathbf{h}}_t$. By merging the roles of LSTM's forget and input gates, GRUs achieve comparable performance with fewer parameters and slightly faster training [Masini et al., 2023].

8.1.3 Encoder–Decoder Architecture

The encoder–decoder architecture of Luong et al. [2015] splits the process of mapping the input sequence $\boldsymbol{X}_i$ to a prediction $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ into two stages. The first stage is the encoder, which compresses $\boldsymbol{X}_i$ into the hidden-state sequence $\big(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_L\big)$ of a LSTM or GRU with hidden size H. This is done to better capture dependencies present in the input.

The decoder generates the output $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ by transforming the encoder's hidden state representations of the input. For example, the decoder might only use the final encoder state $\mathbf{h}_L$ as its starting point. The attention block, developed by Bahdanau et al. [2015] and Vaswani et al. [2017] allows the decoder to focus on different parts of the output of the encoder. More specifically, rather than compressing $\big(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_L\big)$ into a single vector, the decoder computes a score at each time step which measures its relevance compared to the current prediction. This score writes

$$e_t = \mathbf{v}' \tanh\big(W\,\mathbf{h}_t \;+\; U\,\mathbf{h}_L\big), \tag{90}$$

where $W, U \in \mathbb{R}^{H \times H}$ and $\mathbf{v} \in \mathbb{R}^H$ are learnable parameters. The last hidden state $\mathbf{h}_{i,L}$ is the so called query vector. It is the vector the decoder uses to ask which other hidden states $\mathbf{h}_{i,t}$ are relevant to it. We can convert these scores to a probability distribution by applying a softmax transformation

$$\alpha_t = \frac{\exp(e_t)}{\sum_{s=1}^{L} \exp(e_s)}, \quad \text{such that,} \quad \sum_{t=1}^{L} \alpha_t = 1. \tag{91}$$

Notice that α_t enables us to identify the most relevant hidden states in the encoders output $\big(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_L\big)$, which means we can define the context vector as

$$\mathbf{c}^{\text{attn}} = \sum_{t=1}^{L} \alpha_t \, \mathbf{h}_t, \tag{92}$$

where $\mathbf{c}^{\text{attn}} \in \mathbb{R}^H$. Because of the capability of networks as LSTM's and GRU's to capture long term dependencies, we can think of $\mathbf{h}_{i,L}$ as a global summary of the entire sequence and $\mathbf{c}^{\text{attn}}$ focused, dynamic glimpse at the parts of the past most relevant to it. Concatenating both vectors

$$\mathbf{d}_i^{(0)} = \big[\mathbf{c}^{\text{attn}}; \, \mathbf{h}_L\big] \in \mathbb{R}^{2H} \tag{93}$$

yields the input for the decoder in order to make a prediction for a single sequence i

$$\mathbf{d}_i = \text{RNN}_{\text{dec}}\big(\mathbf{d}_i^{(0)}\big), \qquad \hat{\boldsymbol{y}}_i = W'_{\text{out}} \mathbf{d}_i + \boldsymbol{b}_{\text{out}}, \tag{94}$$

where RNN can be the LSTM or GRU. The gradient descent updates flow not only through the LSTM or GRU layers, but also back through the attention scores. This means the parameters W, U, and $\mathbf{v}$ in the attention score are part of $\boldsymbol{\theta}$. Thus the model learns where to place its attention by adjusting W, U, and $\mathbf{v}$.

8.2 Transformer

The Transformer architecture of Vaswani et al. [2017], which revolutionized generative AI by replacing recurrence with self-attention, has shown remarkable success in language and vision [Stryker and Bergmann, 2025]. Here we investigate its performance for multi-horizon volatility forecasting.

Since the Transformer has no built-in notion of time, we add sinusoidal positional encodings to the input features so the model can distinguish different time steps [Vaswani et al., 2017]. We first project $\boldsymbol{x}_t$ to the model dimension, or hidden size d:

$$\tilde{\boldsymbol{x}}_t = W_{\text{in}} \, \boldsymbol{x}_t + \boldsymbol{b}_{\text{in}}, \tag{95}$$

such that $\tilde{\boldsymbol{x}}_t \in \mathbb{R}^d$. This means that each time step is represented by a vector of dimension d. We then add a deterministic positional vector $\boldsymbol{p}_t \in \mathbb{R}^d$ defined by

$$p_{t,2k} = \sin\!\left(\frac{t}{10000^{2k/d}}\right), \quad p_{t,2k+1} = \cos\!\left(\frac{t}{10000^{2k/d}}\right), \tag{96}$$

for $k = 0, \ldots, \frac{d}{2} - 1$. By giving each dimension its own frequency, the model gets many overlapping wave patterns. At any time the exact combination of all these sines and cosines is unique, so the network can infer the position t from that pattern [Vaswani et al., 2017].

At the bottom of the encoder stack, we start with the embedded inputs plus positional encoding as our initial activations. Each encoder layer ℓ takes the activations

$$H^{(\ell-1)} = \begin{pmatrix} \boldsymbol{h}_1^{(\ell-1)\top} \\ \vdots \\ \boldsymbol{h}_L^{(\ell-1)\top} \end{pmatrix} \in \mathbb{R}^{L \times d} \tag{97}$$

from the layer below and transforms it to new activations $H^{(\ell)}$. In other words, at each encoder layer we begin with a matrix of hidden activations where each row $\boldsymbol{h}_t^{(\ell-1)}$ summarizes all information up to time step t. To enable every position t to query all other positions s for information, we apply three learned linear projections (as in Vaswani et al. [2017])

$$Q = H^{(\ell-1)} W^Q, \quad K = H^{(\ell-1)} W^K, \quad V = H^{(\ell-1)} W^V. \tag{98}$$

Here, $\boldsymbol{q}_t$ is the query vector that represents what time step t wants to know from the rest of the sequence, $\boldsymbol{k}_s$ is the key vector that describes what information time step s offers, and $\boldsymbol{v}_s$ is the value vector holding the actual content available at s. We compute the dot product between $\boldsymbol{q}_t$ and $\boldsymbol{k}_s$ to measure how well the question from t matches the information at s. Those matching scores are then normalized and used to put weight on $\boldsymbol{v}_s$, producing a weighted sum that gives the information time step t needs from across the whole sequence.

We split the feature dimension d of Q, K, V into h heads each of size $d_k = d/h$. For example, we partition Q as

$$Q = \big[\, Q_1 \,\big|\, Q_2 \,\big|\, \ldots \,\big|\, Q_h \,\big], \quad Q_j \in^{L \times d_k} . \tag{99}$$

Each head learns to focus on a different type of temporal relationship [Michel et al., 2019]. For instance, one head might capture short-term autocorrelation and another long-term seasonality. For each head j we perform the following transformation:

$$\text{head}_j = \text{softmax}\Big(\frac{Q_j K_j^\top}{\sqrt{d_k}}\Big) V_j \;\in^{L \times d_k}. \tag{100}$$

Here $Q_j K_j^\top$ yields an $L \times L$ matrix of similarity scores between every query position and every key position. A larger dot product implies that the two vectors point in more similar directions, which means the model interprets position s to be relevant to the question posed by t. If the dimensionality of each head d_k is very large, the dot products can grow and cause the softmax gradients to vanish. To prevent this, we follow Vaswani et al. [2017] and rescale by dividing by $\sqrt{d_k}$. The softmax across each row converts the similarity scores into attention weights that sum to 1, ensuring that each output at position t is a convex combination of the value vectors $\boldsymbol{v}_s$.

Then, we concatenate all the heads along the feature dimension d, resulting in an $L \times d$ matrix, and apply a final linear projection:

$$\text{MHA}(H^{(\ell-1)}) = \big[\, \text{head}_1 \,\big|\, \text{head}_2 \,\big|\, \ldots \,\big|\, \text{head}_h \,\big] W^O, \tag{101}$$

where $W^O \in \mathbb{R}^{d\times d}$ is a learned projection matrix, so that the multi-head attention block produces an output of shape $L \times d$ as in Vaswani et al. [2017].
After multi-head attention we apply layer normalization [Ba et al., 2016]. Layer normalization normalizes the activations across the feature dimension for each individual sequence position. Given an input vector $\boldsymbol{z} \in \mathbb{R}^d$, layer normalization computes

$$\mu = \frac{1}{d}\sum_{j=1}^{d} z_j, \quad \sigma^2 = \frac{1}{d}\sum_{j=1}^{d}(z_j - \mu)^2, \tag{102a}$$

$$\mathrm{LayerNorm}(\boldsymbol{z}) = \boldsymbol{\gamma} \odot \frac{\boldsymbol{z} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \boldsymbol{\beta}, \tag{102b}$$

where $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^d$ are learned scale and shift parameters, and ϵ is a small constant for numerical stability. After the multi-head-attention block, we perform

$$Z^{(\ell)} = \mathrm{LayerNorm}\Big(H^{(\ell-1)} + \mathrm{Dropout}\big(\mathrm{MHA}(H^{(\ell-1)})\big)\Big), \tag{103}$$

so that $Z^{(\ell)}$ is an $L \times d$ matrix whose rows have mean 0 and unit variance. This residual connection plus normalization helps keep the activations stable across layers [Ba et al., 2016].
A feed-forward network then transforms each time step independently

$$F^{(\ell)}[t] = W_2\big(\mathrm{ReLU}(W_1\, Z^{(\ell)}[t] + b_1)\big) + b_2, \tag{104}$$

where $W_1 \in^{d\times d_{\mathrm{ff}}}$ and $W_2 \in^{d_{\mathrm{ff}}\times d}$ define a two-layer MLP. This provides nonlinearity complementing the linear attention mechanism. As before, a residual connection and layer normalization follow

$$H^{(\ell)} = \mathrm{LayerNorm}\big(Z^{(\ell)} + \mathrm{Dropout}(F^{(\ell)})\big), \tag{105}$$

yielding the output of the encoder layer.
Each decoder layer mirrors the encoder but with two attention sublayers. In the decoder's first attention sublayer, we employ masked self-attention to prevent information flow from future positions to the current prediction. Concretely, given decoder inputs $D^{(\ell-1)}$, we compute Q, K, V as in (98) with $D^{(\ell-1)}$ replacing $H^{(\ell-1)}$. We apply a causal mask $M \in \{0,1\}^{L\times L}$ with

$$M_{t,s} = \mathbf{1}\{s \le t\}, \tag{106}$$

such that entries corresponding to future positions ($s > t$) are blocked. The masked scores become

$$\mathrm{MaskedScores}_{t,s} = \begin{cases} \frac{QK^\top}{\sqrt{d_k}}, & M_{t,s} = 1, \\ -\infty, & M_{t,s} = 0, \end{cases} \tag{107}$$

and after the softmax we obtain $A = \mathrm{softmax}(\mathrm{MaskedScores})$, which places zero weight on all future positions. The decoder's masked self-attention output is then

$$\mathrm{MHA}_{\mathrm{masked}}(D^{(\ell-1)}) = A\,V, \tag{108}$$

ensuring information does not flow from future positions to the current prediction [Ghojogh and Ghodsi, 2020].

After the decoder's masked self-attention and its normalization layer, the decoder performs cross-attention to incorporate information from the encoder outputs. Let $D^{(\ell)}$ be the decoder activations after these sublayers at layer ℓ, and $H^{(N)}$ be the final encoder output. We compute

$$Q = D^{(\ell)} W^{Q}_{\text{cross}}, \quad K = H^{(N)} W^{K}_{\text{cross}}, \quad V = H^{(N)} W^{V}_{\text{cross}}. \tag{109}$$

Splitting into h heads of size d_k and applying scaled dot-product yields the cross-attention output

$$\text{CrossAttn}(D^{(\ell)}, H^{(N)}) = \left[\, \text{head}_1 \,\middle|\, \text{head}_2 \,\middle|\, \ldots \,\middle|\, \text{head}_h \,\right] W^{O}_{\text{cross}}. \tag{110}$$

Since our goal is to predict h future values, we adapt the specification of the decoder as specified in Vaswani et al. [2017] with an auto-regressive compenent, which follows a similar approach proposed by NLP [2018]. We start the decoder with an initial value of 0 in the same dimension of the model d

$$D^{(0)} \;=\; \left[\mathbf{0}^{\top}\right] \;\in\; \mathbb{R}^{1\times d}, \tag{111}$$

At each horizon $h = 1, 2, \ldots, |\mathcal{H}|$, we already have $h1$ forecasts embedded in d dimensions, $D^{h-1} \in \mathbb{R}^{(h-1)\times d}$. Before we are ready to perform the attention mechanism, we need to add positional encoding since the model needs to know which forecast is which in the sequence. That is, we define

$$\tilde{D}^{(h-1)} \;=\; D^{(h-1)} \;+\; \begin{pmatrix} \boldsymbol{p}_{t+1}^{\top} \\ \vdots \\ \boldsymbol{p}_{t+h-1}^{\top} \end{pmatrix}, \tag{112}$$

where each $\boldsymbol{p}_{t+\tau} \in \mathbb{R}^d$ is defined exactly the same as in 96. We then pass $\tilde{D}^{(i-1)}$ through the N decoder layers, where masked self attention is important to prevent previous information affect the current forecast. After the final layer we obtain the embedding $D^{(N)} \in \mathbb{R}^{h\times d}$, from which we extract the last row $D^{(N)}_{[-1]}$ and compute a scalar forecast via

$$\hat{y}_{t+h} = \boldsymbol{w}_{\text{out}}^{\top}\, D^{(N)}_{[-1]} + b_{\text{out}} \;\in\; \mathbb{R}. \tag{113}$$

which we re-embed in d dimensions following

$$\boldsymbol{e}_{t+h} = W_{\text{emb}}\, \hat{y}_{t+i} + \boldsymbol{b}_{\text{emb}} \;\in\; \mathbb{R}^d, \tag{114}$$

and concatenated to the decoder inputs to form

$$D^{(h)} = \left[D^{(h-1)};\, \boldsymbol{e}_{t+h}^{\top}\right] \;\in\; \mathbb{R}^{h\times d}. \tag{115}$$

This auto-regressive adaptation allows us to fully utilize the Transformer's powerful attention mechanisms combining with the ability to model temporal dependencies.

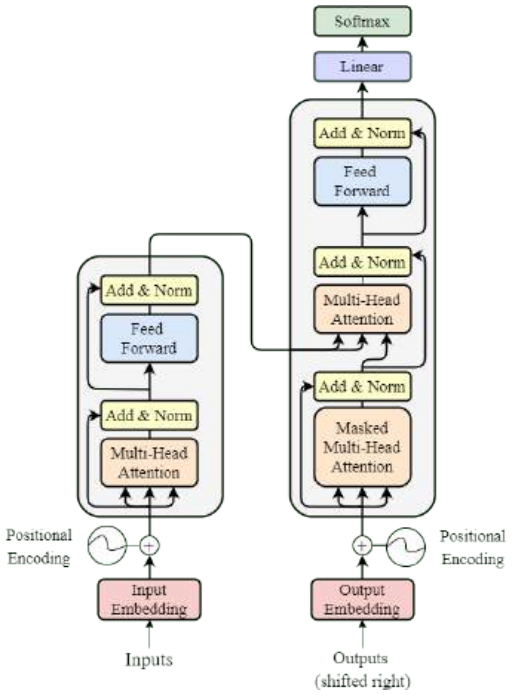


Figure 2: A graphical representation of the vanilla Transformer model of Vaswani et al. [2017].

9 Evaluation Methods

We evaluate our volatility forecasting models using both in-sample and out-of-sample criteria. In-sample metrics assess how well each model fits the training data, while out-of-sample tests measure predictive accuracy on held-out data. Together, these criteria ensure that our models are both well-specified and genuinely capable of delivering reliable forecasts

9.1 Rolling Window Cross-Validation

In our rolling-window scheme, we repeatedly fit each model on a fixed-length block of the most recent observations and then make a h-step-ahead forecast. We begin by estimating the model's parameters on the sample of observations ranging from 2018-01-01 until 2023-01-1. This set containing N observations is our first window and is the set for which hyperparameters get optimized. After making a forecast $h \in \mathcal{H}$ steps away we slide the window forward one day where we simultaneously drop the first observations in the sample such that the number of observations in the train set remains N. Then we re estimate the model parameters on the new training set and again make a forecast h steps away from the training set. We keep doing this until the complete dataset is exhausted. We only move the window one step each time because the number of training samples is approximately inversely proportional to the step size. That is, moving the window 10 steps each time would yield 10 times less training points [Li et al., 2022].

After the first iteration, we keep the hyperparameters fixed to reduce the computational

burden of the procedure. This rolling window approach is chosen over an expanding window where we do not remove observations at the beginning of the dataset such that the training set keeps growing. A rolling window protects us against possible structural breaks in the data, where old observations influences the parameter estimates too much, hurting forecast performance on more recent observations. We do not leave a gap between the train and test set. Figure 3 illustrates the procedure for $h = 1$

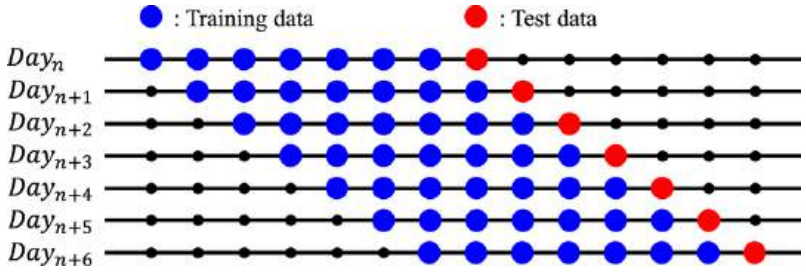


Figure 3: Illustration of the Rolling Window procedure from Park et al. [2021]

The likelihoods are minimized using Sequential Least Squares Programming (SLSQP), which is an iterative method for constrained nonlinear optimization. A more detailed explanation is given in Appendix A. For non-machine learning models we maximize the likelihood of each model at each iteration in the rolling window. However, even small changes in volatility and outliers can cause a different location or number of local minima and possibly a more flat surface of the likelihood. This might mean that at some iteration convergence might fail but succeeds only a few steps further [Kraft, 1988]. The line search might start in a region with better curvature or some constraints might be inactive, allowing easier optimization. If convergence fails, we simply use the parameters trained on the previous training set in the rolling window and try again. Furthermore, if the optimization fails on the first window, we use our initial guess. If the optimization procedure fails more than 10 times in a row we disregard the model. In the case of deep learning models, we perform 200 epochs (one forward and backward pass) on the initial training set. On subsequent rolling windows we perform three additional epochs to tune the weights on the new training set.

9.2 In-Sample Evaluation

In our in-sample analysis, we assess how well each model captures the observed volatility dynamics and how reliably its parameters are estimated. A standard tool for this purpose is the total log-likelihood, $\log L$, which measures the probability of the data given the fitted model. However, models with more parameters can always increase $\log L$ by overfitting, so we need measures that penalizes complex models. All measures are calculated on the first window only.

The Akaike Information Criterion (AIC) of Akaike [1973] is defined as

$$\text{AIC} = -2 \log \hat{L} \; + \; 2\,k, \tag{116}$$

where k is the total number of parameters estimated. The AIC can be interpreted as an approximately unbiased estimate of the Kullback–Leibler divergence between the fitted

model and the true data-generating process. Thus, lower AIC values indicate a better balance between goodness-of-fit and sparsity.

The Bayesian Information Criterion (BIC), introduced by Schwarz [1978], imposes a more strict penalty on model size when the sample is large

$$\mathrm{BIC} = -2 \log \hat{L} \; + \; k \log N. \tag{117}$$

For bigger N, $\log N$ also slowly grows such that extra parameters gets punished increasingly with sample size. This favors simpler models, making BIC useful for selecting the true model.

In practice, we compute both AIC and BIC for each of the competing models. The AIC tends to select models that better mimic out-of-sample predictive accuracy in small-to-moderate samples, while BIC more aggressively avoids overfitting when the sample is large. By comparing both criteria, we gain insight into the trade-off between fit and complexity.

9.3 Out-of-Sample Evaluation

In this section we outline a comprehensive evaluation framework designed to ensure that our conclusions regarding volatility forecasts are both robust and reliable.

9.3.1 Forecasting Error Metrics

Let $m = 1, \ldots, M$ index our competing models and $j = 1, \ldots, n$ index the stocks in our panel. For each model–stock pair (m, j), we write $\hat{h}^{(m)}_{j,t,h}$ for the h-step-ahead forecast of stock j at last training time t, using parameters $\hat{\boldsymbol{\theta}}^{(m,j)}$ estimated on the training window $\{t - N + 1, \ldots, t\}$. Denote by $RK_{j,t}$ the realized-kernel proxy for stock j on day t, and define the scaling factor for stock j by

$$\mathrm{factor}_j \;=\; \frac{\frac{1}{N}\sum_{s=1}^{N}(r_{j,s} - \bar{r}_j)^2}{\frac{1}{N}\sum_{s=1}^{N} RK_{j,s}} \tag{118}$$

so that the adjusted forecast error is

$$e^{(m)}_{j,t,h}(\hat{\boldsymbol{\theta}}^{(m,j)}) \;=\; \hat{h}^{(m)}_{j,t,h} \;-\; \big(x_{j,t} \times \mathrm{factor}_j\big). \tag{119}$$

where $e^{(m)}_{j,t,h}$ depends on estimated model parameters $\hat{\boldsymbol{\theta}}^{(m,j)}$ through $\hat{h}^{(m)}_{j,t}$. The reason for this correction factor is that the close-to-close return r_t corresponds with a period of 24 hours and the realized kernel corresponds with a period of one trading day (6.5 hours). Hence, o average the estimated conditional variance of the close-to-close returns will be at a higher level than the realized kernel.
We then compute, for each (m, j) and h,

$$\widehat{\mathrm{FMSE}}^{(m)}_{j,h}(\hat{\boldsymbol{\theta}}^{(m,j)}) = \frac{1}{T - N - h + 1}\sum_{t=N}^{T-h}\big(e^{(m)}_{j,t}\big)^2, \tag{120a}$$

$$\widehat{\mathrm{FMAE}}^{(m)}_{j,h}(\hat{\boldsymbol{\theta}}^{(m,j)}) = \frac{1}{T - N - h + 1}\sum_{t=N}^{T-h}\big|e^{(m)}_{j,t}\big|, \tag{120b}$$

where

$$\widehat{\mathrm{RFMSE}}_{j,h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)}) = \sqrt{\widehat{\mathrm{FMSE}}_{j,h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)}}. \tag{121}$$

To summarize performance across all n stocks, we report the cross-sectional averages

$$\overline{\mathrm{RFMSE}}_{h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)}) = \frac{1}{n}\sum_{j=1}^{n} \widehat{\mathrm{RFMSE}}_{j,h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)}), \tag{122a}$$

$$\overline{\mathrm{FMAE}}_{h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)}) = \frac{1}{n}\sum_{j=1}^{n} \widehat{\mathrm{FMAE}}_{j,h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)}). \tag{122b}$$

$\overline{\mathrm{RFMSE}}_{h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)})$ squares each error before averaging and then takes a root. This makes it particularly sensitive to occasional large forecast failures. This could also be seen as an advantage when one wishes to penalize extreme misses. In volatility forecasting, where large errors can signal missed regime shifts or crises, $\overline{\mathrm{RFMSE}}_{h}^{(m)}(\hat{\boldsymbol{\theta}}^{(\boldsymbol{m,j})})$ highlights a model's tail-risk performance. $\overline{\mathrm{FMAE}}_{h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)})$ treats all deviations linearly and thus is more robust to isolated large errors. It provides a more balanced view of typical error magnitudes, at the cost of under-penalizing very large misses.

Under standard regularity conditions such as stationarity and ergodicity,

$$\widehat{\mathrm{RFMSE}}_{j,h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)}) \xrightarrow{p} \mathrm{RFMSE}_{j,h}^{(m)}(\boldsymbol{\theta}_0^{(m,j)}) = E\Big[\big(e_{j,t,h}^{(m)}((\boldsymbol{\theta}_0^{m,j}))\big)^2\Big], \tag{123a}$$

$$\widehat{\mathrm{FMAE}}_{j,h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)}) \xrightarrow{p} \mathrm{FMAE}_{j,h}^{(m)}(\boldsymbol{\theta}_0^{(m,j)}) = E\big|e_{j,t,h}^{(m)}(\boldsymbol{\theta}_0^{(m,j)})\big|, \tag{123b}$$

where $\boldsymbol{\theta}_0^{(m,j)}$ denotes the true data-generating parameters. Since $\widehat{\mathrm{RFMSE}}_{j,h}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)})$ and $\widehat{\mathrm{FMAE}}_{j}^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)})$ are random variables, which only converge to their true values for $T \to \infty$ it is impossible to conclude model m_1 performs better than model m_2 if $\widehat{\mathrm{RFMSE}}_{j}^{(m_1)}(\hat{\boldsymbol{\theta}}^{(m_1,j)}) < \widehat{\mathrm{RFMSE}}_{j}^{(m_2)}(\hat{\boldsymbol{\theta}}^{(m_2,j)})$.

9.4 Diebold–Mariano Test

The Diebold–Mariano (DM) test of Diebold and Mariano [2002] compares predictive accuracy between two forecasting models $m_1, m_2 \in \{1, \ldots, M\}$, $m_1 \neq m_2$. For a fixed stock j and horizon h, we define the squared-loss differential at test date t_k as

$$d_{j,t_k,h}^{(m_1,m_2)} \;=\; \big(e_{j,t_k,h}^{(m_1)}\big)^2 \;-\; \big(e_{j,t_k,h}^{(m_2)}\big)^2, \tag{124}$$

where $k = 1, \ldots, K$ and $K = T - N - h + 1$. The sample mean is given by

$$\bar{d}_{j,h} \;=\; \frac{1}{K}\sum_{k=1}^{K} d_{j,t_k,h}^{(m_1,m_2)}. \tag{125}$$

Under the null hypothesis

$$H_0 : \; \mathbb{E}\big[d_{j,t_k,h}^{(m_1,m_2)}\big] = 0,$$

the DM statistic for stock j and horizon h writes

$$\mathrm{DM}_{j,h}^{(m_1,m_2)} \;=\; \frac{\bar{d}_{j,h}}{\sqrt{\widehat{\omega}_{j,h}/K}} \;\xrightarrow{d}\; \mathcal{N}(0,1), \tag{126}$$

where the long-run variance $\widehat{\omega}_{j,h}$ is estimated by the Newey–West estimator

$$\widehat{\omega}_{j,h} \;=\; \widehat{\gamma}_{j,h}(0) \;+\; 2\sum_{\ell=1}^{L}\Big(1-\tfrac{\ell}{L+1}\Big)\,\widehat{\gamma}_{j,h}(\ell), \tag{127}$$

with

$$\widehat{\gamma}_{j,h}(\ell) \;=\; \frac{1}{K}\sum_{k=\ell+1}^{K}\big(d_{j,t_k,h}-\bar{d}_{j,h}\big)\big(d_{j,t_{k-\ell},h}-\bar{d}_{j,h}\big), \qquad \ell=0,1,\dots,L. \tag{128}$$

Adopting the Newey–West estimator with a data-driven lag $L = \lfloor K^{1/3} \rfloor$ ensures valid inference even when $\{d_{j,t_k,h}\}$ exhibits autocorrelation or mild heteroskedasticity. Timmermann and Zhu [2019] extend the classic DM test to panel settings by averaging each stock's loss differential at every test date. Define, for models $m_1 \neq m_2$ and horizon h, the cross-sectional mean differential

$$\bar{d}_{t_k,h}^{(m_1,m_2)} \;=\; \frac{1}{n}\sum_{j=1}^{n}\Big[\big(e_{j,t_k,h}^{(m_1)}\big)^2-\big(e_{j,t_k,h}^{(m_2)}\big)^2\Big], \tag{129}$$

for $k = 1,\dots,K$. Under the null hypothesis

$$H_0: \quad \mathbb{E}\big[\bar{d}_{t_k,h}^{(m_1,m_2)}\big]=0,$$

the panel-DM statistic is

$$\mathrm{DM}_{h}^{(m_1,m_2)} \;=\; \frac{\frac{1}{K}\sum_{k=1}^{K}\bar{d}_{t_k,h}^{(m_1,m_2)}}{\sqrt{\widehat{\omega}_h/K}} \;\xrightarrow{d}\; \mathcal{N}(0,1), \tag{130}$$

where $\widehat{\omega}_h$ is the Newey–West estimator in (127) (without the dependence on stock j) of the long-run variance of $\{\bar{d}_{t_k,h}^{(m_1,m_2)}\}$ with $L = \lfloor K^{1/3} \rfloor$ and

$$\widehat{\gamma}_h(\ell)=\frac{1}{K}\sum_{k=\ell+1}^{K}\big(\bar{d}_{t_k,h}-\overline{\bar{d}}_h\big)\big(\bar{d}_{t_{k-\ell},h}-\overline{\bar{d}}_h\big), \quad \overline{\bar{d}}_h=\frac{1}{K}\sum_{k=1}^{K}\bar{d}_{t_k,h}. \tag{131}$$

This panel version gains power by averaging out idiosyncratic noise across stocks, but its reliance on an asymptotic normal approximation may be delicate if K is small or the loss-differential tail behavior is heavy [Timmermann and Zhu, 2019].

9.5 Model Confidence Set

The Model Confidence Set (MCS) of Hansen et al. [2011a] provides a data-driven way to identify the subset of forecasting models whose predictive ability cannot be distinguished at a given confidence level α. Let $m = 1,\dots,M$ index our candidate models and for each

model m let $\ell_t^{(m)}$ denote its square loss at test date $t = 1, \ldots, K$. We form the $M \times M$ matrix of average loss differences

$$\bar{d}_{m,i} \;=\; \frac{1}{K}\sum_{t=1}^{K}\big[\ell_t^{(m)} - \ell_t^{(i)}\big], \qquad m, i = 1, \ldots, M. \tag{132}$$

To test the null of equal predictive ability within a candidate set $S \subseteq \{1, \ldots, M\}$, we compute the standardized statistics

$$T_{m,i} = \frac{\bar{d}_{m,i}}{\hat{\sigma}_{m,i}/\sqrt{K}}, \tag{133}$$

where $\hat{\sigma}^2_{m,i}$ is a consistent estimate of $(d_t^{(m,i)})$, obtained by a block bootstrap of length b to preserve temporal dependence [Politis and Romano, 1994]. Let

$$T_{\max}(S) = \max_{m,i \in S} |T_{m,i}|. \tag{134}$$

We approximate its sampling distribution by repeatedly resampling blocks of the observed losses and recomputing $T^*_{\max}$. The the empirical $(1-\alpha)$ quantile denoted as $c_{1-\alpha}$ then yields the MCS via the backward elimination algorithm. We begin with the full set of candidate models $S_0 = \{1, \ldots, M\}$. At each iteration r, we compute the maximum pairwise test statistic

$$T_{\max}(S_{r-1}) \;=\; \max_{m,i \in S_{r-1}} \Big| T_{m,i} \Big|. \tag{135}$$

If $T_{\max}(S_{r-1}) \;\leq\; c_{1-\alpha}$, we stop and have found MCS $\;=\; S_{r-1}$..Otherwise, we identify and remove the worst-performing model

$$m^* = \arg \max_{m \in S_{r-1}} \max_{i \in S_{r-1}} \big|T_{m,i}\big|, \tag{136}$$

and set $S_r = S_{r-1} \setminus \{m^*\}$,and repeat the procedure from the second step on this reduced set. We continue until the stopping criterion $T_{\max} \leq c_{1-\alpha}$ is satisfied.

The final MCS contains only those models whose losses are not significantly larger than at least one competitor at level α, thus controlling the family-wise error rate in multiple comparisons.

9.5.1 Borda Scoring of Model Confidence Sets

After computing the Model Confidence Set (MCS) for each stock j and forecast horizon h, we aggregate these binary "included versus excluded" outcomes into a single, horizon-specific ranking via the Borda count discovered by Borda [1781]. For a given horizon h, denote by $\mathcal{S}_j^{(h)} \subseteq \{1, \ldots, M\}$ the set of models retained in the MCS for stock j. Within each stock, every model in $\mathcal{S}_j^{(h)}$ is awarded a base score of 1. Those models excluded from $\mathcal{S}_j^{(h)}$ are then ranked according to the order of their elimination. Models that gets excluded the least, or less quickly than other models get assigned increasing scores $2, 3, \ldots,$

one for each excluded model. We sum these per-stock scores across all n stocks to form the cumulative Borda score for model m at horizon h,

$$B_m^{(h)} = \sum_{j=1}^{n} \text{score}_{j,h}(m), \qquad m = 1, \ldots, M. \tag{137}$$

By construction, a higher aggregate $B_m^{(h)}$ indicates that model m was more consistently retained (or excluded later) across the panel, and hence is deemed superior at that forecast horizon. This voting-based aggregation smooths out idiosyncratic stock-level variability and yields a clear, interpretable ranking of models for each horizon [Goodin, 2005].

9.6 Technique for Order Preference by Similarity to Ideal Solution

In practice, no single metric can fully capture all aspects of forecasting performance. Furthermore, when evaluating dozens of forecasting models across multiple criteria such as $\overline{\text{RFMSE}}_h^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)})$,$\overline{\text{FMAE}}_h^{(m)}(\hat{\boldsymbol{\theta}}^{(m,j)})$, the Diebold–Mariano test, and MCS Borda scores. it is often impossible to declare a single "best" model. Furthermore, the decision which metric is superior in determining forecasts capability is not quantifiable and often comes down to personal preference. The Technique for Order Preference by Similarity to Ideal Solution developed by Hwang and Yoon [1981]) provides an elegant and transparent way to aggregate these metrics into one unified score.

Let p be the number of competing models. For a fixed forecast horizon h we assemble the four performance measures into a $p \times 4$ matrix

$$\mathbf{C} = \begin{pmatrix} c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} \\ \vdots & \vdots & \vdots & \vdots \\ c_{p,1} & c_{p,2} & c_{p,3} & c_{p,4} \end{pmatrix} \tag{138}$$

where columns correspond to (1) average RMSE, (2) average MAE, (3) DM win counts, and (4) MCS Borda scores. Since lower RMSE and MAE indicate better performance, we first invert those two columns by replacing $c_{i,1}$ with $\max_j c_{j,1} - c_{i,1}$, and similarly for MAE.

Next, we normalize each column to unit range,

$$z_{i,k} = \frac{c_{i,k} - \min_j c_{j,k}}{\max_j c_{j,k} - \min_j c_{j,k}} \quad k = 1, \ldots, 4, \tag{139}$$

ensuring all criteria lie in $[0, 1]$ and are directly comparable. Applying a weight vector $\mathbf{w} = (w_1, w_2, w_3, w_4)$ across the four criteria yields the weighted matrix $v_{i,k} = w_k\, z_{i,k}$ such that $\mathbf{V} = [v_{i,k}]_{i=1,\ldots,p;\, k=1,\ldots,4}$. Intuitively, $v_{i,k}$ represents model i's performance on criterion k, scaled by its importance.

This allows us to define the ideal performance vector

$$\mathbf{v}^+ = \big(\max_i v_{i,1},\ \max_i v_{i,2},\ \max_i v_{i,3},\ \max_i v_{i,4}\big) \tag{140}$$

and the worst possible vector

$$\mathbf{v}^- = \big(\min_i v_{i,1},\ \min_i v_{i,2},\ \min_i v_{i,3},\ \min_i v_{i,4}\big). \tag{141}$$

Model i is then scored by the Euclidean distance between the best and worst possible vector

$$d_i^+ \ = \ \left\|\mathbf{v}_i - \mathbf{v}^+\right\|_2, \qquad d_i^- \ = \ \left\|\mathbf{v}_i - \mathbf{v}^-\right\|_2, \tag{142}$$

where $\mathbf{v}_i$ is the ith row of $\mathbf{V}$. A small d_i^+ means model i lies close to the best possible performance on all criteria, while a large d_i^- means it lies far from the worst. Finally, TOPSIS assigns each model the score

$$s_i \ = \ \frac{d_i^-}{d_i^+ + d_i^-}, \tag{143}$$

which ranges from 0 (identical to the worst) to 1 (identical to the best). By construction, models with high s_i simultaneously exhibit low errors, frequent DM victories, and strong inclusion in the MCS, weighted according to $\mathbf{w}$.

9.7 Value-at-Risk

Value-at-Risk (VaR) is a widely used risk-management metric that quantifies the maximum expected loss of a portfolio over a given horizon at a specified confidence level. Let $r_{j,t+1}$ denote the one-day ahead return of stock j, and $\hat{h}_{j,t+1}^{(m)}$ the conditional variance forecast. Again we assume that the returns follow the distribution $\mathcal{D}(\mu, \sigma^2, \boldsymbol{\theta^{(d)}})$ where μ is the mean, variance σ^2 and distribution specific parameter vector $\boldsymbol{\theta^{(d)}}$. The Value at Risk writes

$$\mathrm{VaR}_{j,t+1}^{(m)}(\alpha) \ = \ \hat{\mu}_{j,t+1} \ + \ c_\alpha \sqrt{\hat{h}_{j,t+1}^{(m)}} \tag{144}$$

where $\hat{\mu}_{j,t+1} \ = 0$ is the foretasted mean return and $c_\alpha = \Psi^{-1}(\alpha)$ is the α-quantile of $\mathcal{D}(\mu, \sigma^2, \boldsymbol{\theta^{(d)}})$. For observation and parameter driven models this is feasible since they make assumptions on the distribution of the returns. Machine- and deep learning algorithms in general do not make such assumptions. However, we can compute the VaR empirically. First, over the training window of length N, we standardize each past return by its realized variance

$$\varepsilon_{j,s} \ = \ \frac{r_{j,s}}{\sqrt{RK_{j,s}}}, \qquad s = 1, \ldots, N, \tag{145}$$

where $r_{j,s}$ is the observed return and $RK_{j,s}$ the corresponding realized-kernel estimate. Next, at signifinance level $\alpha = 0.05$, we form the empirical α-quantile of the standardized residuals

$$q_{j,\alpha} \ = \ \inf\big\{q : \ \tfrac{1}{N}\sum_{s=1}^{N} \mathbf{1}\{\varepsilon_{j,s} \le q\} \ge \alpha\big\}. \tag{146}$$

Finally, if the model's one-step-ahead variance forecast at time t is $\hat{h}_{j,t+1}$, the non-parametric VaR is given by

$$\mathrm{VaR}_{j,t+1}^{\mathrm{NP}}(\alpha) \ = \ q_{j,\alpha} \sqrt{\hat{h}_{j,t+1}}. \tag{147}$$

For every model, we can define

$$\text{hit}_{j,t+1}^{(m)}(\alpha) \;=\; \mathbf{1}\{\, r_{j,t+1} < \text{VaR}_{j,t+1}^{(m)}(\alpha)\}, \tag{148}$$

which is an indicator of whether the observed return exceeded the VaR threshold.To assess whether each model's VaR forecasts are statistically reliable, we follow the backtesting framework of Christoffersen [1998]

9.7.1 Unconditional Coverage Test

The unconditional coverage test is used to determine the observed proportion of violations of the VaR threshold, matches the expected frequency. For example, for a 95% VaR, the test checks whether approximately 5% of the time the VaR is violated. Under the null hypothesis violations are correctly predicted at this frequency. The test statistic is defined as

$$LR_{UC} = -2\Big[\log L(\hat{p}) - \log L(p)\Big], \tag{149}$$

where p is the expected exceedance probability and $\hat{p} = x/n'_{VaR}$ is the observed exceedance rate with x being the number of violations. Lastly K is defined as before, as the total number of out-of-sample forecasts. The likelihood under the null writes $L(p) = (p)^x(1-p)^{n_{VaR}-x}$ and $L(\hat{p}) = (\hat{p})^x(1-\hat{p})^{n-x}$ is the likelihood under the alternative. Under the null hypothesis, the test statistic LR_{UC} asymptotically follows a χ^2 distribution with 1 degree of freedom. A significant LR_{UC} value suggests that the violations do not match the expected frequency.

9.7.2 Conditional Coverage Test

The conditional coverage test extends the unconditional coverage test by also examining whether the violations are independent over time.The test statistic writes

$$LR_{CC} = LR_{UC} + LR_{Ind}, \tag{150}$$

where LR_{Ind} is the test statistic for the independence test. Under the null hypothesis, LR_{CC} asymptotically follows a χ^2 distribution with 2 degrees of freedom. A significant result for LR_{CC} indicates either that violations occur at an incorrect frequency, not independent, or both.

9.7.3 Independence Test

The independence test focuses on whether violations are independent over time. This is modeled using a Markov chain framework where the likelihood under independence writes

$$L_{ind} = (p_0)^{n_{00}}(1-p_0)^{n_{01}}(p_1)^{n_{10}}(1-p_1)^{n_{11}}, \tag{151}$$

where n_{ij} represents the number of transitions from state i to state j ($i, j \in \{0, 1\}$) and p_0, p_1 denote the probability of no exceedance and exceedance, respectively. The likelihood under the Markov assumption (independence) is

$$L_{dep} = \prod_{i=1}^{n} \pi_{s_{i-1}s_i}, \tag{152}$$

where $\pi_{s_{i-1}s_i}$ is the transition probability between states. The test statistic is

$$LR_{Ind} = -2\Big[\log L_{ind} - \log L_{dep}\Big], \tag{153}$$

Under the null hypothesis of the Markov assumption, $LR_{Ind} \xrightarrow{p} \chi^2_1$. Rejecting $\boldsymbol{H}_0$ would be indication of evidence of dependence among violations. Often, this implies clustering or patterns that the VaR model fails to capture.

9.7.4 Controlling False Discoveries

Given our large number of models, controlling error rates becomes crucial. in particular, we are seeking to control the False Discovery rate, which is the expected proportion of false positives among all hypotheses found to be significant. In this context, suppose we make FD false discoveries out of D total discoveries. Controlling the FDR amounts to

$$\text{FDR} \;=\; \mathbb{E}\Big[\frac{FD}{D}\Big] \leq \alpha. \tag{154}$$

The Benjamini & Yekutieli (BH) procedure, first introduced by Benjamini and Hochberg [1995] is shown in Algorithm 4 and controls the FDR.

Algorithm 4 Benjamini & Yekutieli Procedure

Input: p-values $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_M$ for hypotheses $H_1, H_2, \ldots, H_M$
Input: Target FDR level α
Output: Rejection set $\mathcal{R}$
1: Sort the p-values in ascending order: $\hat{p}_{(1)} \leq \hat{p}_{(2)} \leq \cdots \leq \hat{p}_{(M)}$
2: Calculate

$$k = \max\Big\{\, i = 1, \ldots, M \;\Big|\; \hat{p}_{(i)} \leq \frac{i}{M}\,\frac{\alpha}{\sum_{i=1}^{M}\frac{1}{i}}\Big\}. \tag{155}$$

3: Reject all hypothesis associated with p-values $\hat{p}_{(1)}, \ldots \hat{p}_{(k)}$:

$$\mathcal{R} \leftarrow \{\, H_{(1)}, H_{(2)}, \ldots, H_{(k)}\} \tag{156}$$

4: **return** $\mathcal{R}$

We employ this specific procedure for each individual stock as its own "family" of tests (one test per horizon). This means we are considering M tests at a time, M being our total number of models.

The main advantage of this approach is that it ensures (154) holds regardless of the distribution of the test statistics, which is importance since because of the use of a rolling window means the test statistics are correlated in possibly complicated ways. This property comes from the harmonic factor $c_M = c_{55} = \sum_{i=1}^{55} 1/i \approx \ln 55 \approx 4.6$, which means the rejection thresholds shrinks by a factor of roughly $\ln M$ reducing the power of the procedure.

10 Results

We present our empirical findings for the full cross-section of the eleven technology stocks, using the evaluation metrics introduced in Section 9. In Appendix E we present supporting results.

10.1 Data Exploration

We first examine key descriptive statistics for our eleven technology stocks over the full sample period and the realized kernel. In table 1 we present some descriptive statistics for each of the stocks return series. We use the Augmented Dickey Fuller (ADF) test developed Dickey and Fuller [1979] to test for stationarity. All series reject the null of a unit root at the $\alpha = 0.001$ level (ADF $p < .001$), which is a very strong indication that returns are stationary.

Ticker	n	Mean	ADF p	JB p	Skewness	Kurtosis
AAPL	1798	0.09	$< .001$	$< .001$	-0.22	5.31
ADBE	1798	0.05	$< .001$	$< .001$	-0.72	8.95
AMD	1798	0.12	$< .001$	$< .001$	0.05	2.68
AMZN	1798	0.07	$< .001$	$< .001$	-0.18	4.18
ASML	1798	0.08	$< .001$	$< .001$	-0.50	4.75
IBM	1798	0.03	$< .001$	$< .001$	-0.50	10.73
INTC	1798	-0.04	$< .001$	$< .001$	-1.16	16.54
MSFT	1798	0.08	$< .001$	$< .001$	-0.26	7.33
NVDA	1798	0.18	$< .001$	$< .001$	-0.24	4.55
TSLA	1798	0.14	$< .001$	$< .001$	-0.05	3.53
TSM	1798	0.08	$< .001$	$< .001$	-0.01	3.81

Table 1: Descriptive Statistics of Daily Returns by Ticker

Likewise, the Jarque–Bera tests produces $p < .001$ for every ticker, indicating significant deviations from normality in each distribution. Mean returns vary across stocks, where NVIDIA delivered the highest average daily gain, whereas Intel underperformed. Other firms such as Apple (0.095%), Microsoft(0.085%), and Tesla (0.144%) also show a positive mean, suggesting near zero but positive average returns during the sample period. Kurtosis values exceed the normal benchmark of 3 for every stock, peaking at 16.54 in Intel. Such high kurtosis indicates very heavy tails and a higher probability of extreme returns on both sides.

Figure 4 shows a graphical representation of r_t for each stock. We see evidence of volatility clustering, since large changes tend to be followed by large changes and small changes tend to be followed by small changes.

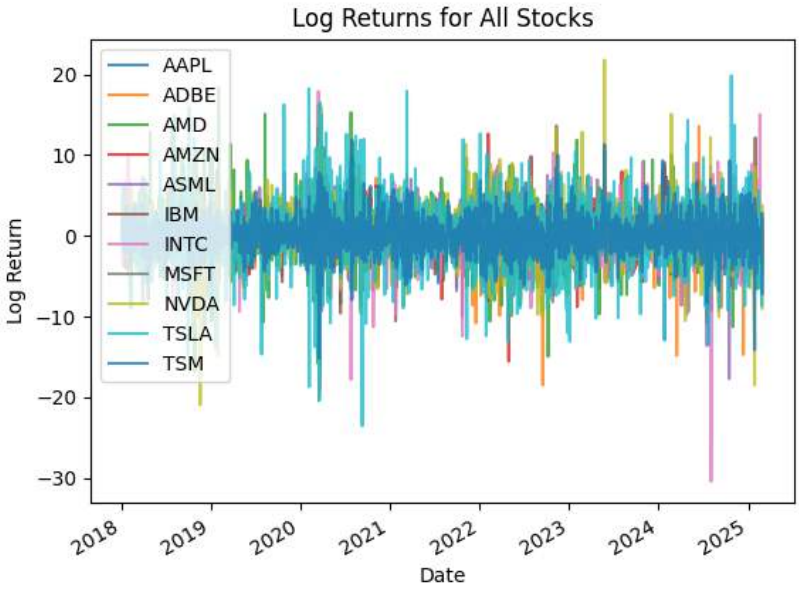


Figure 4: Log returns r_t of (8) of the 11 technology companies.

Figure 5 displays the realized variance in (9) and the realized kernel of Barndorff-Nielsen et al. [2009]. When there is a volatility spike, the realized kernel sits just above the realized volatility. The kernel "sharper peaks" better capture the COVID-19 induced market plunge in 2020 and late-2021/early-2022. This property comes from the utilization of rich ultra high-frequency intraday trade data. However, since both variables are consistent estimators of the quadratic variation QV_t in (5), across the full sample, their long-run averages are almost equal.

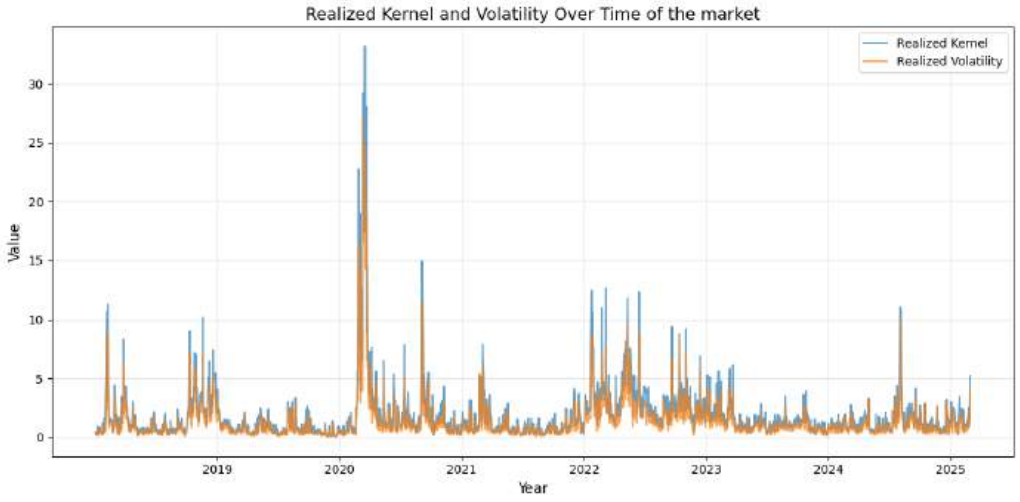


Figure 5: Plot of the realized variance in (9) and the realized kernel of Barndorff-Nielsen et al. [2009] in Section 4.

In Sample Results The Elastic Net regularization parameters (λ, α) vary substantially from ticker to ticker. For more stable equities such as Apple and Microsoft, low values of $\lambda = 0.10$ and $\alpha = 0.10$ mean minimal penalization. We do see that $\alpha = 0.10$ implies that the model favors the Ridge L_1 penalty suggesting that the feature space is correlated. However, for AMD and NVIDIA for example, the model model selects $(\lambda = 1.58, \alpha = 0.50)$ and $(\lambda = 1.63, \alpha = 0.50)$ respectively, which demonstrates the need for more aggressive penalization and a more even balance between L_1 and L_2 penalties. The returns of these equites are more unstable which could lead to idiosyncratic swings and a lower signal to noise ratio in the realized kernel estimator of the volatility. Setting a higher regularization strength therefore raises training-set errors but substantially reduces the model's sensitivity to noisy fluctuations.

The Lasso model, constrained solely by an L_1 penalty, uniformly favors modest regularization ($\lambda = 0.10$) for most equities, driving uninformative coefficients to zero while preserving key predictors. Notably, AMD and NVIDIA again requires a substantially larger penalty ($\lambda = 1.58$,$\lambda = 0.68$).

Ridge regression selects the maximum penalty strength available in our search space for all eleven tickers. This choice reveals that even the largest degree of L2 shrinkage was necessary to stabilize coefficient estimates in the presence multicolinearity across our regressors. By pushing coefficients toward zero but not eliminating them entirely, the model still produces a unique solution, even in the presence of multicolinearity without discarding potentially informative predictors.

Random Forest achieves optimal performance with a maximum depth of six for most stocks. This moderate number prevents the trees from overfitting too much. However, NVIDIA and ASML require deeper trees (depth = 10) which means these stocks favor more complicated tree structures in order to fit nonlinear dependencies. The number of trees varies from 200, to 400 for tickers such as INTC, IBM, NVDA, ADBE, and ASML. This indicates that those stocks exhibit greater variance that benefits from additional averaging. Thus, the Random Forest hyperparameters reveal the model is able to find a balance between model complexity and variance reduction uniquely to each security's volatility dynamics.

Support Vector Regression displays almost the same hyperparameter settings across all stocks, with a penalty $C = 10$. TSM is the exception, which adopts $C = 1$ to allow greater tolerance for training-set deviations. This lower value of C indicates that the SVR requires a smaller margin to avoid overfitting to small fluctuations. The uniform choice of $\varepsilon = 0.10$ across all equities suggests a shared threshold for ignoring minor prediction errors.

XGBoost uses a learning rate between 0.05 and 0.10. A smaller learning rate generally helps prevent overfitting, as it forces the model to learn more gradually and potentially generalize better. The model uses 300 trees for every ticker. Since a lower learning rate usually leads to more trees being necessary to achieve the same performance, the higher learning rates for MSFT and IBM with the same number of trees implies that for these tickers the model was able to learn patterns in fewer rounds. The maximum depth of each tree for most securities is 3. That is, the model limits its trees to three splits, such that trees become simple decision rules that reduce overfitting on noisy data. For ADBE

and ASML the number of splits is increased to 5, indicating that their volatility structure exhibits more complex nonlinear relationships that need deeper trees.

Model	HP	AAPL	MSFT	AMD	INTC	TSM	TSLA	AMZN	IBM	NVDA	ADBE	ASML
EN	λ	0.10	0.10	1.58	0.25	0.25	0.25	0.25	0.10	0.63	0.63	0.25
	α	0.10	0.10	0.50	0.10	0.10	0.10	0.90	0.10	0.90	0.10	0.10
Lasso	λ	0.10	0.10	1.58	0.10	0.10	0.10	0.25	0.10	0.63	0.25	0.25
Ridge	λ	10	10	10	10	10	10	10	10	10	10	10
RF	`Depth`	6	6	6	6	6	6	6	6	10	6	10
	`trees`	200	200	200	400	200	200	200	400	400	400	400
SVR	`C`	10	10	10	10	1	10	10	10	10	10	10
	ε	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
XGB	ν	0.05	0.10	0.05	0.05	0.05	0.05	0.05	0.10	0.05	0.05	0.05
	`Depth`	3	3	3	3	3	3	3	3	3	5	5
	`trees`	300	300	300	300	300	300	300	300	300	300	300

Table 2: Optimal Hyperparameters by Model and Ticker

In Table E.1, E.2, E.3 we present our in-sample results. Across most stocks the GARCH-MIDAS-t model consistently yields the lowest AIC values. It was the optimal model in 7 of 11 series (Apple, Microsoft, IBM, Adobe, Amazon, Intel and TSCM). This implies that the low-frequency component in GARCH-MIDAS significantly improves the fit, at the cost of only a handful extra parameters. The long-run frequency part of the GARCH-MIDAS is even so useful for improving the fit that when we extra penalize the additional parameters with the BIC, it remains the model with the lowest values across the stocks.

As shown in Table 1, the returns for most stocks are far off a normal distribution. Under a Gaussian assumption, the probability mass in the tails is too small, causing the likelihood function to drop dramatically when outliers occur. As a result, models who assume normality-error GARCH-MIDAS underfit to such a degree it cannot make up for not incurring costs for not including any shape parameters. The generalized beta of the second kind (EGB2) and the asymmetric Student-t (AST) both allow more flexibility in the tail and skewness (Appendix E). While in principle this in principle can improve the in-sample fit, in practice the gain in log-likelihood is often not large enough to justify the extra two or three parameters once we add the the parameter penalty in AIC or BIC. By adding only one additional parameter, the Student-t distribution allows to model fatter tails without a large complexity penalty. This translates into the lowest AIC and BIC in 7 of 11 stocks, and the highest log-likelihood across stocks.

We see that the Kalman Filter SV model yields superior AIC and BIC values for series with kurtosis closer to 3. We assume standard normal innovations in the observation equation, however, the latent states are also stochastic. This turns the unconditional distribution of the returns into a mixture of normals, which since these series are almost mesokurtic, might improve the fit just enough such that adding an additional tail parameter in the Student-t distribution might not be beneficial [Kim et al., 1998].

10.2 Out Of Sample Results

Following the in-sample evaluation, we now discuss the forecasting and out-of-sample evaluation results.

10.2.1 Relative RMSE and MAE

Figure 6 plots, for each forecast horizon $h \in \{1, 5, 10, 20\}$, the average RMSE of the ten best models, averaged across the stocks and normalized so that the lowest RMSE at each horizon equals 1 (Section 9.3.1). All other bars show how many times larger a given model's error is relative to the best performing model. For example, a height of 1.20 indicates a 20% higher RMSE than the leader. Numerical results can be found in Appendix E.

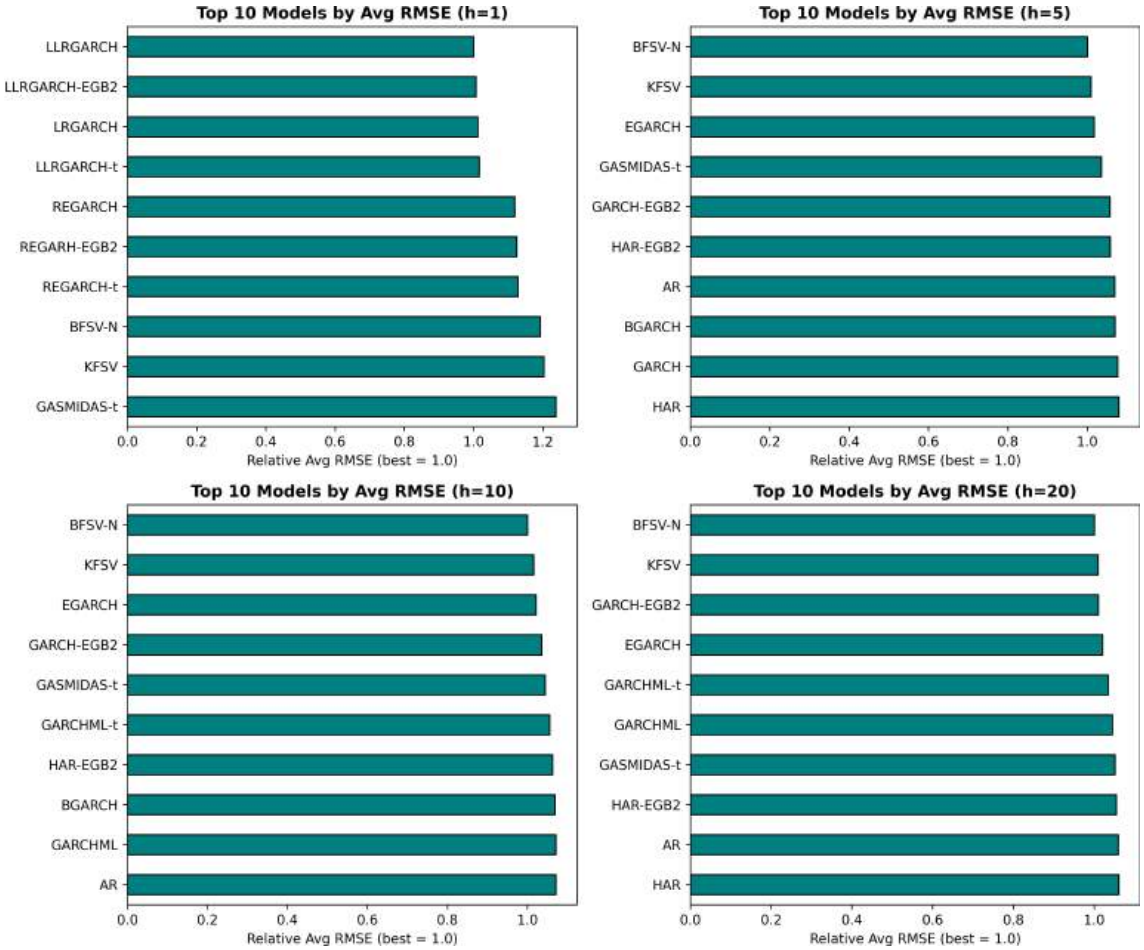


Figure 6: Bar plot of the relative RMSE across the best performing models where the RMSE of the best model is scaled to one.

At the one-step-ahead horizon ($h = 1$), the log-linear GARCH with Normal and EGB2 innovations occupies first and second place, followed by the linear Realized GARCH and the log-linear GARCH–Student-t. Ranks 5–7 are taken by realized EGARCH under Nor-

mal, EGB2 and Student-t errors, and the top ten is completed by the bootstrap SV with normal innovations and Kalman-Filter SV model.

At on week ahead forecasting ($h = 5$) we see that the situation is the other way around. The bootstrapped normal SV model and the Kalman Filter are leading the top 10, while the EGARCH with normal innovations follows closely at third place. The GAS-MIDAS model with student t errors also provides strong performance here in fourth place, followed by the vanilla GARCH and HAR model, both with EGB2 innovations. The AR model with normal errors comes in at place 7 outperforming the Bayesian GARCH which in turn outperforms the plain GARCH and HAR model, all assuming normal errors.

At two weeks ($h = 10$), the SV and EGARCH–Normal models remain dominant, joined in the top ten by the GARCH-ML under both Normal and Student-t innovations. This suggests that allowing volatility to affect the mean is useful for multi-step forecasts. The one-month horizon ($h = 20$) yields a very similar ranking, with only minor re-ordering among these same leaders.

The results for the MAE are shown in Figure E.5. For $h = 1$ both metrics agree that the log-linear Realized GARCH specifications are the best performing models, with some minor ranking differences compared to the RMSE. As the horizon lengthens to one week ($h = 5$), two weeks ($h = 10$) and one month ($h = 20$), both the MAE and RMSE agree that the bootstrap-SV and Kalman-filter SV models deliver the lowest errors. An interesting difference is that when we increase the forecast horizon, we see that the MAE keeps some linear specification in the top 10, while for the RMSE they complete vanish from the ranking after making one-step-ahead forecasts.

10.2.2 Diebold-Mariono Test Results

We applied the panel DM test pairwise across all models at each forecast horizon to identify significant performance differences. For each horizon, every model was matched against each other model, and a "win" was counted whenever one model's forecast error was significantly lower than another's at 5% significance.

Figure E.6 illustrates the Diebold–Mariano win–loss results utilizing a network structure. Each node is a forecasting model and each directed edge points from the "loser" to the "winner." The edges are horizon specific and the nodes grow by the number of edges they receive. That is, how bigger the node, the more the model "wins". Every node has color representing which distribution assumption it makes. What immediately stands out is the dominance of models which assume normal innovations (blue nodes). Models such as the Kalman Filter SV, the bootstrapped-SV with normal innovations, the HAR and GARCH-ML with Gaussian shocks, and even the classic GARCH all occupy the largest, most highly connected positions in the network. By contrast, models who assume Student-t (red) and AST (purple) errors are much less connected, resulting in much smaller nodes. A handful of EGB2 based models (dark-green) have nodes that grow to respectable sizes, especially at longer horizons. However, they never match the consistent centrality of the normality-error nodes. In short, Figure E.6 shows across one-day to four-week forecasts, Gaussian innovations delivers the most robust out-of-sample volatility accuracy. In Figure 7 we exactly show per horizon which models accumulate the most victories at each horizon.

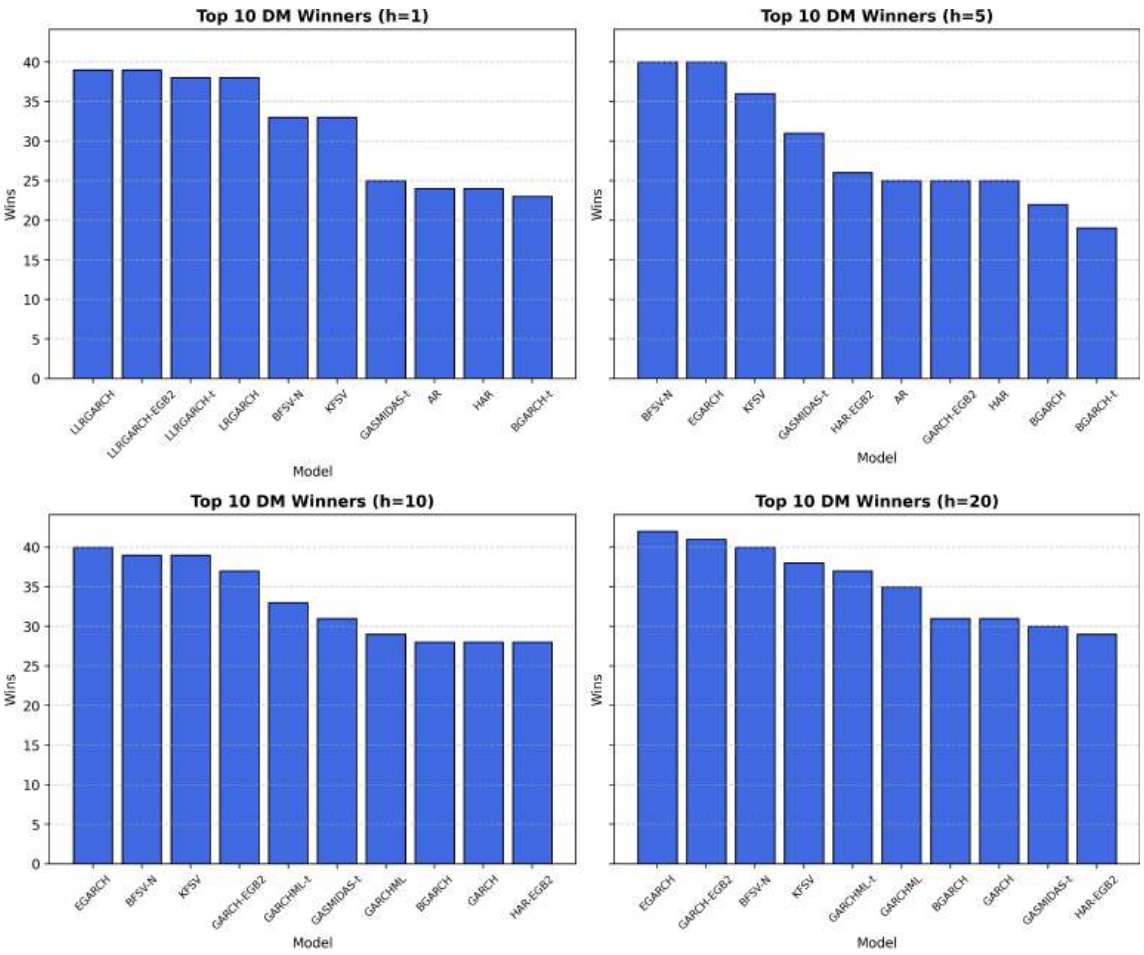


Figure 7: Bar plot of the models with the most "wins" according the panel DM test.

At the one-day horizon ($h = 1$), the realized-GARCH family clearly leads. The log–linear realized GARCH with normal and EGB2 distributions tops the list with 39 significant wins each, closely followed by its Student-t variant and the linear GARCH (normal errors) with 38 wins apiece. Next in line are the Bootstrapped Normal SV and the Kalman-filter SV models (33 wins each), then GAS–MIDAS-t (25 wins), AR(1) and HAR (24 wins each), and finally Bayesian GARCH with Student-t errors (23 wins). These results underline how effective the realized-GARCH framework is for one-step-ahead variance forecasting.

Extending to a one-week horizon ($h = 5$), the picture shifts away from pure realized GARCH. Both the Bootstrapped Normal SV model and EGARCH (normal errors) share the top spot with 40 wins. The Kalman-filter SV model follows with 36 wins, showing that SV-based approaches remain competitive at medium horizons. GAS–MIDAS-t improves to 31 wins—likely thanks to its long-run smoothing component—while HAR (normal and EGB2), AR(1), and GARCH-EGB2 cluster around 25 wins. Bayesian GARCH (normal) just receives more edges than its Student-t variant, which concludes the top ten.

At two weeks out ($h = 10$), EGARCH takes the lead with 40 wins, with Bootstrapped Normal SV and Kalman-filter SV just behind with 39 wins each. GARCH-EGB2 scores 37 wins, outperforming plain GARCH (28 wins) and matching Bayesian GARCH (normal)

and HAR-EGB2. The GARCH-ML model,which allows the mean to depend on volatility, also is useful here, beating its Student-t version and GAS–MIDAS-t. By four weeks ($h = 20$), EGARCH (normal) extends its advantage to 42 wins, trailed by GARCH-EGB2 (41 wins), Bootstrapped Normal SV (40 wins), and Kalman-filter SV (40 wins). GARCH-ML-t (37 wins), GARCH-ML (35 wins), and classic GARCH (31 wins) fill the middle ranks, while GAS–MIDAS-t (31 wins), Bayesian GARCH (30 wins), and HAR-EGB2 (29 wins) form the bottom section of the top ten.

Figure 7 shows which models rack up the most wins overall, but it does not show how those ten "elite" models compare to one another. For that, we turn to the 10×10 DM tables in Figures E.1, E.2, E.3, and E.4. In these tables, orange cells mean the row model beats the column model at 5% significance, and blue cells mean their forecasts are statistically indistinguishable. At $h = 1$ (Figure E.1), nearly all cells are blue. Even though these ten models beat most others, among themselves they produce virtually identical one-day-ahead errors. By $h = 5$ (Figure E.2), many more orange cells appear. Bootstrapped Normal SV and EGARCH dominate the group, while a clear middle tier (KFSV, GAS–MIDAS-t, HAR-EGB2) loses to the leaders but still outperforms the lower half. At $h = 10$ and especially $h = 20$ (Figures E.3 and E.4), the top four models—EGARCH, GARCH-EGB2, Bootstrapped Normal SV, and Kalman-filter SV—no longer beat each other significantly, yet each records significant wins against the other six finalists.

10.2.3 Model Confidence Set

The results of the Model Confidence Set (MCS) analysis using Borda scoring are shown in Figure 8

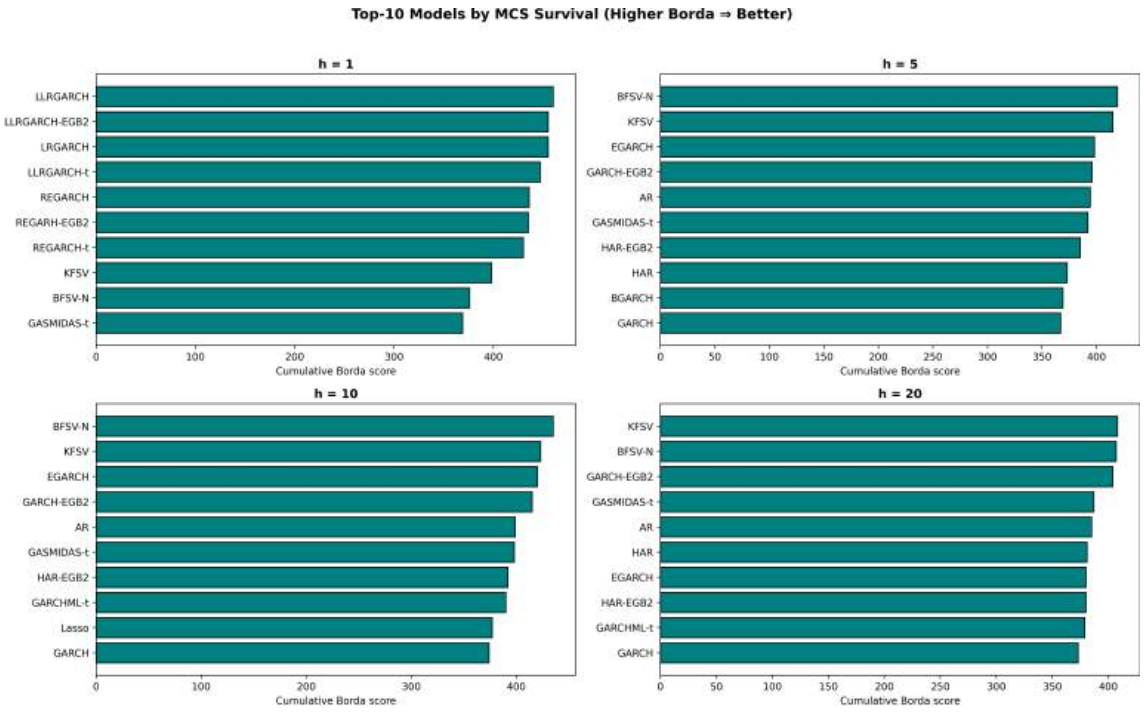


Figure 8: Top-10 models by cumulative Borda score (Section 9.5).

Concerning the one-day-ahead forecasts, the very top of the ranking is dominated by variants of the Linear and Log-Linear Realized GARCH family. In fact the plain Log-Linear Realized GARCH under Normal, EGB2, and Student-t errors occupy four of the five highest spots, followed closely by the linear Realized GARCH. Three versions of the realized EGARCH model (standard, EGB2, and Student-t) form the next block of models. The good performance of linear and log-linear GARCH specification indicate that incorporating the realized kernel directly is a valuable addition. Parameter driven models such as the Kalman Filter SV, Bootstrapped-normal-SV follow in forecast performance. As with the DM test—the GAS–MIDAS-t model makes the top ten. A key difference, however, is that the Bayesian GARCH does not appear in the Borda ranking, whereas the DM test (Figure 7) did classify it as one of the top models.

By five-day horizons the bootstrapped normal SV takes first place by a small margin compared to the Kalman Filter SV model. EGARCH with Gaussian innovations follows close behind in third, while GARCH-EGB2 takes fourth place. The simple AR(1) model, GAS–MIDAS-t, and two HAR-based specifications (HAR and HAR-EGB2) all survive and obtain places six through eight. The Bayesian GARCH and the vanilla GARCH (normal innovations) complete the top ten. This ordering closely mirrors the DM-test in Figure 7.

At ten days out, Boostrapped normal SV model again claims the highest Borda score, while the Kalman Filter SV and EGARCH (normal) are tied for second place. GARCH-EGB2 stays within the top four, and the AR(1) takes fifth place. GAS–MIDAS-t and HAR-EGB2 occupy position six and seven, while the GARCH-ML-t model climbs into eighth. Surprisingly, we find LASSO regression (ninth) as the first machine learning model to make it within the top ten of best performing models, leaving the classic GARCH as tenth.

Finally, at a full month ahead the Kalman Filter SV model beats the Bootstrapped-normal-SV model and GARCH-EGB2 which come in at second and third. GAS–MIDAS-t moves up into fourth, while AR(1) and the basic HAR model make up the middle of the top 10. EGARCH under normal errors appears only in seventh, slightly behind HAR at six, signaling that long-memory aggregation overtakes pure leverage adjustments at very long horizons. HAR-EGB2 and GARCH-ML-t complete the lower half of the top ten, with plain GARCH(1,1) again in tenth place.

10.2.4 Value at Risk

The results of the VaR analysis can be found in Appendix **??** on an individual basis. In summary across all stocks, XGBoost yields the most negative VaR predictions when losses are extreme, which indicates the most conservative tail coverage. Random Forest is often the second most conservative, particularly for high-volatility stocks, though its behavior can vary a little. The linear models (Lasso, Ridge, ElasticNet) generally give similar results. Each provide moderate predictions that cover most but not the largest losses. SVR predicts the least negative VaR, making it the least conservative. Furthermore, more volatile stocks (TSLA, NVDA, AMD, ASML) show wider model spreads and larger dominance on tail events by ensemble methods as Random Forest and XGBoost, while calmer stocks (MSFT, IBM, AMZN) show all models closely together with minimal difference. In each case, when the volatility of the returns seems to be higher, VaR estimates for all models grow more conservative and the differences among models becomes more clear.

Furthermore, the results for deep learning models follows a clear pattern. For low-volatility stocks (TSM, MSFT, IBM, AAPL), all models predict small and similar VaRs . The Transformer and Feedforward network are slightly less negative and the encoder-attention-decoder RNNs slightly more negative, but differences are small. For high-volatility stocks (TSLA, NVDA, AMD), predictions are less uniform. LSTM- and GRU-based models, especially the encoder-decoder versions, produce much more negative VaRs while Feed-forward and Transformer give more optimistic bounds. However, in all cases, no model perfectly captures the very worst actual losses. The predicted lines often understate extreme tails where actual returns occasionally plunge below all VaRs.

The backtesting results for one-day-ahead VaR forecasts are not reliable in the case of machine learning models. The use of historical residuals in the unconditional coverage (UC) test for example yield inflated FDR-adjusted p-values. By construction, we set the VaR threshold in such a way that a fixed fraction of the historical errors lie beyond it. Since we are calculating the cutoff on the same empirical sample on which we evaluate backtests, the long-run frequency of violations will converge to the significance level 5%), thereby artificially improving UC performance. Furthermore, since we remove and add observations the historical distribution, as well as the VaR threshold, will jump around in each iteration of the rolling window.

For the observation driven models, we see that GARCH-type models achieve high UC p-values in most cases, meaning that we cannot reject that violations occur at the correct frequency. This is especially the case of models who assume error distributions with heavy tails, such as the Student-t,AST and EGB2. Esec Because they have fatter tails, the 95% VaR sits further out in the loss distribution, the number of days when actual losses breach that VaR closely matches the 5% rate we expect. In contrast, standard volatility models (AR/HAR models and GARCH-family under the normal, Student–t, AST, or EGB2) tend to fail the UC test at h=1. These models often underestimates, or overestimates risk at the 5% level due to misspecified errors distributions, as shown by low p-values.

In the case of the EGB2 specification, its very heavy-tailed form pushes the 95% VaR threshold far below even the worst observed losses. In other words, because EGB2 assumes so much mass in the far tails, it predicts a loss level that is more extreme than any return we actually saw. Hence VaR predictions are much more negative than the minimum realized return. This makes the model overly conservative. It essentially never sees a breach, signaling a clear failure of the unconditional coverage test.

Because we fit an EGB2 error by maximum-likelihood to a return series that contain at least one or two very large outliers, caused by the COVID-19 pandemic in 2020 and widespread supply chain issues in 2024, SQPLS will often explain those outliers by driving the left and right tail shape parameters $p, q \geq 2$ toward values that make the distribution extremely heavy-tailed. That is, the likelihood can be increased by driving p and q to values very close to their bound, which puts more probability mass out in the extremes, and driving the a 95% VaR to values far below the observed loss. Figure 9 provides a good example of this. In 2024 Intel suffered a 30% decrease in returns. To match this event, we end up predicting even more extreme losses at the 5% level (175% in Figure 9), because as $p \to 2$, the entire tail stretches.

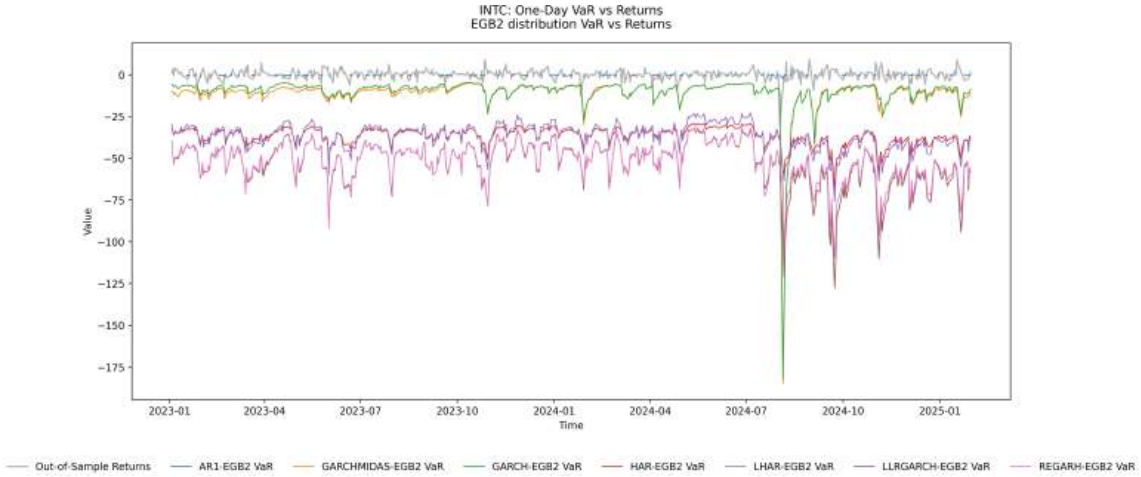


Figure 9: Plot of the one-step-ahead daily 95% Value at Risk estimates for Intel for models with EGB2 innovations.

Similar dynamics cause very poor performance for the AST distribution in the UC test. Furthermore, models with Student-t innovations yield higher p-values for the UC test across the board, suggesting that the Student-t is just heavy enough to capture genuine fat-tail behavior without the runaway flexibility of AST or EGB2. However, when innovations are assumed normal, UC p-values fall off again probably because the normal distribution's thin tails systematically understate extreme losses, yielding too many VaR breaches.

The independence (IND) test shows little variation across models. At $h = 1$ virtually in almost all model specifications we fail to reject the null of independent VaR violations, indicating no evidence of clustering of VaR exceptions at the daily horizon. The lack of evidence of serial correlation in the hit rate essentially means that it behaves as a Bernoulli random variable. Even if we grossly overestimate the risk, breaches still happen on days that look completely random. However, we should be careful with trusting these results. For similar reasons as fot the UC test, the flexibility in the tail parameters of the AST and EGB2 distribution causes overestimation of the risk to such a degree that brings VaR breaches to near 0%. Conversely, assuming Gaussian errors puts the VaR line so close to the center of the log returns that virtually every large move breaks it, inflating the hit rates toward 100%. In both cases, the hit rate is stuck at the boundary (either only hits or no hits), leaving no variation in the sequence. Our rolling-window refitting reinforces this effect. If today's return breaches the VaR, this observation will become part of the estimation sample of tomorrow's volatility forecast. Because that extreme loss in it input data of the model, it raises its estimated volatility and tail-shape parameters on the new window, likely resulting in another breach. Therefore, the independence test has almost no power and will return a high p-value.

Since all model classes satisfy the i.i.d violation assumption under the independence test, the conditional coverage test performance is determined entirely by the UC test. Accordingly, the conditional coverage (CC) test essentially reflects the UC results.

11 Key Results

Applying the TOPSIS scores as described in Section 9.6, we obtain a unified model ranking that balances RMSE, MAE, DM and MCS Borda scores. Figure 10 presents the resulting TOPSIS scores and the overall best models at each horizon.

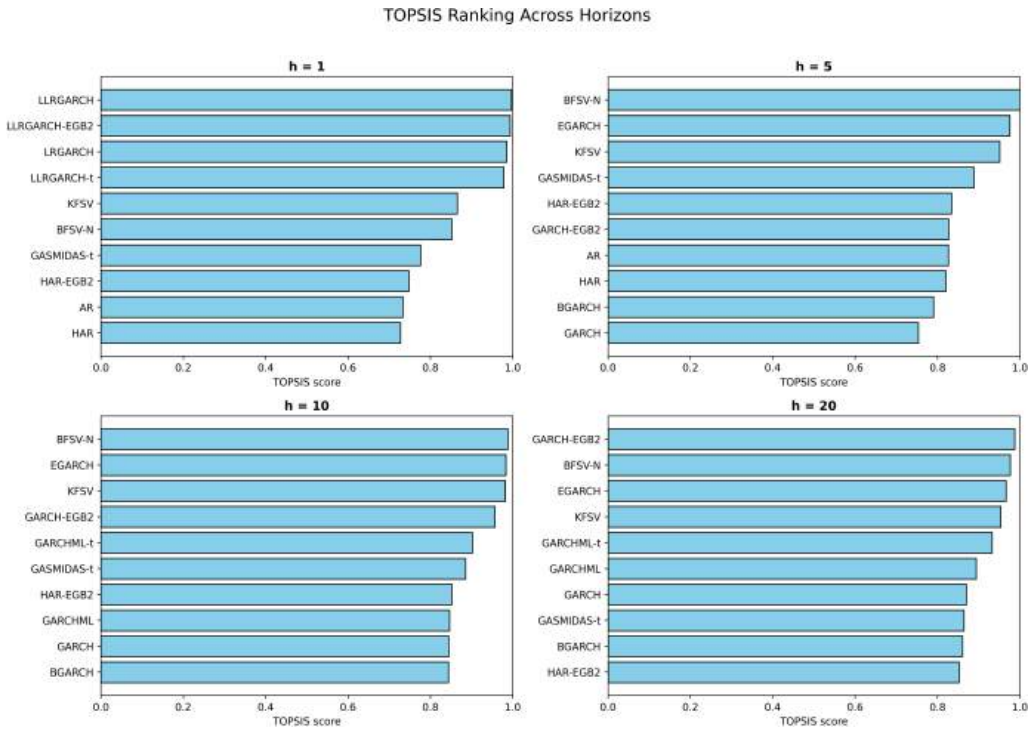


Figure 10: Bar plot of the models with the most "wins" according the panel DM test of section 9.4.

From Figure 10 it becomes clear that one-step-ahead forecasting performance is dominated by GARCH specifications that allow the conditional variance to be affected by the realized kernel through the measurement equation. The incorporation of the realized measure allows the model to utilize high-frequency data as a more accurate measure of volatility compared to relying solely on daily returns. The joint modeling of the returns, and the volatility allows for a more nuanced understanding of how returns and volatility influence each other. Combined with their ability to capture the asymmetry and the leverage effect, these properties makes them well suited to respond more quickly to changes in market conditions and explain their superior performance when making short term volatility forecasts.

A few models distinguish themselves by performing well consistently over all forecast horizons. In particular,the SV models under a normal distribution, the Bootstrapped normal SV model and Kalman Filter counterpart show up near the top for $h = 5$, 10, and 20, and were still competitive at $h = 1$. The strength of the SV model mainly comes from modeling the volatility as an unobserved latent process. Furthermore, the use of a Kalman filter yields smoothed volatility estimates that are less sensitive to measurement

noise. This ability of the model to capture signal over noise produces more stable multi-step forecasts. By using a particle-filter (or bootstrap) approach to fit the SV model, we do not rely on a Quasi Maximum Likelihood approach to estimate the parameters. Instead, the model produces many possible parameter sets, each weighted by how well it explains past data. When we forecast, we average over these particles which makes the variance forecasts more stable.These features explain why SV models with Normal errors consistently outperform or match the best GARCH-type and HAR-type competitors across all forecast lengths.

The solid performance of the EGARCH reflects its ability to capture two important features of financial volatility. By modeling the logarithm of the variance, EGARCH automatically guarantees positive forecasts. This means we can estimate the model more easily, with less parameter constraints. which makes the estimation more stable. Second, EGARCH explicitly incorporates that big shocks in either direction, positive or negative affect the conditional variance. Second, the inclusion of the leverage effect where negative shocks (large price drops) and positive shocks (price jumps) impact volatility differently allows the model to make accurate h-step-away volatility forecasts.

In general, in the universal TOPSIS ranking, the clear winners are the models who find the balance between reacting quickly to new shocks and preserving the slow-moving volatility signal. In the EGARCH for example, modeling the log-variance smooths out day-to-day noise and guarantees positive volatility forecasts, while its leverage term immediately amplifies downside moves. Furthermore, the GAS–MIDAS, which feeds in macro-scale inflation trends for persistence and uses score-driven GAS updates to react sharply when returns deviate from expectation. Exactly this design is what these models allow to dominate in forecasting volatility on both the very short and medium term.

One striking finding based on Figure 10 is that none of the purely machine learning or deep learning models are among the top-ranked performers. To investigate this further, Figure 11 presents the average training error (blue) against the validation MSE for the Transformer. The curves are obtained by running the Transformer for 200 epochs, on the initial training window on each stock, recording the MSE at each epoch (one full forward an backward pass), and then averaging these per epoch across all stocks.

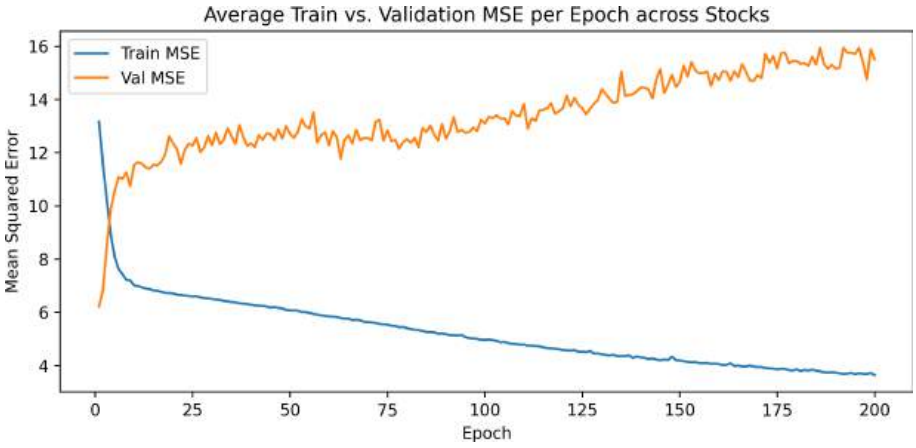


Figure 11: Plot of the validation and training loss of the Transformer.

We see that the training MSE (blue) falls smoothly and quite rapidly from its initial value. This is what we would expect to see. The model is learning the patterns on the training data effectively, such that the error keeps decreasing as we perform more forward and backward passes. In contrast, the validation MSE (orange) rises and even oscillates as epochs increase. By epoch 200 the gap is large with a mean training MSE near 4–5, whereas the validation MSE is around 14–15. The steadily declining training error alongside a rising validation error reflects that a classic case of severe overfitting. As training proceeds, the model continues to fit the training data (driving down its MSE) but generalizes increasingly poorly, so the validation loss climbs. In other words, by the end of training the large train–validation gap confirms that the model has very low bias on training data but high variance on holdout data. An interesting observation from Figure 11 is that the Transformer reaches its optimal performance immediately after conducting one epoch. This implicates the Transformers fits whatever patterns it can in the very first pass and then immediately fail to to capture any deeper dynamics on subsequent epochs [Li et al., 2022].

Several factors could simultaneously cause this poor performance. First, despite our sample consisting out of seven years of daily volatility observations, in the context of deep learning models, this is considered as a very limited dataset. Because of the enormous number of parameters, the transformer finds a combination of weights that drives training loss down very quickly, remembering each training set, including the noise. Furthermore, in contrast to econometric time series models, the Transformer does not process data sequentially. Despite the auto-regressive structure of the model, positional encoding orders the observations, but does not imply that close time steps should be more strongly related. That means that, despite the ability of Transformers to learn long-run dependencies, it must learn correlations from data. The realized kernel estimate of the underlying volatility can be driven by short-lived shocks and mean-reversion. Econometric models are structured to describe these relations by definition. Since the Transformer does not have such mechanism, it needs much more observations than available in our study to model temporal relationships [Li et al., 2022].

Second, the core of the Transformer is the self-attention mechanism. Because self-attention treats each time step identically computing similarities between time steps using scaled dot-products among all time step embeddings. In this framework, any two time steps can influence each other if their query/key features align. It might be the case that high-noise can produce inflated dot-product scores. The softmax normalization then amplifies these events with high relevance such that the model reacts to noise rather than signal [Doe et al., 2023].

Furthermore, under the rolling forecasting setting the previous and current training samples are only one data point apart from each other. Since making longer predictions typically requires longer input sequences, training sample have a higher degree of similarity, only differing by one observation. This means that we are effectively tune the weights on the same pattern over and over again. That redundancy means the model sees very little new information from sample to sample and memorizes the repeated patterns and noise, failing to generalize [Li et al., 2022].

12 Discussion

The main objective of this study was to determine whether equipping modern machine- and deep-learning techniques with stock-specific daily market sentiment data could improve daily volatility forecasts for major technology stocks. The results show that none of the top-ranked models at any forecasting horizon included any machine-learning or deep-learning models, indicating that the inclusion of qualitative news information did not yield an improvement over small parameter econometric specification relying solely on past returns and realized estimators of volatility.

Despite the state-of-the-art Transformer model's proven ability in natural language processing, our findings demonstrate that this proficiency does not extend to volatility forecasting. In the context of our data, qualitative sentiment signals appears to offer no additional predictive power beyond that already captured by econometric measures of return and high-frequency-based volatility estimators.

Our results confirm the findings of Hansen and Lunde [2011], who demonstrated that jointly modeling returns and high-frequency volatility estimators through a measurement equation benefits forecasting accuracy, particularly at short horizons. In contrast, studies such as Ramos-Pérez et al. [2021], Goel et al. [2024], and Frank [2023] document cases where machine learning and deep-learning approaches outperform traditional models in financial time-series forecasting. Similarly, Rahimikia and Poon [2020] and Audrino and Offner [2020] report that HAR models fitted with news-based sentiment yield significant accuracy gains during turbulent market conditions, and Lei et al. [2021] finds that incorporating Twitter-derived sentiment into an LSTM reduces forecast errors. However, our findings align with Bodilsen and Lunde [2025], who observe only marginal effects from firm-specific news sentiment, and with Díaz et al. [2024], who caution that any edge from ML or deep-learning methods in volatility forecasting is far from guaranteed and often depends on extensive tuning or very large datasets.

These results carry important implications for risk managers and financial professionals. By identifying which models provide the most accurate volatility forecasts, our study supports improving risk-sensitive processes in the financial industry—such as trading strategy development, option valuation, and portfolio risk control. Furthermore, the lack of informative power in daily sentiment data suggests that market-pricing mechanisms efficiently incorporate news as it arrives. Therefore, decision-makers should be cautious when relying on sentiment signals extracted by advanced NLP architectures for volatility forecasting.

While our study offers valuable insights into high-frequency volatility forecasting, it suffers from several limitations. First, our focus on the largest eleven technology stocks over the 2018–2025 period means that our findings may not transfer to other sectors or time periods. Tech stocks are notoriously volatile and frequently traded. Therefore, the performance of our considered models could differ on indices which show to be a lot less liquid and more stable.

Second, our news-sentiment measure relies exclusively on headlines sourced from Business Insider [2025]. This introduces due to single source dependence because Business Insider [2025] may not fully posses records of all news causing shifts in the market. It is very well possible that traders pay more attention to other outlets such as social media platforms as

Twitter that provide qualitative data not captured in traditional news. All these factors mean that our sentiment findings are conditional on the source we chose. Furthermore, not every stock had at least one article per day, which forced us to impute missing sentiment values for days without articles. This imputation, using a state-transition approach, introduces uncertainty and potential bias. If important news was missing from our source or occurred outside trading hours, our sentiment measure might miss import observations. Since we only extract the title of the articles, our approach does not capture things like the tone or the specific content of articles. Assigning a label solely based on the article's title might be too granular to encode the data with actual predictive power

Third, the machine learning implementation in our study has some practical limitations. We performed careful hyperparameter tuning for the ML models. However, we did this for the first window in the rolling window evaluation process. This means that as the windows progressed further to the end of the sample, the optimal values of the hyper-parameters could be not applicable anymore. Additionally, due to computational restraints, we had to limit the extensiveness of the parameter grids in the grid search, possibly not including the optimal settings for any hyperparameter. Even worse, since the computational costs for deep learning models are even higher, we did not perform hyperparameter estimation at all. For instance, our neural networks had certain fixed sizes, and we did not try an extensive grid of layer depths or neuron counts beyond our manual specification. We also note that we trained each model per stock individually, not pooling data across stocks. This aggravated the already existent problem that 1277 train- and 521 test observations simply do not meet the huge data requirements of ML methods to discover underlying patterns in the realized kernel estimator of the volatility. Finally, our predictor set consisted solely of features engineered from the log returns, realized-kernel and market sentiment. We did not incorporate firm-level financial data such as profit and loss statements, extra technical indicators or additional macro variables. Since all have been shown in other studies to enhance ML performance, our comparison likely understates what a more optimized feature-rich ML pipeline could achieve.

To address these limitations, future research should consider expanding the cross-section to include securities with different volatility dynamics. This would increase the generalisability of the result beyond one specific industry. Furthermore, alongside pooling data across firms, incorporating firm-level financial data, additional technical indicators, and macroeconomic variables may further enhance the performance of ML and deep-learning methods in volatility forecasting as this may mitigate data-scarcity constraints and enable more complex architectures such as Transformers and neural networks to really deploy their pattern recognition capabilities.

Future research could include using sentiment derived regressors in realized specifications of the GARCH, alongside or replacing the realized kernel variance estimator. On the deep learning side, researchers could employ different kinds of Transformers, such as the Informer of Wu et al. [2020], FedFormer of Yang et al. [2022], and selective state space models with Mamba blocks of Gu and Dao [2024] which are designed to alleviate architectural and computational bottlenecks of the vanilla Transform in time series forecasting applications.

Conclusion

This study set out to evaluate whether augmenting modern machine- and deep-learning techniques with stock-specific daily sentiment data could improve daily volatility forecasts for leading technology stocks. Our study emphasizes, when compared fairly, especially using univariate data, that machine learning has not yet proven to yield game-changing performance upgrades in volatility forecasting when compared against well-specified traditional models such as the realized GARCH and stochastic volatility frameworks. In an era of a revolution by AI, driven by increasingly complicated models, it is tempting to believe that more complexity automatically translates to better forecasts. However, our results show that without the solid grounding of economic theory and the ability to capture market effects, even the most advanced methods like the Transformer may fall short. Blending AI-driven techniques with established econometric frameworks offers a clear path forward, combining power with interpretability to build volatility models finance professionals can rely upon.

Appendices

A Constrained Maximum Likelihood via SQPLS

Let the filtered parameter of true $\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)$ and $\mu_t(\boldsymbol{\theta}, \mu_1)$, as defined in (23) for the conditional variance be $\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)$ and $\hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)$. We define the average log-likelihood as

$$L_T\big(\boldsymbol{r_T}, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)\big) \;=\; \frac{1}{T}\sum_{t=2}^{T} \log\, p\big(r_t \mid \mathcal{F}_t, \mu_t(\boldsymbol{\theta}, \mu_1), \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2),\, \boldsymbol{\theta}\big). \tag{A.1}$$

The Maximum Likelihood Estimator is then

$$\hat{\boldsymbol{\theta}}_T \;=\; \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\; L_T\big(\boldsymbol{r_T}, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big). \tag{A.2}$$

We wish to solve the constrained MLE problem

$$\begin{aligned} \hat{\boldsymbol{\theta}}_T = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\; & -L_T\big(\boldsymbol{r}_{1:T}, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2),\, \boldsymbol{\theta}\big) \\ \text{s.t.} \quad & h(\boldsymbol{\theta}) \geq 0, \quad g(\boldsymbol{\theta}) = 0, \end{aligned}$$

where L_T is the average log-likelihood from and $h : \mathbb{R}^p \to \mathbb{R}^{m_I}$, $g : \mathbb{R}^p \to \mathbb{R}^{m_E}$ (m_I and m_E the number of inequality and equality constraints respectively) are differentiable.

We define the (negative-)Lagrangian

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\omega}) = -\,L_T\big(\boldsymbol{r}_{1:T}, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big) \;+\; \boldsymbol{\lambda}^T h(\boldsymbol{\theta}) \;+\; \boldsymbol{\omega}^T g(\boldsymbol{\theta}), \tag{A.3}$$

with multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{m_I}$, $\boldsymbol{\omega} \in \mathbb{R}^{m_E}$.

At iterate $\big(\boldsymbol{\theta}_k, \boldsymbol{\lambda}_k, \boldsymbol{\omega}_k\big)$, we have

$$\begin{aligned} W_k &= \nabla^2_{\theta\theta}\, \mathcal{L}\big(\boldsymbol{\theta}_k, \boldsymbol{\lambda}_k, \boldsymbol{\omega}_k\big), \\ H_k &= \nabla h(\boldsymbol{\theta}_k) \quad \in \mathbb{R}^{m_I \times p}, \\ G_k &= \nabla g(\boldsymbol{\theta}_k) \quad \in \mathbb{R}^{m_E \times p}, \\ \Lambda_k &= \operatorname{diag}\big(\boldsymbol{\lambda}_k\big) \quad \in \mathbb{R}^{m_I \times m_I}, \\ D_k &= \operatorname{diag}\big(h(\boldsymbol{\theta}_k)\big) \quad \in \mathbb{R}^{m_I \times m_I}, \\ \nabla_\theta \mathcal{L}_k &= -\nabla_\theta L_T\big(\boldsymbol{r}_{1:T}, \hat{\sigma}_t^2, \boldsymbol{\theta}_k\big) \;+\; H_k^T\, \boldsymbol{\lambda}_k \;+\; G_k^T\, \boldsymbol{\omega}_k. \end{aligned}$$

This Newton direction $(d_\theta, d_\lambda, d_\omega)$ solves the KKT-linear system

$$\begin{pmatrix} W_k & H_k^T & G_k^T \\ \Lambda_k\, H_k & D_k & 0 \\ G_k & 0 & 0 \end{pmatrix} \begin{pmatrix} d_\theta \\ d_\lambda \\ d_\omega \end{pmatrix} = - \begin{pmatrix} \nabla_\theta \mathcal{L}_k \\ \Lambda_k\, h(\boldsymbol{\theta}_k) \\ g(\boldsymbol{\theta}_k) \end{pmatrix}, \tag{A.4}$$

which is a quadratic–linear system. Kraft [1988] solves it by redefining it as a constrained least-squares problem and applying a QR-based factorization. Once solved we apply update

B Probability Distributions

In this section we define the probability distributions used to specify the distribution of returns. For notational convenience, let $p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big)$

B.1 Normal distribution

The normal (Gaussian) distribution, first developed by Gauss [1809], with mean parameter $\hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)$ and variance $\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2) > 0$ has probability density function

$$p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big) \;=\; \frac{1}{\sqrt{2\pi\,\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} \exp\Big(-\frac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{2\,\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\Big),$$

and hence the natural logarithm of the density is

$$\log p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big) = -\tfrac{1}{2}\log\big(2\pi\,\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)\big) - \frac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{2\,\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}.$$

B.2 Student-t distribution

The Student-t distribution developed by Student [1908] with degrees of freedom parameter $\nu > 2$ has probability density function

$$p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big) = \frac{\Gamma\big(\frac{\nu+1}{2}\big)}{\Gamma\big(\frac{\nu}{2}\big)\,\sqrt{(\nu-2)\,\pi\,\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} \Big(1 + \frac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{(\nu-2)\,\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\Big)^{-\frac{\nu+1}{2}},$$

with natural logarithm

$$\begin{aligned}\log p\big(r_t \mid \mathcal{F}_t, \mu_t(\boldsymbol{\theta}, \mu_1), \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2), \boldsymbol{\theta}\big) = {} & \log\Gamma\big(\tfrac{\nu+1}{2}\big) - \log\Gamma\big(\tfrac{\nu}{2}\big) - \tfrac{1}{2}\log\big((\nu-2)\,\pi\,\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)\big) \\ & - \frac{\nu+1}{2}\log\Big(1 + \frac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{(\nu-2)\,\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\Big).\end{aligned}$$

B.3 Exponential Generalized Beta distribution of the second kind (EGB2)

The Exponential Generalized Beta distribution of the second kind (EGB2; McDonald 1991) with shape parameters $p > 0$ and $q > 0$ has density

$$p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big) = \frac{\sqrt{\Omega}\,\exp\big(p(\sqrt{\Omega}\,\frac{r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} + \Delta)\big)}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\,B(p,q)\,\big(1 + \exp(\sqrt{\Omega}\,\frac{r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} + \Delta)\big)^{p+q}}.$$

Here

$$\Delta = \psi(p) - \psi(q), \qquad \Omega = \psi'(p) + \psi'(q), \qquad B(p,q) = \frac{\Gamma(p)\,\Gamma(q)}{\Gamma(p+q)},$$

and the log-density is

$$\begin{aligned}\log p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big) = {} & \tfrac{1}{2}\log(\Omega) + p\big(\sqrt{\Omega}\,\tfrac{r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} + \Delta\big) \\ & - \tfrac{1}{2}\log(\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)) - \log B(p,q) \\ & - (p+q)\,\log\big(1 + \exp(\sqrt{\Omega}\,\tfrac{r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} + \Delta)\big).\end{aligned}$$

B.4 Asymmetric Student-t (AST) distribution

The Asymmetric Student-t (AST) distribution of Zhu and Galbraith [2010a] with skewness parameter $0 < \alpha < 1$, left-tail $\nu_1 > 2$ and right-tail $\nu_2 > 2$ has density

$$p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big) = \frac{s\,B}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}}\Big[(1 - I_t)\left(1 + \tfrac{1}{\nu_1}\big(m + s\,\tfrac{r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}}\big)^{2\alpha^*}\right)^{-\frac{\nu_1+1}{2}} + I_t\left(1 + \tfrac{1}{\nu_2}\big(m + s\,\tfrac{r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}}\big)^{2(1-\alpha^*)}\right)^{-\frac{\nu_2+1}{2}}\Big],$$

with

$$I_t = \begin{cases} 1, & m + s\,\frac{r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} > 0, \\ 0, & \text{otherwise}, \end{cases} \qquad K(\nu) = \frac{\Gamma\big(\frac{\nu+1}{2}\big)}{\sqrt{\pi\,\nu}\,\Gamma\big(\frac{\nu}{2}\big)}, \quad B = \alpha\,K(\nu_1) + (1-\alpha)\,K(\nu_2),$$

and the log-density writes

$$\begin{aligned}\log p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big) &= \log(s) + \log(B) - \tfrac{1}{2}\log(\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)) \\ &\quad - (1 - I_t)\,\frac{\nu_1 + 1}{2}\,\log\Big(1 + \tfrac{1}{\nu_1}\big(m + s\,\tfrac{r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}}\big)^{2\alpha^*}\Big) \\ &\quad - I_t\,\frac{\nu_2 + 1}{2}\,\log\Big(1 + \tfrac{1}{\nu_2}\big(m + s\,\tfrac{r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}}\big)^{2(1-\alpha^*)}\Big).\end{aligned}$$

with

$$\alpha^* = \frac{\alpha\,K(\nu_1)}{B}, \quad m = 4\,B\Big(-\frac{\alpha^{*2}\,\nu_1}{\nu_1 - 1} + \frac{(1-\alpha^*)^2\,\nu_2}{\nu_2 - 1}\Big), \quad s = \sqrt{4\Big(\frac{\alpha\,\alpha^{*2}\,\nu_1}{\nu_1 - 2} + \frac{(1-\alpha)\,(1-\alpha^*)^2\,\nu_2}{\nu_2 - 2}\Big) - m^2}.$$

B.5 Gamma Distribution

The Gamma kernel distribution its pdf writes

$$f\Big(X_t; \frac{\nu}{2}, \frac{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}{\nu}\Big) = \frac{1}{\Gamma\big(\frac{\nu}{2}\big)}\Big(\frac{\nu}{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\Big)^{\frac{\nu}{2}} X_t^{\frac{\nu}{2}-1}\exp\Big(-\frac{\nu X_t}{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\Big).$$

The log-likelihood function is given by

$$\log f\Big(X_t; \frac{\nu}{2}, \frac{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}{\nu}\Big) = \log\Gamma\Big(\frac{\nu}{2}\Big)^{-1} + \frac{\nu}{2}\log\Big(\frac{\nu}{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\Big) + \Big(\frac{\nu}{2} - 1\Big)\log(X_t) - \frac{\nu X_t}{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}.$$

B.6 Weibull Distribution

The Weibull distribution from Weibull [1939] its PDF is:

$$f\Big(X_t; \frac{\nu}{2}, \frac{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}{\nu}\Big) = \frac{\nu^2}{4\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\left(\frac{\nu X_t}{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\right)^{\frac{\nu}{2}-1}\exp\left(-\left(\frac{\nu X_t}{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\right)^{\frac{\nu}{2}}\right).$$

The log-likelihood function is given by:

$$\log f\Big(X_t; \frac{\nu}{2}, \frac{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}{\nu}\Big) = \log\left(\frac{\nu^2}{4\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\right) + \Big(\frac{\nu}{2} - 1\Big)\log\left(\frac{\nu X_t}{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\right) - \left(\frac{\nu X_t}{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\right)^{\frac{\nu}{2}},$$

C Model Specifics

C.1 The AR(1) model

One-step-ahead forecasts are computed recursively as:

$$\widehat{RK}_{T+1|T} = \hat{\mu}_T(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}) = \hat{\mu} + \hat{\phi}\, RK_T \tag{C.1}$$

The general h-step-ahead forecast, conditional on information at time T, is:

$$\hat{RK}_{T+h|T} = \hat{\mu} \sum_{j=0}^{h-1} \hat{\phi}^j + \hat{\phi}^h RK_T = \hat{\mu} \cdot \frac{1-\hat{\phi}^h}{1-\hat{\phi}} + \hat{\phi}^h RK_T, \quad \text{for } |\hat{\phi}| < 1. \tag{C.2}$$

C.2 The HAR model

One-step-ahead forecasts are computed as

$$\widehat{RK}_{T+1|T} = \hat{\mu_T}(\hat{\boldsymbol{\theta}}_T) = \hat{\phi}_0 + \hat{\phi}_1 RK_T^{(d)} + \hat{\phi}_2 RK_T^{(w)} + \hat{\phi}_3 RK_T^{(m)}. \tag{C.3}$$

Multi-step-ahead forecasts are computed recursively by updating the lagged averages with the previous forecasts

$$\widehat{RK}_{T+h|T} = \hat{\mu}_{T+h|T}(\hat{\boldsymbol{\theta}}_T) = \hat{\phi}_0 + \hat{\phi}_1 \widehat{RK}_{T+h-1}^{(d)} + \hat{\phi}_2 \widehat{RK}_{T+h-1}^{(w)} + \hat{\phi}_3 \widehat{RK}_{T+h-1}^{(m)}. \tag{C.4}$$

C.2.1 The Log HAR model

The h step away forecasts for $\log RK_{T+h|T}$ are made in the same way as in (C.3) and (C.4). Exponentiating both sides yields the forecast

$$\widehat{RK}_{T+h|T} = \exp\left(\hat{\mu}_{T+h|T}(\hat{\boldsymbol{\theta}}_T)\right) = \exp\left(\widehat{\log RK}_{T+h|T}\right). \tag{C.5}$$

C.3 The GARCH Model

Forecasts from a GARCH(1,1) model are directly computed on the conditional variance. For example, the one-step-ahead forecast is simply:

$$\widehat{\sigma}^2_{T+1|T}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2) = \hat{\omega} + \hat{\alpha} r_T^2 + \hat{\beta}\hat{\sigma}_T^2, \tag{C.6}$$

This recursion implies that, for $h \geq 2$, the conditional variance forecasts follow a geometric sequence

$$\widehat{\sigma}^2_{T+h|T}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2) = \hat{\omega} + (\hat{\alpha} + \hat{\beta})\widehat{\sigma}^2_{T+h-1|T}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2), \tag{C.7}$$

C.4 The GARCH ML Model

Let $L_T\big(\boldsymbol{r}_T, \mu_t(\boldsymbol{\theta}, \mu_1), \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2), \boldsymbol{\theta}\big)$ denote the total log-likelihood. To avoid parameters being not identifiable, we estimate the following penalized log-likelihood

$$L_T\left(r_T, \sigma_T(\boldsymbol{\theta^{(m)}}), \boldsymbol{\theta^{(d)}}\right) - 0.01\gamma^2 \tag{C.8}$$

The one-step-ahead forecast is:

$$\hat{\sigma}^2_{T+1|T}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2) = \hat{\omega} + \Big(\hat{\alpha} + \hat{\delta}\tanh(-\hat{\gamma} r_T)\Big)\left(\frac{r_T - \hat{\mu} - \hat{\lambda}\hat{\sigma}_T^2(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2)}{\hat{\sigma}_T(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1)}\right)^2 + \hat{\beta}\hat{\sigma}_T^2(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2). \tag{C.9}$$

For horizons $h > 1$, the process becomes more complicated. Petropoulos et al. [2023] explains that since the GARCH-M-L model involves a non-linear recursion and asymmetric feedback terms, analytical expressions for multi-step-ahead forecasts of conditional variance are not available. Instead, we use simulation-based forecasting.
Let $\hat{\boldsymbol{\theta}}_{\boldsymbol{T}} = (\hat{\mu}, \hat{\omega}, \hat{\alpha}, \hat{\beta}, \hat{\delta}, \hat{\gamma}, \hat{\boldsymbol{\theta}}^{(\boldsymbol{d})}_{\boldsymbol{T}})$ denote the estimated parameters where $\hat{\boldsymbol{\theta}}^{(\boldsymbol{d})}_{\boldsymbol{MLE}}$ denote the distribution specific estimates. Given the last observed return r_T and conditional variance $\hat{\sigma}_T^2$, we simulate S paths over a forecast horizon h. For each simulation $s = 1, \ldots, S$, and step $j = 1, \ldots, h$

$$\varepsilon^{(s)}_{T+j} \sim \mathcal{D}\big(0, 1, \hat{\boldsymbol{\theta}}^{(d)}_{\text{MLE}}\big). \tag{C.10a}$$

$$\begin{aligned}\hat{\sigma}^{2,(s)}_{T+j}\big(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2\big) = \hat{\omega} + \big(\hat{\alpha} + \hat{\delta}\,\tanh(-\hat{\gamma}\, r^{(s)}_{T+j-1})\big)\left(\frac{r^{(s)}_{T+j-1} - \hat{\mu} - \hat{\lambda}\,\hat{\sigma}^{2,(s)}_{T+j-1}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2)}{\hat{\sigma}^{(s)}_{T+j-1}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma_1})}\right)^2 \\ + \hat{\beta}\,\hat{\sigma}^{2,(s)}_{T+j-1}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2).\end{aligned} \tag{C.10b}$$

Then, the h step away forecasts writes

$$\mathbb{E}\left[\hat{\sigma}^2_{T+h}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2) \mid \mathcal{F}_T\right] \approx \frac{1}{S}\sum_{s=1}^{S} \hat{\sigma}^{2,(s)}_{T+j}\big(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}_1^2\big). \tag{C.11}$$

C.5 The EGARCH Model

When we have $\varepsilon_t \sim \mathcal{N}(0, 1)$ we use the standard result

$$\mathbb{E}\big[|\varepsilon_t|\big] = \int_{-\infty}^{\infty} |\varepsilon| \frac{1}{\sqrt{2\pi}} e^{-\varepsilon^2/2}\, dz = \sqrt{\frac{2}{\pi}} \tag{C.12}$$

where a similar result for the Student-t distribution yields

$$\mathbb{E}\big[|\varepsilon_t|\big] = \sqrt{\frac{\nu-2}{\nu}} \int_{-\infty}^{\infty} |\varepsilon| \frac{\Gamma\big(\frac{\nu+1}{2}\big)}{\sqrt{\nu\pi}\,\Gamma\big(\frac{\nu}{2}\big)} \Big(1 + \tfrac{\varepsilon^2}{\nu}\Big)^{-\frac{\nu+1}{2}}\, dx = \sqrt{\frac{\nu-2}{\pi}} \frac{\Gamma\big(\frac{\nu-1}{2}\big)}{\Gamma\big(\frac{\nu}{2}\big)}, \quad \nu > 2, \tag{C.13}$$

and when we assume an EGB2 distribution for ε_t we define $\mathbb{E}|\varepsilon_t| = 0$.

The one-step-ahead forecast of the conditional variance is obtained via

$$\log\big(\hat{\sigma}^2_{T+1}(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1^2)\big) = \hat{\omega} + \hat{\beta}\log(\hat{\sigma}_T^2) + \hat{\alpha}\left(\left|\frac{r_T}{\hat{\sigma}_T}\right| - \mathbb{E}|z_t|\right) + \hat{\gamma}\frac{r_T}{\hat{\sigma}_T(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1)}, \tag{C.14a}$$

$$\hat{\sigma}^2_{T+1}(\hat{\boldsymbol{\theta}_T}, \hat{\sigma}^2_1) = \exp\Big(\log\big(\hat{\sigma}^2_{T+1}(\hat{\boldsymbol{\theta}_T}, \hat{\sigma}^2_1)\big)\Big). \tag{C.14b}$$

For forecast horizons $h > 1$ we adopt simulation-based forecasting

$$\varepsilon^{(s)}_{T+j} \sim \mathcal{D}(0, 1, \hat{\boldsymbol{\theta}}^{(\boldsymbol{d})}_{\boldsymbol{MLE}}) \tag{C.15a}$$

$$\log\Big(\hat{\sigma}^{2,(s)}_{T+j}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}^2_1)\Big) = \hat{\omega} + \hat{\beta}\log\Big(\hat{\sigma}^{2,(s)}_{T+j-1}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}^2_1)\Big) + \hat{\alpha}\left(\Big|\varepsilon^{(s)}_{T+j-1}\Big| - \mathbb{E}|\varepsilon^{(s)}_{T+j-1}|\right) + \hat{\gamma}\varepsilon^{(s)}_{T+j-1}, \tag{C.15b}$$

$$\hat{\sigma}^{2,(s)}_{T+j}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}^2_1) = \exp\left(\log\left(\hat{\sigma}^{2,(s)}_{T+j}(\hat{\boldsymbol{\theta}}_T, \hat{\sigma}^2_1)\right)\right), \tag{C.15c}$$

C.6 The GARCH-MIDAS Model

To model the long-run component $\tau_t(\boldsymbol{\theta}, \boldsymbol{x}_t, \tau_1)$, we use CPI inflation data in the U.S with a monthly frequency, obtained from Federal Reserve Bank of St. Louis [2025]. We define an observation as x_s observed at dates $\{\tau_s\}$. We locate for each daily return date d_t the most-recent index s of the low-frequency CPI inflation series whose date does not exceed d_t. That is,

$$j(t) \;=\; \max\{\, s : \tau_s < d_t\}, \tag{C.16}$$

such that $\boldsymbol{x}_{j(t)}$ is the most recent CPI observation available strictly before day d_t. This allows us to form a window of the past K observations of the CPI. We define

$$\mathbf{x}_t = \big(x_{j(t)-K}, \ldots, x_{j(t)}\big)' \;\in\; \mathbb{R}^K. \tag{C.17}$$

as a K vector of lagged CPI values.

For the one-step-ahead forecast at time $T+1$, the conditional variance is simply given by

$$\hat{\sigma}^2_{T+1|T}(\hat{\boldsymbol{\theta}_T}, \hat{g}_1, \hat{\tau}_1) = \hat{\tau}_{T+1|T}(\hat{\boldsymbol{\theta}_T}, \hat{\tau}_1)\hat{g}_{T+1|T}(\hat{\boldsymbol{\theta}_T}, \hat{g}_1), \tag{C.18}$$

For multi-step horizons $h \geq 2$ no new inflation data arrives. Therefore, we hold the long-term component $\hat{\tau}_{T+h|T}(\hat{\boldsymbol{\theta}_T}, \hat{\tau}_1) = \hat{\tau}_{T+1|T}(\hat{\boldsymbol{\theta}_T}, \hat{\tau}_1)$ constant and propagate the short-run component using the standard GARCH recursion in (C.7) such that the h-step variance forecast becomes

$$\hat{\sigma}^2_{T+h|T}(\hat{\boldsymbol{\theta}_T}, \hat{g}_1, \hat{\tau}_1) \;=\; \hat{\tau}_{T+1|T}(\hat{\boldsymbol{\theta}_T}, \hat{\tau}_1)\hat{g}_{T+h|T}(\hat{\boldsymbol{\theta}_T}, \hat{g}_1), \tag{C.19}$$

C.7 Realized GARCH Models

For Realized GARCH models, the additional measurement equation assumes a normal distribution for RK_t such that we have an extra error $u_t \sim \mathcal{N}(0, \sigma^2_u)$. Hence, the MLE maximization problem defined in (A.2) contains an extra term, which corresponds to the log-likelihood of a normal distribution. The maximization problem is given by

$$\hat{\boldsymbol{\theta}_T} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{1}{T}\left(\sum_{t=2}^{T} \log\, p\big(r_t \mid \mathcal{F}_t, \mu_t(\boldsymbol{\theta}, \mu_1), \sigma^2_t(\boldsymbol{\theta}, \sigma^2_1),\, \boldsymbol{\theta}\big) - \sum_{t=2}^{T} \frac{1}{2}\Big[\log(2\pi) + \log(\sigma^2_u) + \frac{u^2_t}{\sigma^2_u}\Big]\right) \tag{C.20}$$

C.7.1 Linear Realized GARCH

Hansen and Lunde [2011] illustrate that the Realized GARCH model implies a VARMA(1,1) structure for the vector $(h_t,\, RM_t)$ when we substitute the GARCH equation into the measurement equation. This can be exploited for efficient multi-step forecasting. We can write

$$\mathbf{y}_t = \begin{pmatrix} \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2) \\ RK_t \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} \omega \\ \xi + \varphi\,\omega \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} \beta & \gamma \\ \varphi\,\beta & \varphi\,\gamma \end{pmatrix}, \quad \mathbf{e}_t = \begin{pmatrix} 0 \\ w_t \end{pmatrix}. \tag{C.21}$$

as the system

$$\mathbf{y}_t = \mathbf{c} + \mathbf{A}\,\mathbf{y}_{t-1} + \mathbf{e}_t. \tag{C.22}$$

Hence, we can define the h–step-ahead forecast as

$$\mathbf{y}_{T+h|T} = \boldsymbol{A}^h\,\mathbf{y}_T + \sum_{j=0}^{h-1} \boldsymbol{A}^j\,\boldsymbol{c}, \tag{C.23}$$

where for $j \geq 1$ it is possible to show $\boldsymbol{A}^j = (\beta + \varphi\gamma)^{j-1}\boldsymbol{A}$. Since we are only interested in the future path of $\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2)$ is of interest, the model admits the reduced-form AR(1)

$$\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2) = (\omega + \gamma\,\xi) + (\beta + \varphi\,\gamma)\,\sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2) + \gamma\,w_{t-1}, \tag{C.24}$$

such that

$$\hat{\sigma}_{T+h|T}^2(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1^2) = (\hat{\beta} + \hat{\varphi}\hat{\gamma})^h\,\hat{\sigma}_T^2(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1^2) + \sum_{j=0}^{h-1} (\hat{\beta} + \hat{\varphi}\hat{\gamma})^j(\hat{\omega} + \hat{\gamma}\,\hat{\xi}). \tag{C.25}$$

C.7.2 Log-Linear Realized GARCH

Hansen and Lunde [2011] note that the multi-step forecasts can be obtained similarly as for the linear realized specification by defining a VARMA(1,1) for $\tilde{\boldsymbol{y}}_{\boldsymbol{t}} = (\log\sigma_t^2, \log RK_t)'$ and take the exponent

$$\boldsymbol{y}_{\boldsymbol{T+h}} = \exp\left(\boldsymbol{A}^h\tilde{\boldsymbol{y}}_{\boldsymbol{T}} + \sum_{j=0}^{h-1} \boldsymbol{A}^j\boldsymbol{c}\right) \tag{C.26}$$

C.7.3 Realized EGARCH

Hansen and Lunde [2011] rewrite the Realized EGARCH in its VAR(1) companion form using

$$\hat{y}_t = \begin{pmatrix} \log\sigma_t^2(\boldsymbol{\theta}, \sigma_1^2) \\ \log RK_t \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} \omega \\ \xi \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} \beta & 0 \\ 1 & 0 \end{pmatrix}, \quad \boldsymbol{\varepsilon}_t = \begin{pmatrix} \tau(\varepsilon_{t-1}) + \delta\,u_{t-1} \\ u_t \end{pmatrix}, \tag{C.27}$$

and applying the same procedure as in section C.7.1. Because $\mathbb{E}[\boldsymbol{\varepsilon}_t] = 0$, the h–step forecast is the same as in (C.26).

C.8 GAS models

We derive the score function for the Normal Distribution. We define the PDF of the normal distribution using $\log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)$

$$p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2),\, \boldsymbol{\theta}\big) = \frac{1}{\sqrt{2\pi\, e^{\log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}}} \exp\Big(-\frac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{2\, e^{\log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}}\Big) \tag{C.28}$$

Defining $\log p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2),\, \boldsymbol{\theta}\big)$ and taking the derivative yields

$$\log p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2),\, \boldsymbol{\theta}\big) = -\tfrac{1}{2}\log(2\pi) - \tfrac{1}{2}\,\log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2) - \frac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{2\, e^{\log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}}. \tag{C.29}$$

$$\begin{aligned}
\nabla_t &= \frac{\partial}{\partial \log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\, \log p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2),\, \boldsymbol{\theta}\big) && \text{(C.30)}\\
&= -\tfrac{1}{2} + \frac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{2\, e^{\log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} && \text{(C.31)}\\
&= -\frac{1}{2} + \frac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{2\, \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\,. && \text{(C.32)}
\end{aligned}$$

We can follow a similar approach when we have a Student-t distribution

$$\begin{aligned}
\log p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2),\, \boldsymbol{\theta}\big) &= \log \Gamma\Big(\tfrac{\nu+1}{2}\Big) - \log \Gamma\Big(\tfrac{\nu}{2}\Big) - \tfrac{1}{2}\log\big((\nu-2)\pi\big) && \text{(C.33a)}\\
&\quad - \tfrac{1}{2}\,\log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2) - \tfrac{\nu+1}{2}\,\log\Big(1 + \tfrac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{(\nu-2)\, e^{\log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}}\Big), && \text{(C.33b)}
\end{aligned}$$

$$\begin{aligned}
\nabla_t &= \frac{\partial}{\partial \log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}\, \log p\big(r_t \mid \mathcal{F}_t, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2),\, \boldsymbol{\theta}\big) && \text{(C.34a)}\\
&= -\tfrac{1}{2} + \frac{\nu+1}{2}\, \frac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{(\nu-2)\, e^{\log \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)} + (r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}\,. && \text{(C.34b)}
\end{aligned}$$

For the AST distribution we derive

$$\nabla_t = -\frac{1}{2} + I_t \frac{v_2+1}{2} \left(\frac{s(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)) \frac{s(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} + m}{4(1-a^*)^2 v_2 \left(\frac{\frac{s(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} + m)^2}{4(1-a^*)^2 v_2} + 1 \right) \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)^{1/2}} \right) \tag{C.35}$$

$$+ (1 - I_t) \frac{v_2+1}{2} \left(\frac{s(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)) \frac{s(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} + m}{4(a^*)^2 v_1 \left(\frac{(\frac{s(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))}{\sqrt{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} + m)^2}{4(a^*)^2 v_1} + 1 \right) \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)^{1/2}} \right) \tag{C.36}$$

and for the EGB2 distribution we obtain

$$\nabla_t = \frac{\sqrt{\Omega}\,(-q-p)\,(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1)) \exp\left(\frac{\sqrt{\Omega}(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))}{\sqrt{\sigma_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} + \Delta \right)}{2\sqrt{\sigma_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}(\exp\left(\frac{\sqrt{\Omega}(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))}{\sqrt{\sigma_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} + \Delta \right) + 1)} - \frac{1}{2} - \frac{\sqrt{\Omega} p (r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))}{2\sqrt{\sigma_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}} \tag{C.37}$$

The score of the Gamma distribution writes

$$\nabla_t = -\frac{\nu}{2} + \frac{\nu X_t}{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}. \tag{C.38}$$

Finally, the score function of the Weibell distribution is given by

$$\nabla_t = 2 - \frac{\nu}{2} + \frac{\nu^2 2^{\frac{\nu}{2}-1} X_t \left(\frac{\nu X_t}{\hat{\sigma}_t^2(\boldsymbol{\theta},\hat{\sigma}_1^2)}\right)^{\frac{\nu}{2}-1}}{\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)^2} \tag{C.39}$$

When $h = 1$ we utilize the recursion in the model definition of the GARCH in equation (38b). When $h \geq 2$ we need to obtain forecasts using simulation. When the conditional mean $\hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1) = 0$, we have

$$r_{T+h}^{(i)} \sim p\big(r_t \mid \mathcal{F}_T, \hat{\sigma}_{T+h-1}^{2,(i)}(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1^2),\, \hat{\boldsymbol{\theta}}_{\boldsymbol{T}}\big), \tag{C.40a}$$

$$\widehat{s}_{T+h}^{(i)} = \frac{\partial}{\partial \log \hat{\sigma}^2(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1^2)} \log p\big(r_{T+h}^{(i)} \mid, \hat{\sigma}_{T+h-1}^{2,(i)}(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1^2),\, \mathcal{F}_{T+h-1}; \widehat{\boldsymbol{\theta}}_{\mathrm{T}}\big) \tag{C.40b}$$

$$\log \hat{\sigma}_{T+h}^{2,(i)}(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1^2) = \omega + \beta\, \log \hat{\sigma}_{T+h-j}^{2,(i)}(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1^2) + \alpha\, \widehat{s}_{T+h}^{(i)}. \tag{C.40c}$$

Then, the h step away forecasts writes

$$\mathbb{E}[\hat{\sigma}_{T+h}^2(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1^2) \mid \mathcal{F}_T] \approx \frac{1}{S}\sum_{i=1}^{S} \exp\left(\log \hat{\sigma}_{T+h}^{2,(i)}(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \hat{\sigma}_1^2)\right). \tag{C.41}$$

C.9 Realized GAS

Similar as for the Realized GARCH models, the optimization problem in (A.2) changes due to the inclusion of RK_t,assuming it follows some distribution where $\log f\Big(X_t; \frac{\nu}{2}, \frac{2\hat{\sigma}_t^2(\boldsymbol{\theta},\hat{\sigma}_1^2)}{\nu}\Big)$ is the log-density of either the Gamma or Weibell distribution. The optimization problem then writes

$$\hat{\boldsymbol{\theta}_{\boldsymbol{T}}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}};\, L_T\big(\boldsymbol{r}_{\boldsymbol{T}}, \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1), \hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2), \boldsymbol{\theta}\big) + \frac{1}{T}\sum_{t=2}^{T} \log f\Big(X_t; \frac{\nu}{2}, \frac{2\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)}{\nu}\Big) \tag{C.42}$$

where the joint score using the Gamma/Weibell distribution that drives the volatility update writes

$$\log \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2) = \omega + \beta \log \sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2) + \alpha \Big[\underbrace{\tfrac{\nu}{2}\Big(\tfrac{X_{t-1}}{\log \sigma_{t-1}^2(\boldsymbol{\theta},\sigma_1^2)} - 1\Big)}_{\text{Gamma score}} + \underbrace{\nabla_{t-1}}_{\substack{\text{score of}\\ r_t\text{-density}}}\Big], \tag{C.43a}$$

$$\log \sigma_t^2(\boldsymbol{\theta}, \sigma_1^2) = \omega + \beta \log \sigma_{t-1}^2(\boldsymbol{\theta}, \sigma_1^2) + \alpha\Big(\underbrace{2 - \frac{\nu}{2} + \frac{\nu^2 2^{\frac{\nu}{2}-1} X_t \left(\frac{\nu X_t}{\sigma_{t-1}^2(\boldsymbol{\theta},\sigma_1^2)}\right)^{\frac{\nu}{2}-1}}{\sigma_{t-1}^4(\boldsymbol{\theta}, \sigma_1^2)}}_{\text{Weibull score part}} + \nabla_{t-1}\Big). \tag{C.43b}$$

C.10 GAS-MIDAS

If we assume a Normal distribution where we do not set the conditional mean equal to zero, we have already shown in (C.8) that

$$\nabla_t = -\frac{1}{2} + \frac{(r_t - \hat{\mu}_t(\boldsymbol{\theta}, \hat{\mu}_1))^2}{2\,\hat{\sigma}_t^2(\boldsymbol{\theta}, \hat{\sigma}_1^2)} \tag{C.44}$$

with this score function, it is not difficult to see that if $S_t = 1$, $s_t = \nabla_t$ such that the short run component writes

$$g_t(\boldsymbol{\theta}, g_1) \;=\; (1 - \alpha - \beta) \;+\; \alpha\,\frac{(r_{t-1} - \mu)^2}{\tau_{t-1}(\boldsymbol{\theta}, \tau_1)} \;+\; \beta g_{t-1}(\boldsymbol{\theta}, g_1), \tag{C.45}$$

which is exactly the GARCH(1,1) specification as in the GARCH-MIDAS updating equation in (33c). The difference is that here it is a result of assuming normality instead of enforcing the short run component to follow a GARCH specification.

Notice that if we assume a Student-t distribution, we already have derived ∇_t in (C.34). Hence, we can calculate

$$S_t \;= \mathbb{E}[\nabla_t \nabla_t]^{-1} \;=\; -\,\frac{2\,g_t^2\,(\nu + 3)}{\nu}\,. \tag{C.46}$$

such that the full GAS-Midas short term component under the Student-t distribution then writes

$$g_{t+1} = 1 + \alpha\Big(1 + \tfrac{3}{\nu}\Big)\Big[\tfrac{(\nu+1)\,r_t^2}{\nu\,\tau_t\,g_t + r_t^2} - 1\Big]\,g_t + (\alpha + \beta)\,(g_t - 1). \tag{C.47}$$

The h away forecasts are computed in a similar manner as for the GARCH-MIDAS model in (C.19).

C.11 The Kalman filter

We start with the linearized state–space form of the SV model in (46). The Kalman filter in Algorithm 5 produces the filtered estimates $\{\hat{s}_{t|t}\}$ and their prediction-step counterparts $\{\hat{s}_{t|t-1}\}$. To start the recursion, we use a diffuse prior $s_0 \sim \mathcal{N}(a, P1)$ and let $P_0 \to \infty$ such that a is irrelevant. We can use $\{\hat{s}_{t|t}\}$ and $\{\hat{s}_{t|t-1}\}$ to derive the one-step-ahead volatility forecasts

$$\widehat{\sigma}^2_{T+1|T}(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \sigma_1^2) = \exp\big(\hat{s}_{T+1|T}\big). \tag{C.48}$$

The availability of the one-step-ahead forecast in exact form comes directly from all disturbances being Gaussian. This also allows us to use the Gaussian log-likelihood

$$\ell(\psi, \phi, \sigma_\eta) = -\tfrac{1}{2}\sum_{t=1}^{N}\Big(\log F_t + v_t^2/F_t + \log(2\pi)\Big), \tag{C.49}$$

The h-step-ahead forecasts follow by iterating the prediction step of the Kalman filter

$$\hat{s}_{t+h|t} = \psi\sum_{j=0}^{h-1}\phi^j \;+\; \phi^h\,\hat{s}_{t|t}, \tag{C.50a}$$

$$\widehat{\sigma}^2_{T+1|T}(\hat{\boldsymbol{\theta}}_{\boldsymbol{T}}, \sigma_1^2) = \exp\big(\hat{s}_{T+h|T}\big). \tag{C.50b}$$

and the corresponding observation forecast in the transformed scale is

$$\hat{z}_{t+h|t} = -1.27 + \hat{s}_{t+h|t}. \tag{C.51}$$

Algorithm 5 Kalman Filter for Transformed SV Model

Input: Transformed observations $\{z_t\}_{t=1}^N$, QML estimates $\hat{\psi}, \hat{\phi}, \hat{\sigma}_\eta^2$, known $H = \pi^2/2$, prior (s_0, P_0).
Output: Filtered state estimates $\{\hat{s}_{t|t},\, P_{t|t}\}_{t=1}^N$.

Initialization:

$$\hat{s}_{1|0} \leftarrow s_0, \quad P_{1|0} \leftarrow P_0.$$

for $t \leftarrow 1$ **to** N **do**

Predict:

$$\hat{s}_{t|t-1} \leftarrow \hat{\psi} + \hat{\phi}\, \hat{s}_{t-1|t-1}, \tag{C.52}$$
$$P_{t|t-1} \leftarrow \hat{\phi}^2\, P_{t-1|t-1} + \hat{\sigma}_\eta^2, \tag{C.53}$$
$$\hat{z}_{t|t-1} \leftarrow -1.27 + \hat{s}_{t|t-1}. \tag{C.54}$$

Innovation:

$$v_t \leftarrow z_t - \hat{z}_{t|t-1}, \tag{C.55}$$
$$F_t \leftarrow P_{t|t-1} + H. \tag{C.56}$$

Kalman gain:

$$K_t \leftarrow \frac{P_{t|t-1}}{F_t}.$$

Update:

$$\hat{s}_{t|t} \leftarrow \hat{s}_{t|t-1} + K_t\, v_t, \tag{C.57}$$
$$P_{t|t} \leftarrow \big(1 - K_t\big)\, P_{t|t-1}. \tag{C.58}$$

end
return $\{\hat{s}_{t|t},\, P_{t|t}\}_{t=1}^N$

D Machine/Deep Learning

D.0.1 Imputation of Sentiment data

Let $\{MS_t\}_{t=1}^T$ be the observed series with $MS_t \in \{0, 1, 2\} \cup \{NaN\}$. Denote by $\mathcal{O} = \{t\colon MS_t \neq NaN\}$ the set of days for which we do have at least one headline. Next, for each category we define

$$\hat{\pi}_j = \frac{\sum_{t=1}^T \mathbf{1}\{\, MS_t = j\}}{\sum_{t=1}^T \mathbf{1}\{\, MS_t \neq \text{NaN}\}}, \quad j \in \{0, 1, 2\}, \tag{D.1}$$

which is the empirical (marginal) probability of seeing sentiment value j in the observed data. Concretely, it is the proportion of non-missing days on which the sentiment took the value j. If the first observed index is $t_0 > 1$, then for $t = 1, \ldots, t_0 - 1$ we draw

$$MS_t \overset{\text{i.i.d.}}{\sim} \big(\hat{\pi}_0, \hat{\pi}_1, \hat{\pi}_2\big), \tag{D.2}$$

Next, we employ a Markov-Chain style imputation technique based on the transition probability matrix

$$\hat{P}_{ij} = \Pr(MS_t = j \mid MS_{t-1} = i) = \frac{\sum_{t=2}^{T} \mathbf{1}\{\, MS_{t-1} = i,\ MS_t = j\}}{\sum_{t=2}^{T} \mathbf{1}\{\, MS_{t-1} = i\}}, \quad i, j \in \{0, 1, 2\}, \tag{D.3}$$

for $i, j \in \{0, 1, 2\}$. Here the numerator is the number of times the series moves from i to j, and the denominator is the total number of times it leaves state i. More intuitively, each entry represents the probability that the market-sentiment state on day t is j given it was i on day $t - 1$. Now proceeding forward from $t = 2$ to T, any missing MS_t is imputed by sampling

$$MS_t \mid MS_{t-1} = i \ \sim \big(\hat{P}_{i,0},\, \hat{P}_{i,1},\, \hat{P}_{i,2}\big), \quad i \in \{0, 1, 2\}, \tag{D.4}$$

where MS_{t-1} is either observed or has been imputed in the previous step.

D.1 Activation Functions

A key ingredient in modern neural networks is the choice of activation function, which introduces nonlinearity and allows the model to capture complex feature interactions. We use the Rectified Linear Unit (ReLU) activation of Nair and Hinton [2010]

$$(x) = \max(0,\, x). \tag{D.5}$$

Since this is a simple threshold operation it makes it fast to compute than other common but more complex functions like sigmoid or tanh. This simplicity translates to faster training. Furthermore, ReLU helps alleviate the vanishing gradient problem, where gradients can become very small during backpropagation, especially in very deep networks. In addition to ReLU, two other classic activation functions are

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{D.6a}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{D.6b}$$

Here, $\sigma(x)$ is the sigmoid, which squashes inputs into the $(0, 1)$ interval, and $\tanh(x)$ maps to $(-1, 1)$. Both can suffer from vanishing gradients when $|x|$ is large, motivating the use of ReLU in deep networks.

D.2 Dropout Regularization

Dropout regularization is a technique used in neural networks to prevent overfitting by randomly dropping out (ignoring) neurons during training [Srivastava et al., 2014]. This means that during each forward and backward pass, a certain percentage of neurons are temporarily "turned off," preventing them from contributing to the network's learning process. This forces the network to learn more robust and generalized representations, making it less reliant on any single neuron and improving its ability to generalize to unseen data.

D.3 Backpropagation in Feedforward Networks

We start with considering a standard multi-layer feedforward neural network with layers indexed by $l = 0, 1, \ldots, \mathbb{L}$ such that l is the input layer and $\mathbb{L}$ is the last layer .Let $W^l = [w_{ij}^l]$ be the weight matrix connecting layer $l-1$ to layer l and b^l the bias vector for layer l. Furthermore, a_i^l denotes the activation of the i_{th} unit in layer l and $z_i^l = \sum_i w_{ji}^l \, a_i^{l-1} + b_j^l$ is the input of that unit. Thus, $a_i^l = \sigma(z_i^l)$, where $\sigma(\cdot)$ is one of the activation functions in the previous section. Since all functions are differentiable, let σ' denote its derivative.
Backpropagation aims to minimize the Mean Squared Error loss. For a single training example with target output y_k and network output $\hat{y}_k = a_k^L$, the MSE is

$$C(\boldsymbol{\theta}) \;=\; \frac{1}{2}\sum_k (y_k - \hat{y}_k)^2 \,. \tag{D.7}$$

where the factor of $\frac{1}{2}$ is included for convenience since its a simpler derivative. To train the network, we compute the gradients of $C(\boldsymbol{\theta})$ with respect to each weight stacked in $\boldsymbol{\theta}$ using the chain rule, and then adjust weights to minimize $C(\boldsymbol{\theta})$ using gradient descend.

First, we consider a network with one hidden layer and an output layer. Let the hidden layer have indices j and output layer indices k. The weight w_{kj}^2 connects hidden unit j to output unit k, and w_{ji}^1 connects input i to hidden unit j. To train the network, we need to find $\frac{\partial C(\boldsymbol{\theta})}{\partial w_{kj}^2}$. That is, we need to finds the weights that minimize the error. Because $C(\boldsymbol{\theta})$ only depends on the weights indirectly, we need to decompose the derivative using the chain rule

$$\frac{\partial C(\boldsymbol{\theta})}{\partial w_{kj}^2} \;=\; \frac{\partial C(\boldsymbol{\theta})}{\partial a_k^2}\,\frac{\partial a_k^2}{\partial z_k^2}\,\frac{\partial z_k^2}{\partial w_{kj}^2} \,. \tag{D.8}$$

$$\frac{\partial C(\boldsymbol{\theta})}{\partial a_k^2} = (a_k^2 - y_k), \tag{D.9a}$$

$$\frac{\partial a_k^2}{\partial z_k^2} = \sigma'(z_k^2), \tag{D.9b}$$

$$\frac{\partial z_k^2}{\partial w_{kj}^2} = a_j^1, \tag{D.9c}$$

where $a_k^2 \equiv \hat{y}_k$ such that (D.9a) describes how the loss changes if the output activation changes. If a_k^2 is bigger than y_k further increasing it would makes the error bigger so the derivative is positive. If a_k^2 is smalller, the derivative would be negative Next, (D.9b)

shows that the activation a_k^2 changes if the input z_k^2 changes, which is just the slope of the activation function. Lastly, (D.9c) tells us how the input changes if the weights change. Combining these expressions yields

$$\frac{\partial C(\boldsymbol{\theta})}{\partial w_{kj}^2} = (a_k^2 - y_k)\,\sigma'(z_k^2)\,a_j^1, \tag{D.10}$$

where we combine the first two terms in the error signal δ_k

$$\delta_k^2 \equiv (a_k^2 - y_k)\,\sigma'(z_k^2) \quad \Longrightarrow \frac{\partial C(\boldsymbol{\theta})}{\partial w_{kj}^2} = \delta_k^2 a_j^1 \tag{D.11}$$

which tells us how sensitive the error is with respect to z_k^2. Now we define how the cost changes with respect to the weights that connect the input to the hidden layer. Again with the chain rule, we back-propagate through the output layer

$$\frac{\partial C(\boldsymbol{\theta})}{\partial w_{ji}^1} = \sum_k \frac{\partial C(\boldsymbol{\theta})}{\partial a_k^2}\,\frac{\partial a_k^2}{\partial z_k^2}\,\frac{\partial z_k^2}{\partial a_j^1}\,\frac{\partial a_j^1}{\partial z_j^1}\,\frac{\partial z_j^1}{\partial w_{ji}^1} = \sum_k \delta_k^2\, w_{kj}^2\, \sigma'(z_j^1)\, a_i^0, \tag{D.12}$$

where each output neuron k sends back its own error signal δ_ks weighted by the weights connecting the hidden layer to the output layer w_{kj}, by the hidden layer slope of the activation function $\sigma(z_j^1)$ and the input a_i^0. Summing the effects over all outputs k gives us the total change in the outputs for a given change in w_{ji}^1 [Stansbury, 2020, González, 2018]

Next we show how to extend this single–hidden-layer derivation to a network with $l = 1, \ldots, \mathbb{L}$ layers. Let $\boldsymbol{\theta} = (\,\mathrm{vec}(W^1),\, b^1,\, \ldots,\, \mathrm{vec}(W^L),\, b^L)$ denote the vector of network parameters. After computing the output-layer error signals δ^L, we propagate errors backward through all hidden layers as

$$\delta^l = \frac{\partial C(\boldsymbol{\theta})}{\partial \mathbf{z}^l} = \left(W^{\,l+1}\right)^\top \delta^{\,l+1} \circ \sigma'\!\left(\mathbf{z}^l\right). \tag{D.13}$$

Once all $\{\delta^l\}$ are known, the gradients with respect to weights and biases in each layer are collected in into a single vector

$$\Delta C(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial C}{\partial W^1} \\ \frac{\partial C}{\partial b^1} \\ \vdots \\ \frac{\partial C}{\partial W^L} \\ \frac{\partial C}{\partial b^L} \end{pmatrix}, \tag{D.14}$$

with

$$\frac{\partial C(\boldsymbol{\theta})}{\partial W^l} = \delta^l \left(a^{\,l-1}\right)^\top, \qquad \frac{\partial C(\boldsymbol{\theta})}{\partial b^l} = \delta^l, \quad l = 1, \ldots, \mathbb{L}. \tag{D.15}$$

Finally, applying a gradient-descent step with learning rate $\eta > 0$ updates all parameters

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\,\Delta C(\boldsymbol{\theta}) \tag{D.16}$$

D.4 Backpropagation Through Time (BPTT) for Recurrent Networks

In contrast to Feedforward backpropagation, which assumes the network to be static with $\mathbb{L}$ layers, RNNs such as LSTM and GRU networks, "unroll"over time steps. That is, at each time step t, the network receives input x_t and produces output $\hat{y}_t$. When the network is "unrolled", meaning the network has completed a forward pass and stored intermediate values h_t and $\hat{y}_t$, Backpropagation Through Time (BPTT) is the usual backpropagation algorithm applied from time T back to time $t = 1$. More specifically, this means that we consider the hidden states $h_0,\ h_1,\ \ldots,\ h_T$ as the $\mathbb{L}$ layers of a feedforward network. However, the difference is that in RNN's the same weight influences the loss at multiple time steps Cho et al. [2014] The loss over the sequence is

$$C(\boldsymbol{\theta}) = \sum_{t=1}^{T} \frac{1}{2} \left\| \hat{y}_t - y_t \right\|^2. \tag{D.17}$$

The output error signal is defined similarly as in the usual backpropagation algorithm

$$\delta_t^{\hat{y}} = \frac{\partial C(\boldsymbol{\theta})}{\partial z_t^{\hat{y}}} = (\hat{y}_t - y_t)\, \sigma'\big(z_t^{\hat{y}}\big). \tag{D.18}$$

The hidden-state error δ_t^h propagates back both through the output layer and through time . In particular, this means that the hidden state error at time t is determined by the output at time t and the hidden state at time $t+1$, which comes from the regressive nature of the hidden states (h_t influences h_{t+1})

$$\delta_t^h \ = \ \frac{\partial E}{\partial z_t^h} \ = \ \sigma'(z_t^h)\Big(W_{hy}^T\, \delta_t^y \ + \ W_{hh}^T\, \delta_{t+1}^h\Big), \tag{D.19}$$

where $W_{hy}^T\, \delta_t^y$ is the back propagated error from the output and $W_{hh}^T\, \delta_{t+1}^h$ comes from the next time step or "layer". This is exacly where the vanishing gradient problem in RNNs comes from. When T is large, repeated multiplication by W_{hh} can shrink or blow up the gradient. Summing over time, we find

$$\frac{\partial C(\boldsymbol{\theta})}{\partial W_{hy}} = \sum_{t=1}^{T} \delta_t^{\hat{y}}\, h_t^\top, \tag{D.20}$$

To find $\partial E/\partial W_{hh}$ we need to consider that W_{hh} influences $h_{t+1}, h_{t+2} \ldots$ indirectly through h_t. BPPT deals with this by summing up direct and indirect gradient contributions.

$$\frac{\partial C(\boldsymbol{\theta})}{\partial W_{hh}} = \sum_{t=1}^{T} \sum_{i=1}^{t} \left(\frac{\partial C(\boldsymbol{\theta})}{\partial h_t} \frac{\partial h_t}{\partial h_i} \right) \frac{\partial h_i}{\partial W_{hh}} \tag{D.21}$$

For LSTMS and GRU's the backpropagation principles remain the same. However because of the internal gates for each time step, the computations are more involved.

We define

$$\delta_t^h \ = \ \frac{\partial C(\boldsymbol{\theta})}{\partial h_t}, \qquad \delta_t^c \ = \ \frac{\partial C(\boldsymbol{\theta}))}{\partial c_t}\,.$$

Hochreiter and Schmidhuber [1997] state that at each time t the LSTM computes

$$\begin{aligned}
\delta_t^o &= \delta_t^h \odot \tanh(c_t), \\
\delta_t^c &= \delta_t^h \odot o_t \odot \big(1 - \tanh^2(c_t)\big) + \delta_{t+1}^c \odot f_{t+1}, \\
\delta_t^f &= \delta_t^c \odot c_{t-1}, \quad \delta_t^i = \delta_t^c \odot g_t, \quad \delta_t^g = \delta_t^c \odot i_t.
\end{aligned} \tag{D.22}$$

To back propagate through the sigmoid/tanh nonlinearities, the pre-activation errors write

$$\begin{aligned}
\tilde{\delta}_t^f &= \delta_t^f \odot f_t(1 - f_t), \quad \tilde{\delta}_t^i = \delta_t^i \odot i_t(1 - i_t), \\
\tilde{\delta}_t^o &= \delta_t^o \odot o_t(1 - o_t), \quad \tilde{\delta}_t^g = \delta_t^g \odot \big(1 - g_t^2\big).
\end{aligned} \tag{D.23}$$

Finally, the weight gradients accumulate over all T time steps:

$$\begin{aligned}
\frac{\partial C(\boldsymbol{\theta})}{\partial W_{xf}} &= \sum_{t=1}^{T} \tilde{\delta}_t^f \, x_t^\top, \quad \frac{\partial C(\boldsymbol{\theta})}{\partial W_{hf}} = \sum_{t=1}^{T} \tilde{\delta}_t^f \, h_{t-1}^\top, \\
\frac{\partial C(\boldsymbol{\theta})}{\partial W_{xi}} &= \sum_{t=1}^{T} \tilde{\delta}_t^i \, x_t^\top, \quad \frac{\partial C(\boldsymbol{\theta})}{\partial W_{hi}} = \sum_{t=1}^{T} \tilde{\delta}_t^i \, h_{t-1}^\top, \\
\frac{\partial C(\boldsymbol{\theta})}{\partial W_{xo}} &= \sum_{t=1}^{T} \tilde{\delta}_t^o \, x_t^\top, \quad \frac{\partial C(\boldsymbol{\theta})}{\partial W_{ho}} = \sum_{t=1}^{T} \tilde{\delta}_t^o \, h_{t-1}^\top, \\
\frac{\partial C(\boldsymbol{\theta})}{\partial W_{xg}} &= \sum_{t=1}^{T} \tilde{\delta}_t^g \, x_t^\top, \quad \frac{\partial C(\boldsymbol{\theta})}{\partial W_{hg}} = \sum_{t=1}^{T} \tilde{\delta}_t^g \, h_{t-1}^\top.
\end{aligned} \tag{D.24}$$

The process for the GRU is very similar where the LSTM gates are replaced with the GRU's update and reset gates.

E Additional Results

Model	AAPL	MSFT	IBM	ADBE	ASML	AMZN	INTC	TSM	NVDA	AMD	TSLA
EGARCH	5153.79	4858.43	4739.74	5371.10	5717.99	5311.10	5596.02	5389.67	6327.99	6649.76	6999.60
GARCH	5195.96	4889.50	4760.82	5402.31	5752.23	5336.80	5596.73	5384.49	6346.68	6658.59	6996.52
GARCH-AST	5130.78	4836.58	4493.23	5313.96	5727.74	5264.31	5319.10	5355.94	6297.95	6572.98	6850.90
GARCH-EGB2	5150.64	4848.23	4594.90	5338.71	5727.81	5283.39	5423.24	5358.97	6307.33	6599.24	6910.08
GARCH-MIDAS	5168.48	4852.75	4769.83	5364.41	5711.87	5313.13	5551.76	5358.88	6309.63	6601.88	6961.91
GARCH-t	5135.99	4850.68	4496.02	5327.97	5731.65	5264.22	5321.23	5352.91	6300.28	6571.41	6852.37
GARCHMIDAS-AST	7568.70	7258.56	4477.45	7761.05	8113.91	7694.62	7894.92	7759.46	8706.31	9025.11	9387.37
GARCHMIDAS-EGB2	5123.26	4815.29	4582.55	5299.82	**5689.17**	5253.91	5378.72	5331.29	6268.30	6546.77	6872.17
GARCHMIDAS-t	**5106.17**	**4810.16**	**4475.70**	**5280.15**	5689.90	**5230.68**	**5283.33**	**5324.14**	6257.96	6520.84	6813.27
GARCHML	5165.68	4894.01	4793.89	5401.65	5750.19	5367.10	5615.70	5401.48	6358.80	6655.36	7020.53
GARCHML-t	5175.06	4900.56	4779.58	5410.28	5757.06	5376.92	5573.73	5407.51	6367.31	6666.36	7024.80
GASMIDAS	5181.64	4866.92	4776.44	5392.33	5717.22	5333.18	5551.76	5362.53	6327.27	6603.32	6962.19
GASMIDAS-t	5149.83	4855.40	4492.43	5321.14	5733.98	5268.29	5292.57	5369.08	6309.73	6567.88	6867.69
KFSV	5688.50	5830.86	5720.38	5769.22	5694.08	5674.14	5701.65	5724.97	**5518.98**	**5725.44**	**5923.63**
LLRGARCH	6307.94	5881.24	5769.30	6381.95	6974.62	6742.73	6647.08	6354.72	7224.84	7413.37	8264.70
LLRGARCH-EGB2	6269.98	5854.05	5564.74	6285.70	6961.94	6656.48	6455.50	6323.41	7185.71	7338.65	8128.37
LLRGARCH-t	6267.45	5859.86	5557.27	6298.41	6958.67	6651.86	6434.64	6321.30	7184.72	7334.82	8126.05
LRGARCH	10684.13	10113.56	9342.16	10395.72	10247.66	10960.79	11044.50	9284.76	13114.35	13458.02	15399.96
REGARCH	6561.51	6115.06	6007.34	6658.46	7209.98	6929.89	6823.08	6541.40	7433.65	7581.43	8502.30
REGARCH-t	6521.44	6094.58	5799.14	6573.71	7195.31	6843.79	6617.06	6508.12	7399.52	7503.74	8336.98
REGARH-EGB2	6523.39	6094.07	5806.15	6558.01	7198.07	6847.63	6636.70	6510.33	7399.14	7507.56	8344.92

Table E.1: AIC Values for Different Volatility Models Across Selected Stocks (Lowest in Bold)

Model	AAPL	MSFT	IBM	ADBE	ASML	AMZN	INTC	TSM	NVDA	AMD	TSLA
EGARCH	5174.33	4878.97	4760.29	5391.64	5738.54	5331.65	5616.57	5410.22	6348.54	6670.30	7020.15
GARCH	5211.37	4904.91	4776.23	5417.72	5767.64	5352.21	5612.14	5399.90	6362.09	6673.99	7011.93
GARCH-AST	5161.60	4867.40	4524.05	5344.78	5758.56	5295.13	5349.92	5386.76	6328.77	6603.80	6881.72
GARCH-EGB2	5176.32	4873.91	4620.58	5364.40	5753.49	5309.07	5448.92	5384.65	6333.01	6624.92	6935.76
GARCH-MIDAS	5199.25	4883.53	4800.60	5395.19	5742.65	5343.91	5582.53	5389.65	6340.40	6632.65	6992.69
GARCH-t	5156.53	4871.22	4516.57	5348.52	5752.20	5284.77	5341.78	5373.46	6320.82	6591.96	6872.92
GARCHMIDAS-AST	7614.86	7304.72	4517.87	7807.22	8160.07	7740.78	7941.08	7805.63	8752.48	9071.27	9433.53
GARCHMIDAS-EGB2	5164.29	4856.32	4623.58	5340.86	5730.20	5294.95	5419.75	5372.32	6309.34	6587.80	6913.21
GARCHMIDAS-t	**5142.08**	**4846.07**	**4511.61**	**5316.06**	5725.80	**5266.58**	**5319.23**	**5360.05**	6293.86	6556.75	6849.18
GARCHML	5201.63	4929.97	4829.85	5437.61	5786.14	5403.05	5651.65	5437.44	6394.76	6691.31	7056.48
GARCHML-t	5216.15	4941.65	4820.67	5451.37	5798.16	5418.01	5614.82	5448.60	6408.41	6707.45	7065.89
GASMIDAS	5212.41	4897.69	4807.22	5423.11	5748.00	5363.96	5582.53	5393.30	6358.05	6634.10	6992.97
GASMIDAS-t	5185.74	4891.31	4528.33	5357.05	5769.88	5304.20	5328.48	5404.98	6345.63	6603.79	6903.59
KFSV	5703.91	5846.27	5735.78	5784.63	**5709.49**	5689.55	5717.06	5740.38	**5534.39**	**5740.85**	**5939.04**
LLRGARCH	6349.04	5922.34	5810.39	6423.04	7015.71	6783.82	6688.17	6395.81	7265.93	7454.46	8305.80
LLRGARCH-EGB2	6321.34	5905.41	5616.11	6337.06	7013.31	6707.85	6506.86	6374.77	7237.07	7390.01	8179.73
LLRGARCH-t	6313.68	5906.09	5603.49	6344.64	7004.90	6698.09	6480.87	6367.53	7230.95	7381.05	8172.28
LRGARCH	10725.22	10154.66	9383.25	10436.82	10288.75	11001.88	11085.59	9325.85	13155.45	13499.11	15441.06
REGARCH	6597.47	6151.01	6043.29	6694.41	7245.93	6965.85	6859.03	6577.36	7469.61	7617.39	8538.26
REGARCH-t	6562.53	6135.67	5840.23	6614.80	7236.40	6884.88	6658.15	6549.21	7440.61	7544.84	8378.08
REGARH-EGB2	6569.62	6140.30	5852.38	6604.24	7244.30	6893.86	6682.93	6556.56	7445.37	7553.79	8391.15

Table E.2: BIC Values for Different Volatility Models Across Selected Stocks (Lowest in Bold)

Model	AAPL	MSFT	IBM	ADBE	ASML	AMZN	INTC	TSM	NVDA	AMD	TSLA
EGARCH	-2572.89	-2425.21	-2365.87	-2681.55	-2855.00	-2651.55	-2794.01	-2690.84	-3160.00	-3320.88	-3495.80
GARCH	-2594.98	-2441.75	-2377.41	-2698.15	-2873.12	-2665.40	-2795.36	-2689.24	-3170.34	-3326.29	-3495.26
GARCH-AST	-2559.39	-2412.29	-2240.62	-2650.98	-2857.87	-2626.16	-2653.55	-2671.97	-3142.97	-3280.49	-3419.45
GARCH-EGB2	-2570.32	-2419.11	-2292.45	-2664.36	-2858.90	-2636.70	-2706.62	-2674.49	-3148.67	-3294.62	-3450.04
GARCH-MIDAS	-2578.24	-2420.38	-2378.91	-2676.21	-2849.93	-2650.57	-2769.88	-2673.44	-3148.81	-3294.94	-3474.96
GARCH-t	-2563.99	-2421.34	-2244.01	-2659.98	-2861.83	-2628.11	-2656.62	-2672.46	-3146.14	-3281.71	-3422.19
GARCHMIDAS-AST	-3775.35	-3620.28	248.05	-3871.53	-4047.96	-3838.31	-3938.46	-3870.73	-4344.16	-4503.55	-4684.68
GARCHMIDAS-EGB2	-2553.63	-2399.64	-2283.27	-2641.91	**-2836.58**	-2618.96	-2681.36	-2657.64	-3126.15	-3265.38	-3428.09
GARCHMIDAS-t	**-2546.09**	**-2398.08**	**-2230.85**	**-2633.08**	-2837.95	**-2608.34**	**-2634.66**	**-2655.07**	-3121.98	-3253.42	-3399.64
GARCHML	-2575.84	-2440.01	-2389.95	-2693.83	-2868.09	-2676.55	-2800.85	-2693.74	-3172.40	-3320.68	-3503.26
GARCHML-t	-2579.53	-2442.28	-2381.79	-2697.14	-2870.53	-2680.46	-2778.86	-2695.75	-3175.66	-3325.18	-3504.40
GASMIDAS	-2584.82	-2427.46	-2382.22	-2690.16	-2852.61	-2660.59	-2769.88	-2675.26	-3157.64	-3295.66	-3475.10
GASMIDAS-t	-2567.92	-2420.70	-2239.21	-2653.57	-2859.99	-2627.15	-2639.29	-2677.54	-3147.86	-3276.94	-3426.84
KFSV	-2841.25	-2912.43	-2857.19	-2881.61	-2844.04	-2834.07	-2847.82	-2859.48	**-2756.49**	**-2859.72**	**-2958.81**
LLRGARCH	-3145.97	-2932.62	-2876.65	-3182.97	-3479.31	-3363.37	-3315.54	-3169.36	-3604.42	-3698.68	-4124.35
LLRGARCH-EGB2	-3124.99	-2917.02	-2772.37	-3132.85	-3470.97	-3318.24	-3217.75	-3151.70	-3582.85	-3659.32	-4054.18
LLRGARCH-t	-3124.72	-2920.93	-2769.63	-3140.21	-3470.34	-3316.93	-3208.32	-3151.65	-3583.36	-3658.41	-4054.03
LRGARCH	-5334.06	-5048.78	-4663.08	-5189.86	-5115.83	-5472.39	-5514.25	-4634.38	-6549.18	-6721.01	-7691.98
REGARCH	-3273.76	-3050.53	-2996.67	-3322.23	-3597.99	-3457.95	-3404.54	-3263.70	-3709.83	-3783.72	-4244.15
REGARCH-t	-3252.72	-3039.29	-2891.57	-3278.85	-3589.66	-3413.90	-3300.53	-3246.06	-3691.76	-3743.87	-4160.49
REGARH-EGB2	-3252.69	-3038.03	-2894.07	-3270.01	-3590.04	-3414.81	-3309.35	-3246.17	-3690.57	-3744.78	-4163.46

Table E.3: Log-Likelihood Values for Different Volatility Models Across Selected Stocks (Least Negative in Bold)

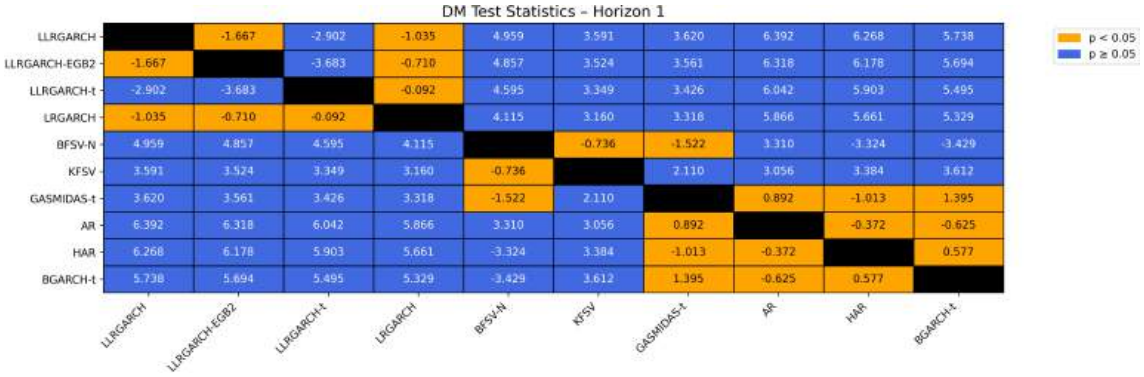


Figure E.1: Comparison matrix of DM test results of the top 10 best performing models at horizon h = 1

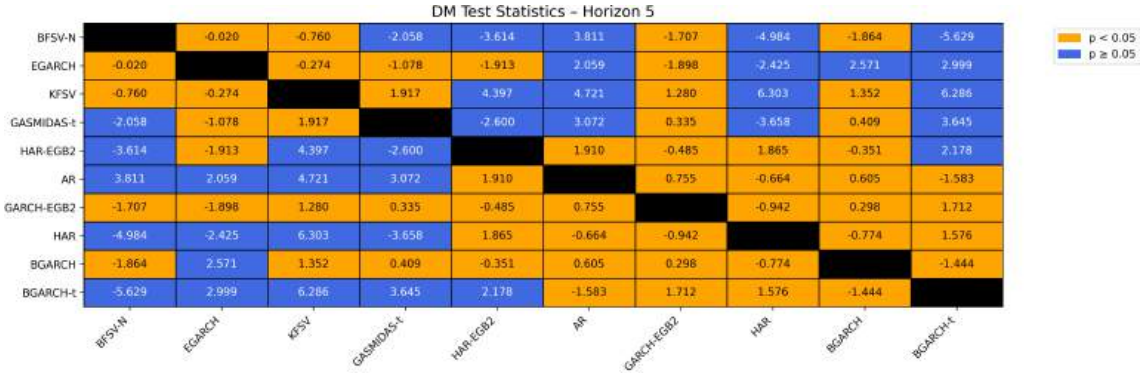


Figure E.2: Comparison matrix of DM test results of the top 10 best performing models at horizon h = 5

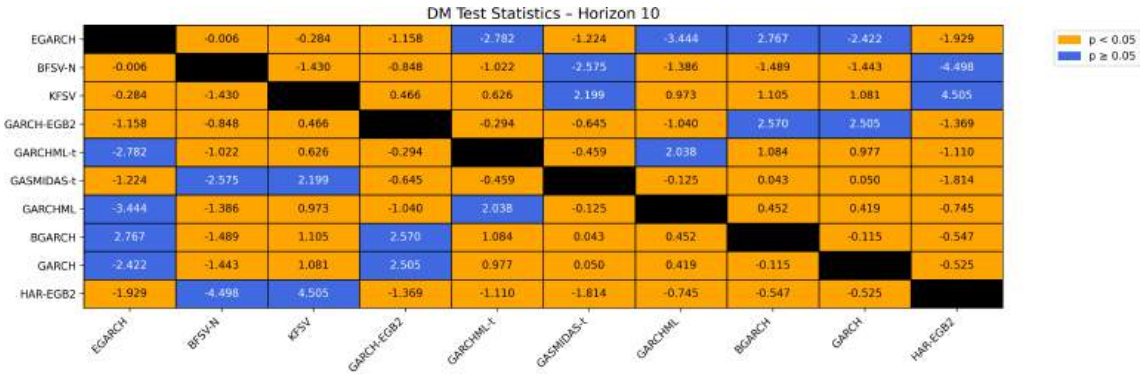


Figure E.3: Comparison matrix of DM test results of the top 10 best performing models at horizon h = 10

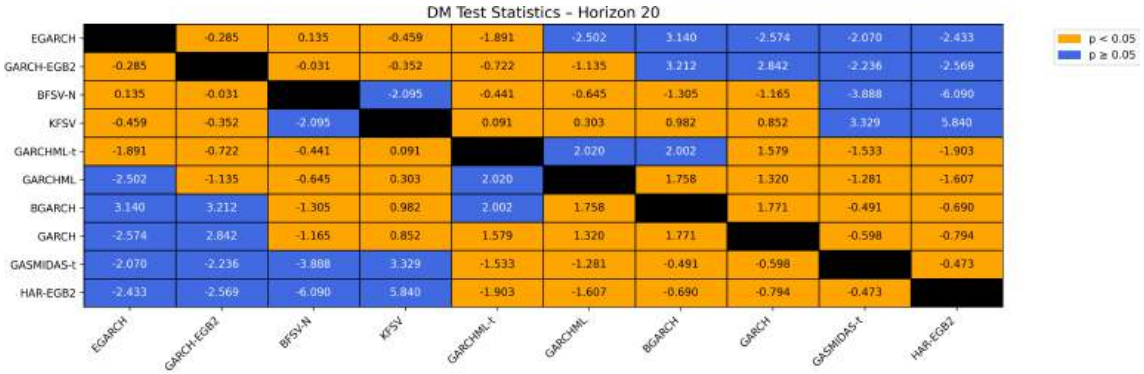


Figure E.4: Comparison matrix of DM test results of the top 10 best performing models at horizon h = 20

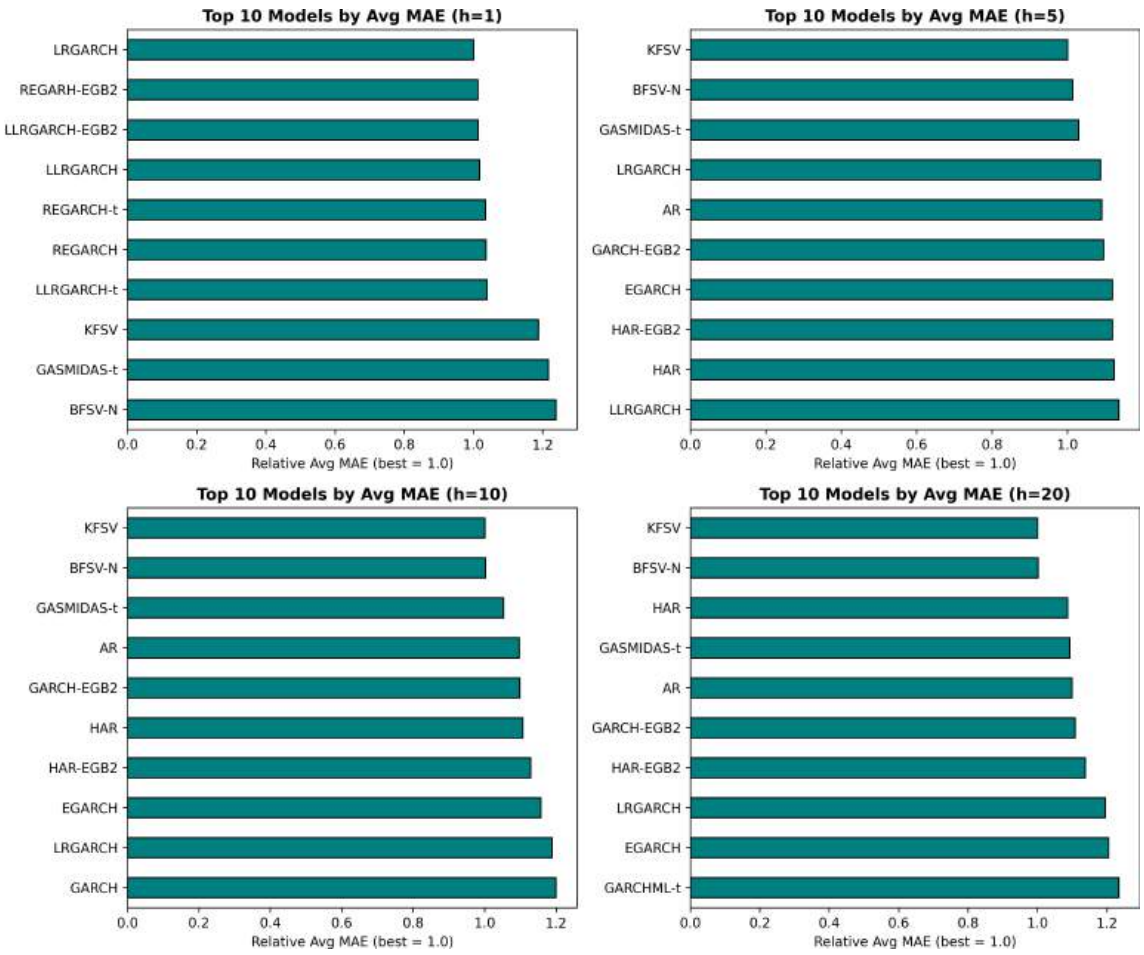


Figure E.5: Bar plot of the relative RMSE across the best performing models where the MAE of the best model is scaled to one.

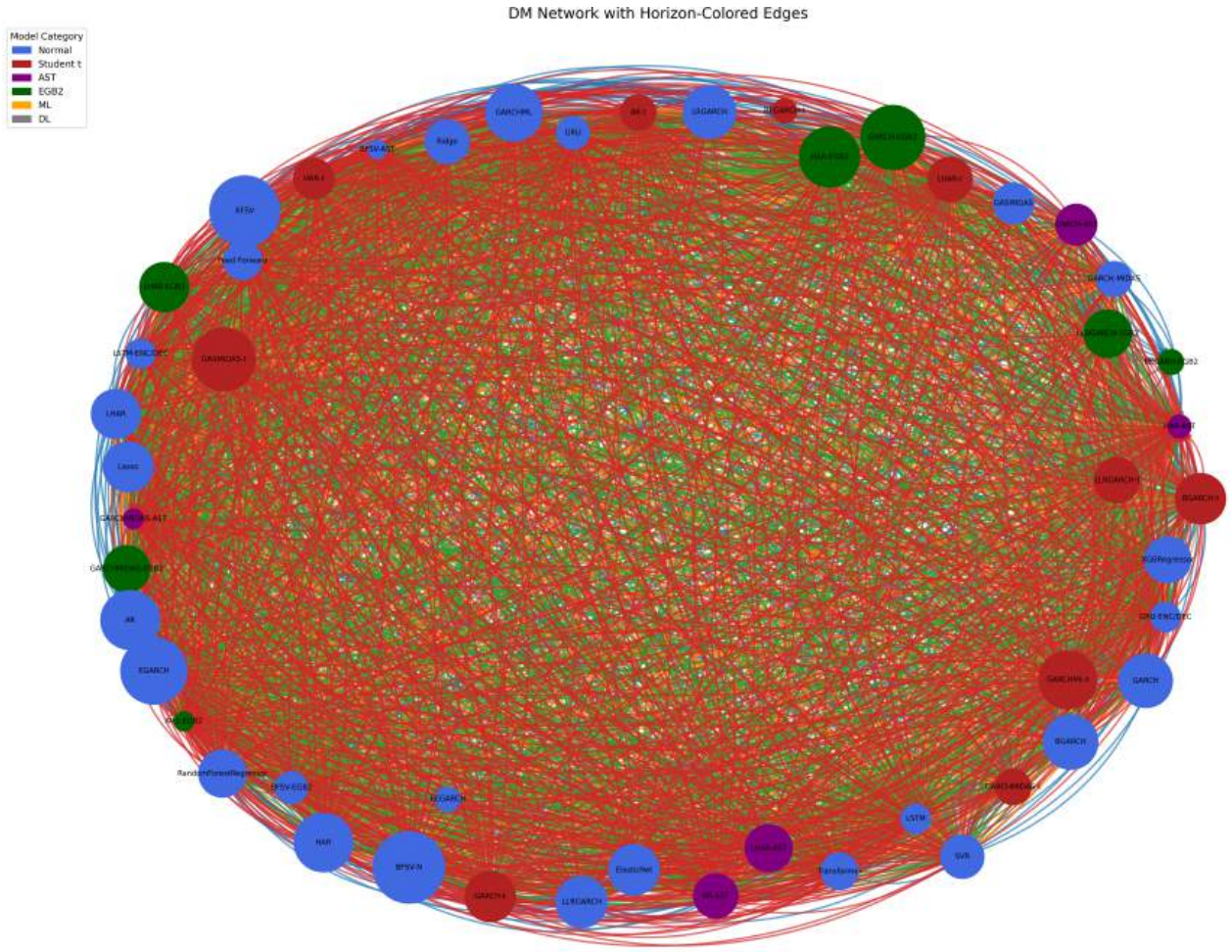


Figure E.6: Network Graph of DM test result. Each model receives an edge from another model if it gets beaten by it at 5% significance. If a model receives more edges it gets bigger, such that bigger nodes are the most powerful forecasting models.

E.1 Apple

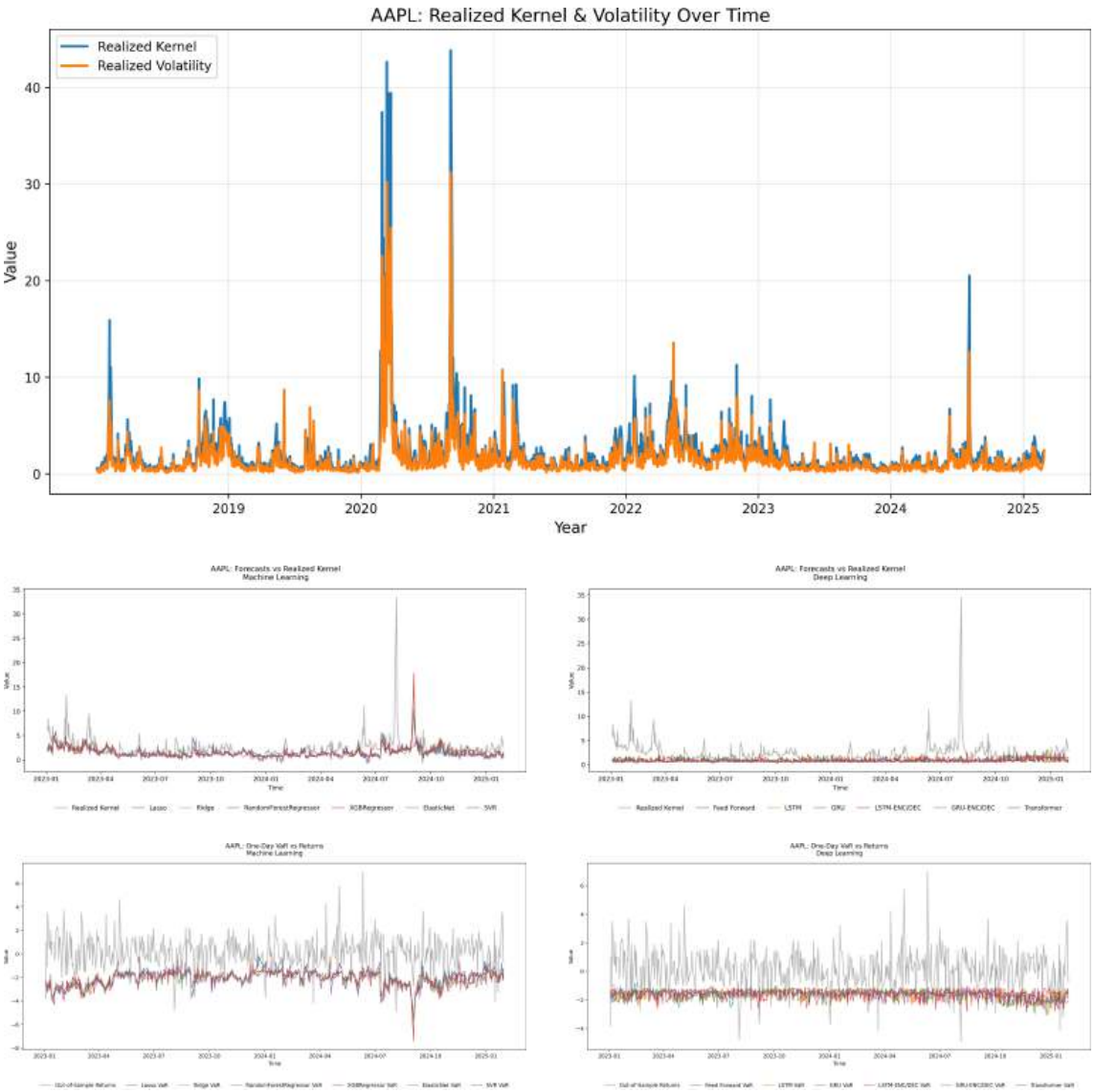
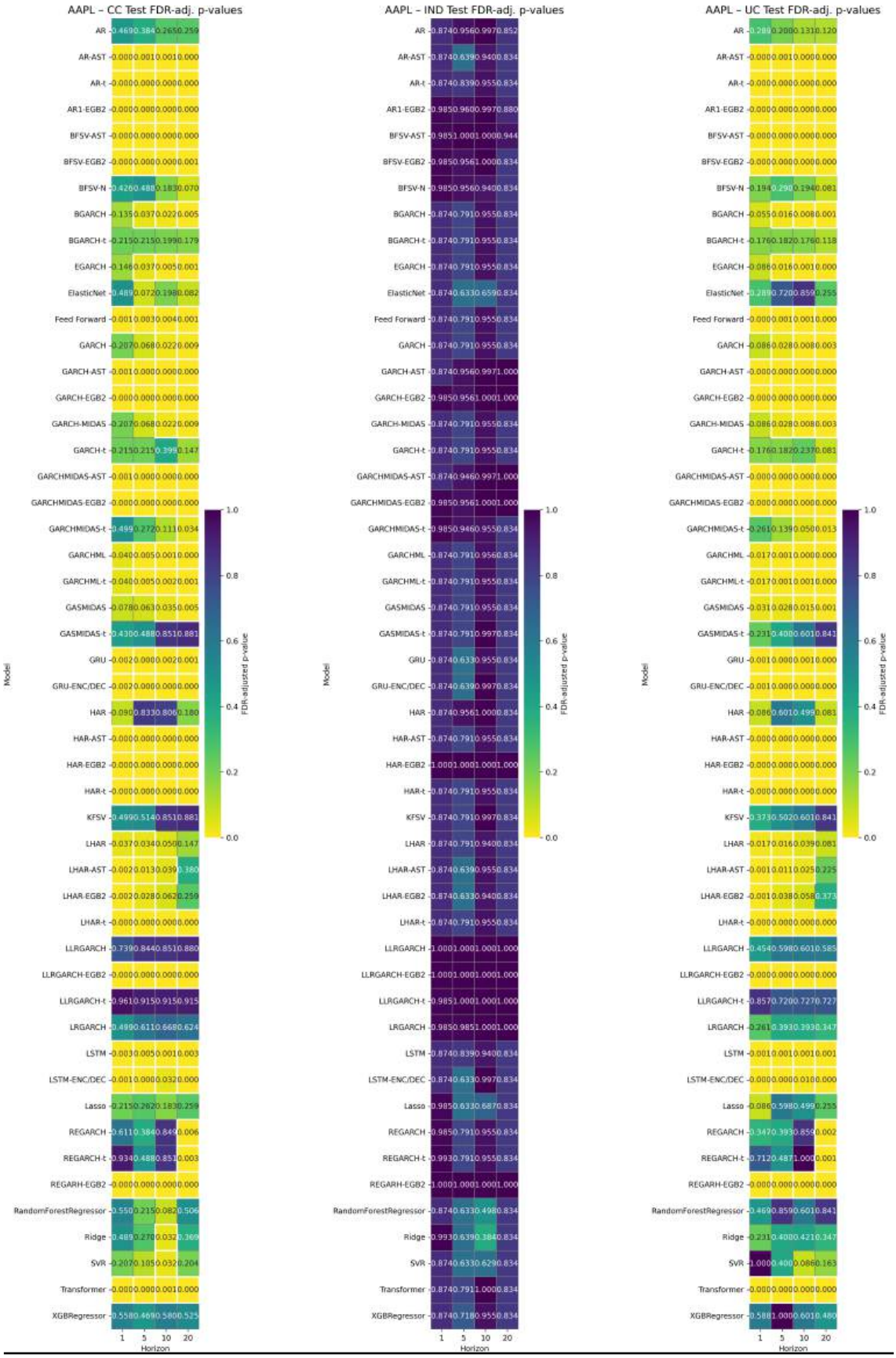
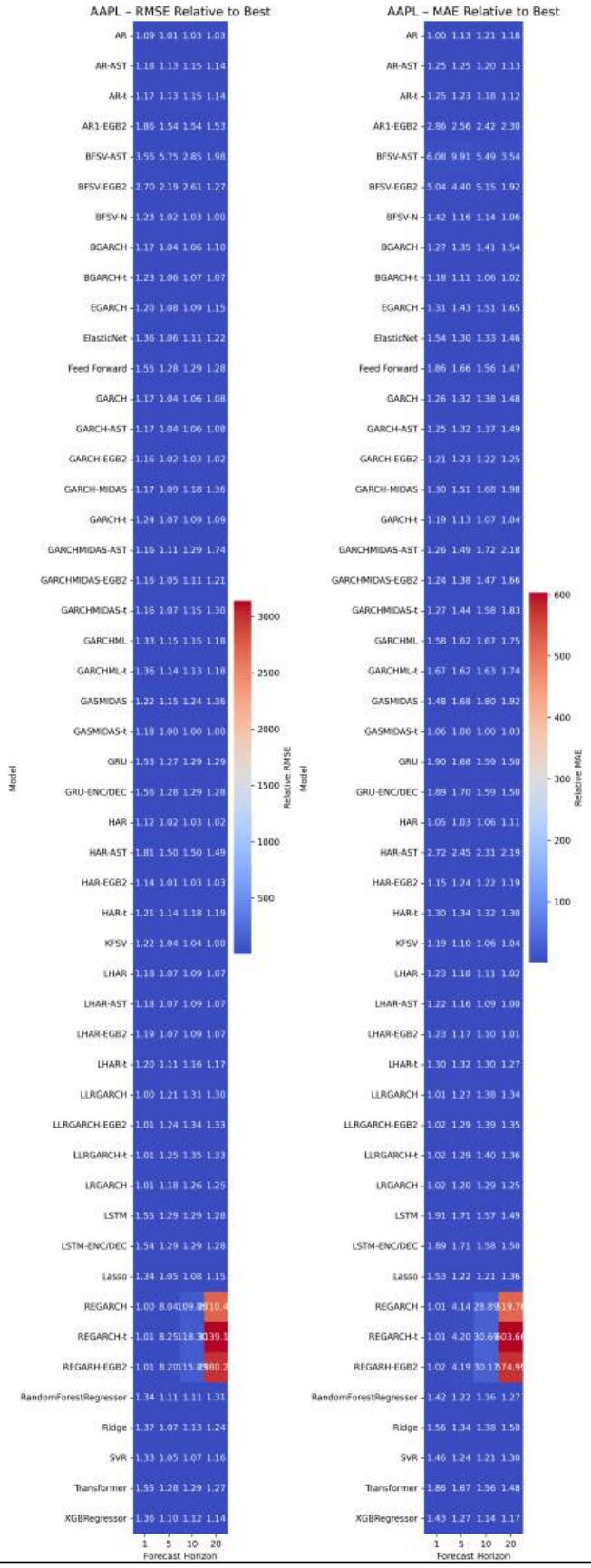


Figure E.7: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for AAPL—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


AAPL – CC Test FDR-adj. p-values
AAPL – IND Test FDR-adj. p-values
AAPL – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value
AR
AR-AST
AR-t
AR1-EGB2
BFSV-AST
BFSV-EGB2
BFSV-N
BGARCH
BGARCH-t
EGARCH
ElasticNet
Feed Forward
GARCH
GARCH-AST
GARCH-EGB2
GARCH-MIDAS
GARCH-t
GARCHMIDAS-AST
GARCHMIDAS-EGB2
GARCHMIDAS-t
GARCHML
GARCHML-t
GASMIDAS
GASMIDAS-t
GRU
GRU-ENC/DEC
HAR
HAR-AST
HAR-EGB2
HAR-t
KFSV
LHAR
LHAR-AST
LHAR-EGB2
LHAR-t
LLRGARCH
LLRGARCH-EGB2
LLRGARCH-t
LRGARCH
LSTM
LSTM-ENC/DEC
Lasso
REGARCH
REGARCH-t
REGARH-EGB2
RandomForestRegressor
Ridge
SVR
Transformer
XGBRegressor


AAPL – RMSE Relative to Best
AAPL – MAE Relative to Best
Model
Relative RMSE
Relative MAE
Forecast Horizon
1
5
10
20
AR
AR-AST
AR-t
AR1-EGB2
BFSV-AST
BFSV-EGB2
BFSV-N
BGARCH
BGARCH-t
EGARCH
ElasticNet
Feed Forward
GARCH
GARCH-AST
GARCH-EGB2
GARCH-MIDAS
GARCH-t
GARCHMIDAS-AST
GARCHMIDAS-EGB2
GARCHMIDAS-t
GARCHML
GARCHML-t
GASMIDAS
GASMIDAS-t
GRU
GRU-ENC/DEC
HAR
HAR-AST
HAR-EGB2
HAR-t
KFSV
LHAR
LHAR-AST
LHAR-EGB2
LHAR-t
LLRGARCH
LLRGARCH-EGB2
LLRGARCH-t
LRGARCH
LSTM
LSTM-ENC/DEC
Lasso
REGARCH
REGARCH-t
REGARH-EGB2
RandomForestRegressor
Ridge
SVR
Transformer
XGBRegressor

E.2 Adobe

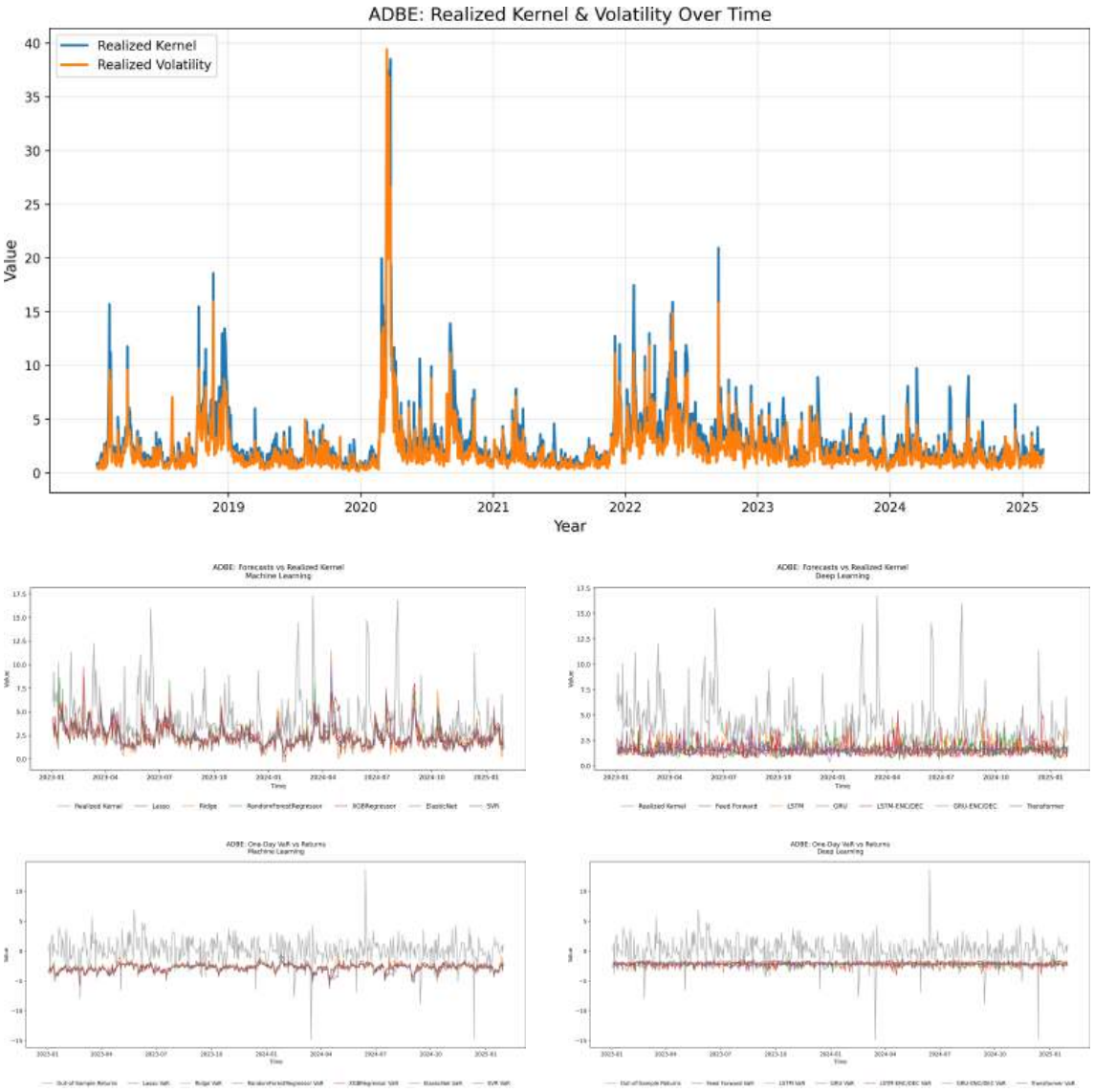
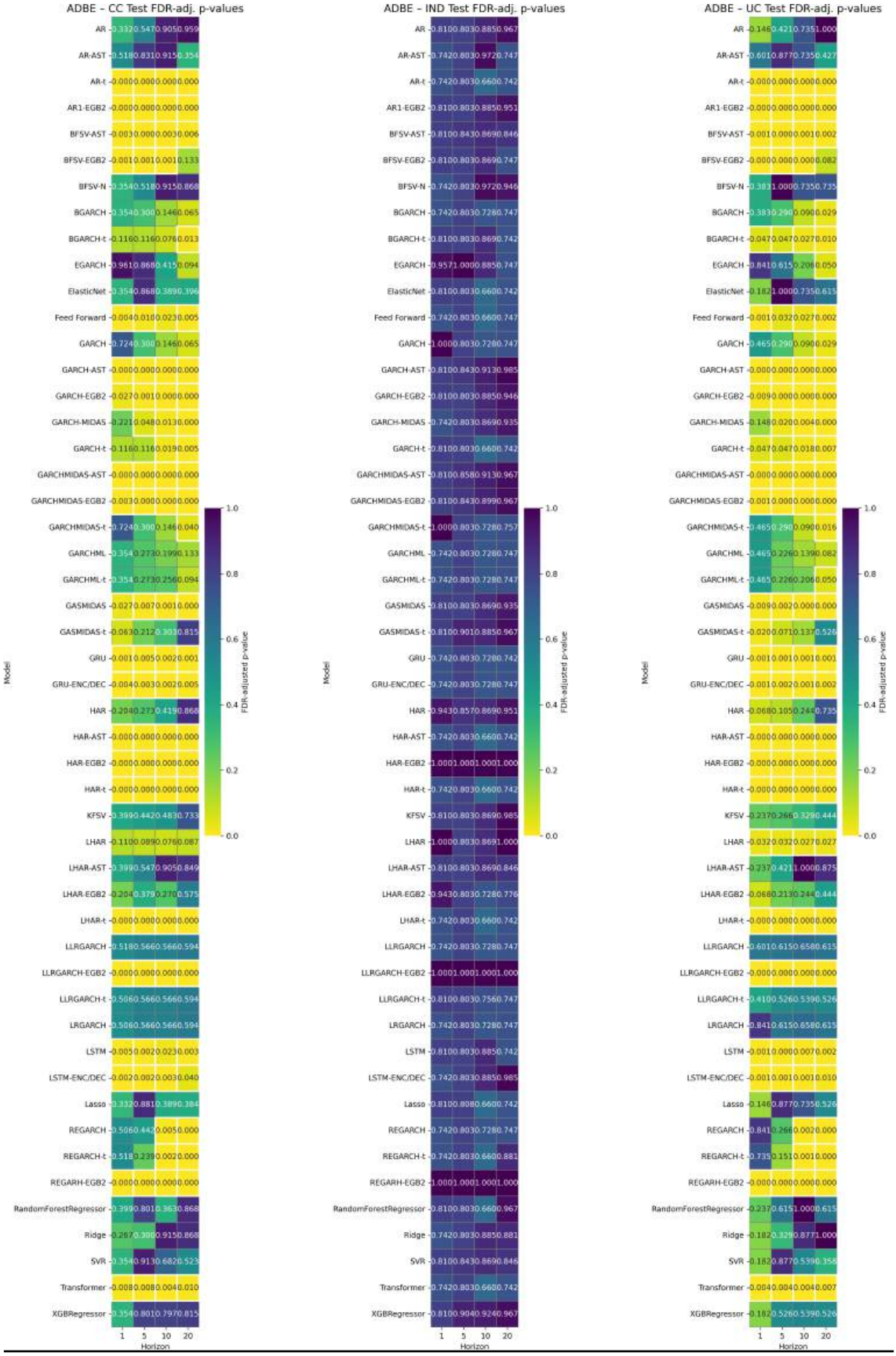
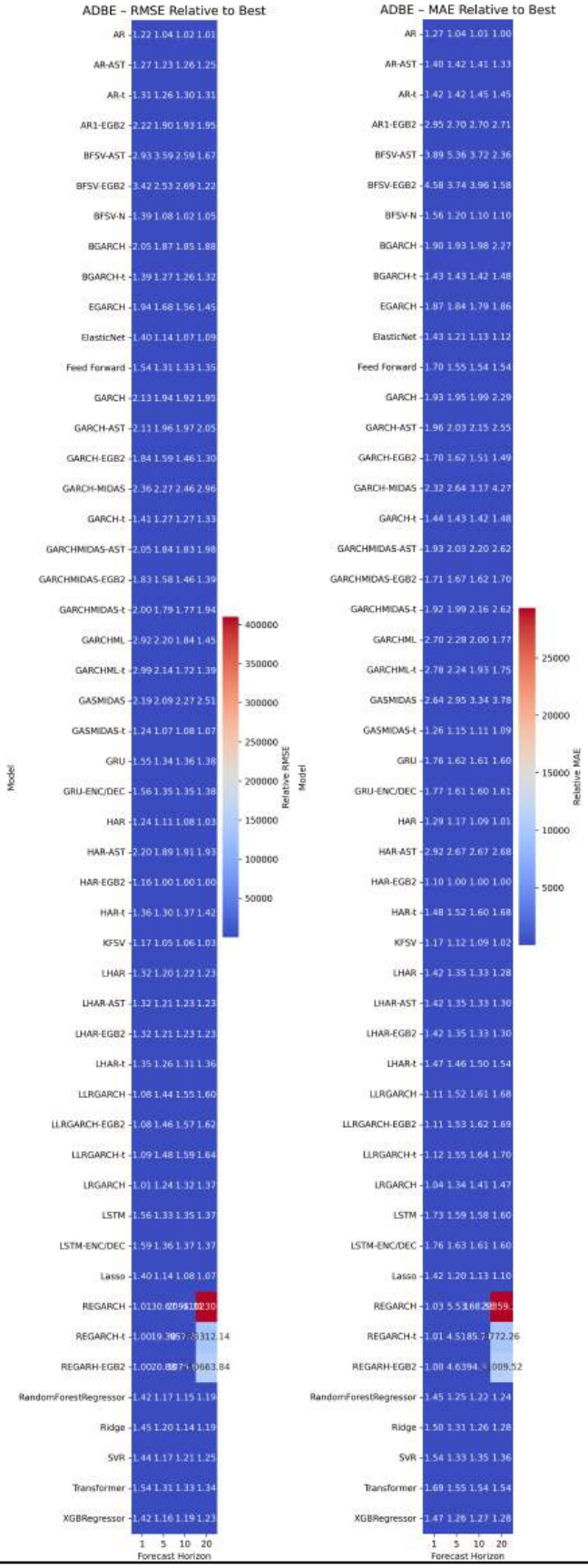


Figure E.10: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for Adobe—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


ADBE – CC Test FDR-adj. p-values
ADBE – IND Test FDR-adj. p-values
ADBE – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value


ADBE – RMSE Relative to Best
ADBE – MAE Relative to Best
Model
Relative RMSE
Relative MAE
Forecast Horizon
1 5 10 20
AR 1.22 1.04 1.02 1.01
AR-AST 1.27 1.23 1.26 1.25
AR-t 1.31 1.26 1.30 1.31
AR1-EGB2 2.22 1.90 1.93 1.95
BFSV-AST 2.93 3.59 2.59 1.67
BFSV-EGB2 3.42 2.53 2.69 1.22
BFSV-N 1.39 1.08 1.02 1.05
BGARCH 2.05 1.87 1.85 1.88
BGARCH-t 1.39 1.27 1.26 1.32
EGARCH 1.94 1.68 1.56 1.45
ElasticNet 1.40 1.14 1.07 1.09
Feed Forward 1.54 1.31 1.33 1.35
GARCH 2.13 1.94 1.92 1.95
GARCH-AST 2.11 1.96 1.97 2.05
GARCH-EGB2 1.84 1.59 1.46 1.30
GARCH-MIDAS 2.36 2.27 2.46 2.96
GARCH-t 1.41 1.27 1.27 1.33
GARCHMIDAS-AST 2.05 1.84 1.83 1.98
GARCHMIDAS-EGB2 1.83 1.58 1.46 1.39
GARCHMIDAS-t 2.00 1.79 1.77 1.94
GARCHML 2.92 2.20 1.84 1.45
GARCHML-t 2.99 2.14 1.72 1.39
GASMIDAS 2.19 2.09 2.27 2.51
GASMIDAS-t 1.24 1.07 1.08 1.07
GRU 1.55 1.34 1.36 1.38
GRU-ENC/DEC 1.56 1.35 1.35 1.38
HAR 1.24 1.11 1.08 1.03
HAR-AST 2.20 1.89 1.91 1.93
HAR-EGB2 1.16 1.00 1.00 1.00
HAR-t 1.36 1.30 1.37 1.42
KFSV 1.17 1.05 1.06 1.03
LHAR 1.32 1.20 1.22 1.23
LHAR-AST 1.32 1.21 1.23 1.23
LHAR-EGB2 1.32 1.21 1.23 1.23
LHAR-t 1.35 1.26 1.31 1.36
LLRGARCH 1.08 1.44 1.55 1.60
LLRGARCH-EGB2 1.08 1.46 1.57 1.62
LLRGARCH-t 1.09 1.48 1.59 1.64
LRGARCH 1.01 1.24 1.32 1.37
LSTM 1.56 1.33 1.35 1.37
LSTM-ENC/DEC 1.59 1.36 1.37 1.37
Lasso 1.40 1.14 1.08 1.07
REGARCH 1.03 5.53 168.29 859.
REGARCH-t 1.01 4.51 85.77 772.26
REGARH-EGB2 1.08 4.63 94.[illegible] 1009.52
RandomForestRegressor 1.42 1.17 1.15 1.19
Ridge 1.45 1.26 1.14 1.19
SVR 1.44 1.17 1.21 1.25
Transformer 1.54 1.31 1.33 1.34
XGBRegressor 1.42 1.16 1.19 1.23
400000
350000
300000
250000
200000
150000
100000
50000
25000
20000
15000
10000
5000

E.3 AMD

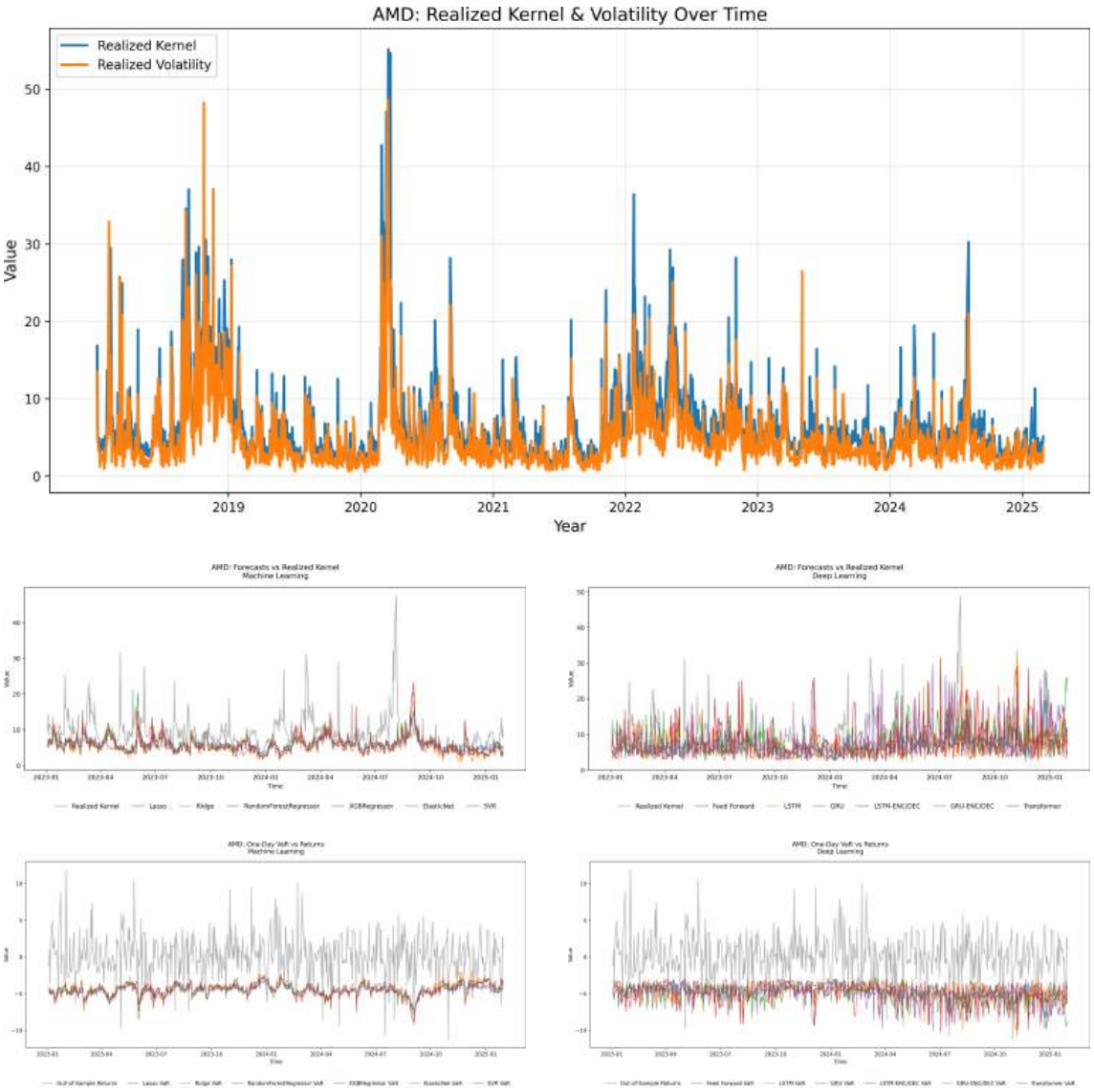
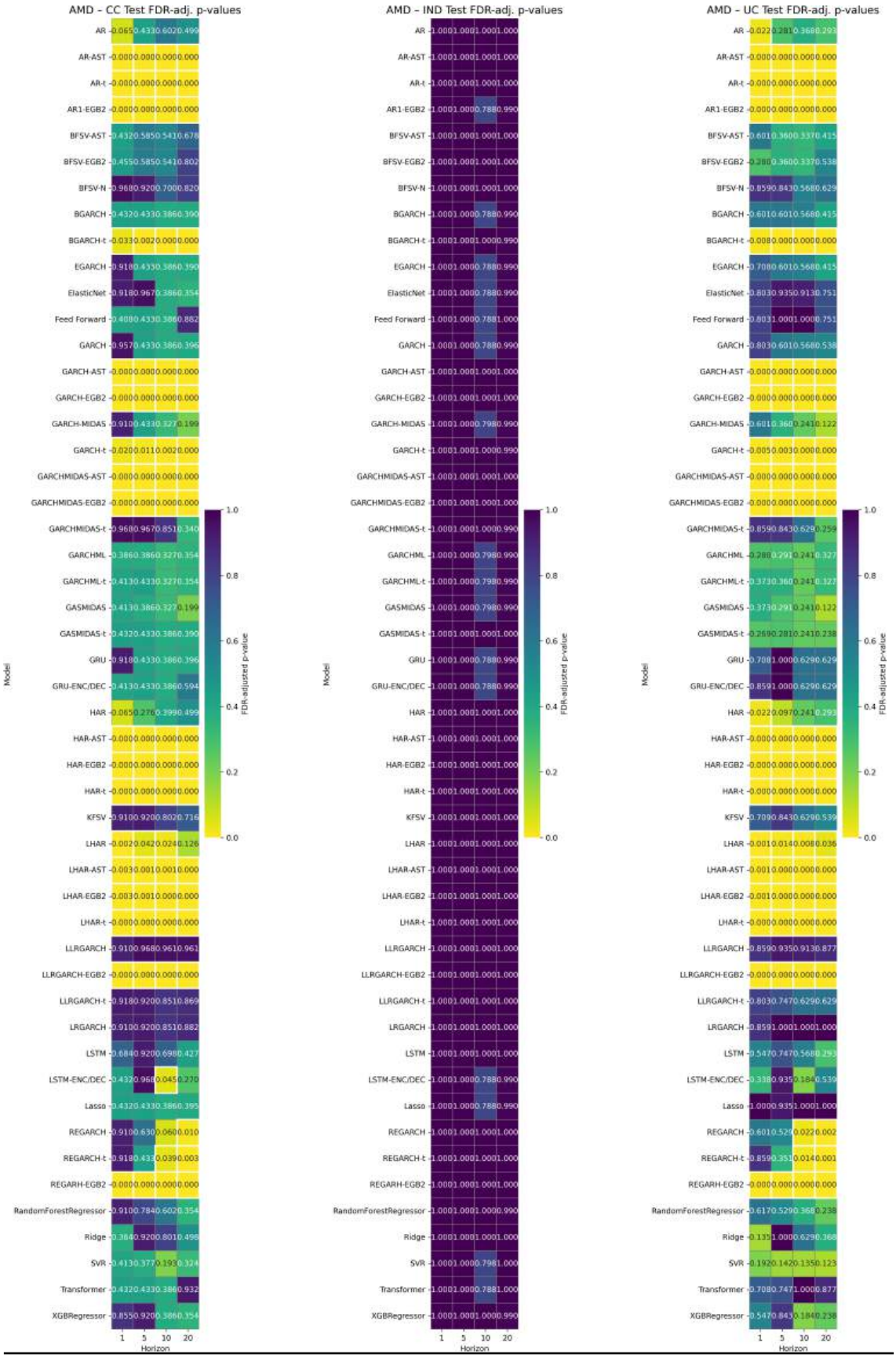
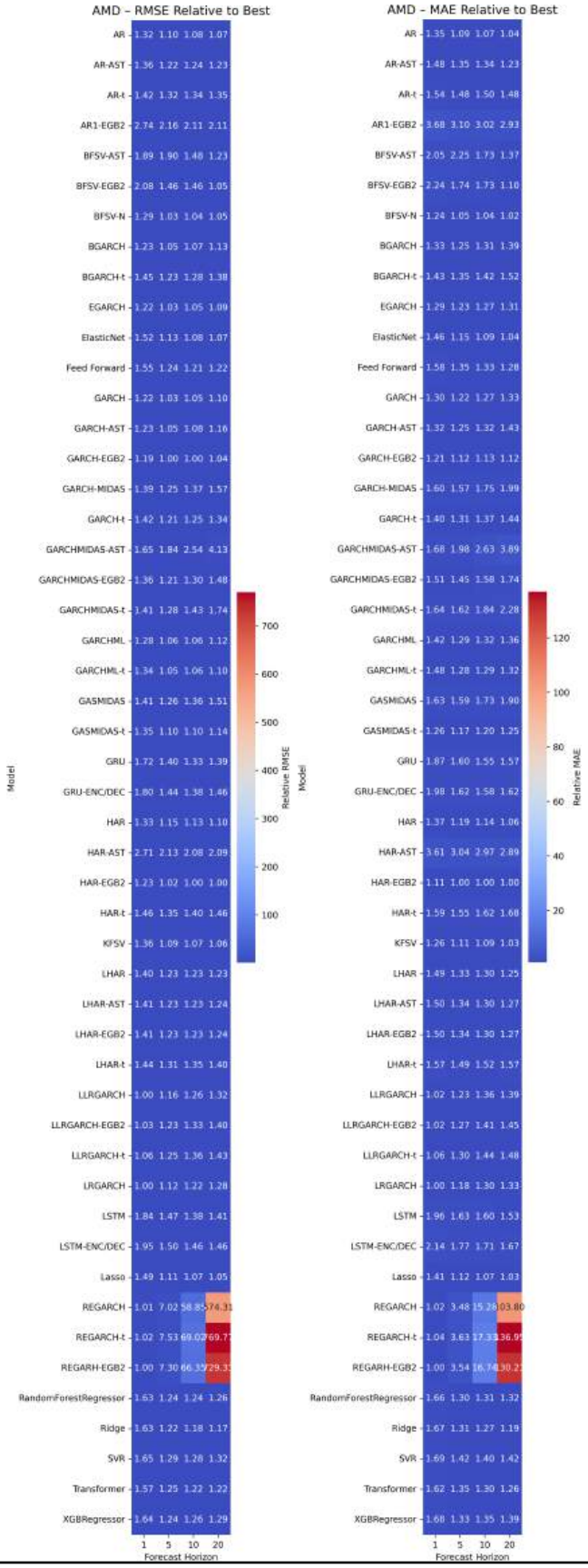


Figure E.13: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for AMD—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


AMD – CC Test FDR-adj. p-values
AMD – IND Test FDR-adj. p-values
AMD – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value


AMD – RMSE Relative to Best
AMD – MAE Relative to Best
Model
Relative RMSE
Relative MAE
Forecast Horizon
1
5
10
20
AR
AR-AST
AR-t
AR1-EGB2
BFSV-AST
BFSV-EGB2
BFSV-N
BGARCH
BGARCH-t
EGARCH
ElasticNet
Feed Forward
GARCH
GARCH-AST
GARCH-EGB2
GARCH-MIDAS
GARCH-t
GARCHMIDAS-AST
GARCHMIDAS-EGB2
GARCHMIDAS-t
GARCHML
GARCHML-t
GASMIDAS
GASMIDAS-t
GRU
GRU-ENC/DEC
HAR
HAR-AST
HAR-EGB2
HAR-t
KFSV
LHAR
LHAR-AST
LHAR-EGB2
LHAR-t
LLRGARCH
LLRGARCH-EGB2
LLRGARCH-t
LRGARCH
LSTM
LSTM-ENC/DEC
Lasso
REGARCH
REGARCH-t
REGARH-EGB2
RandomForestRegressor
Ridge
SVR
Transformer
XGBRegressor

E.4 Amazon

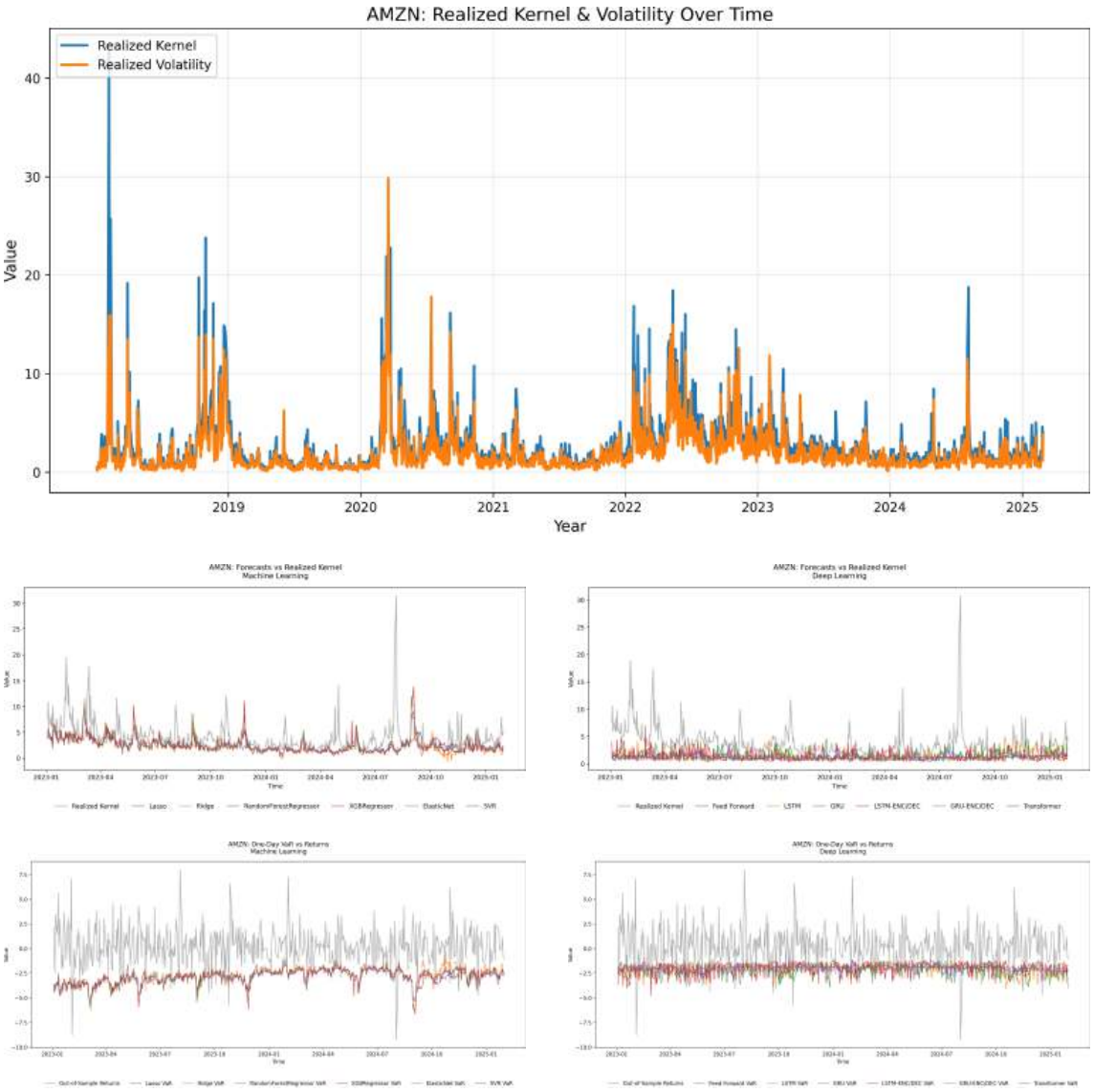
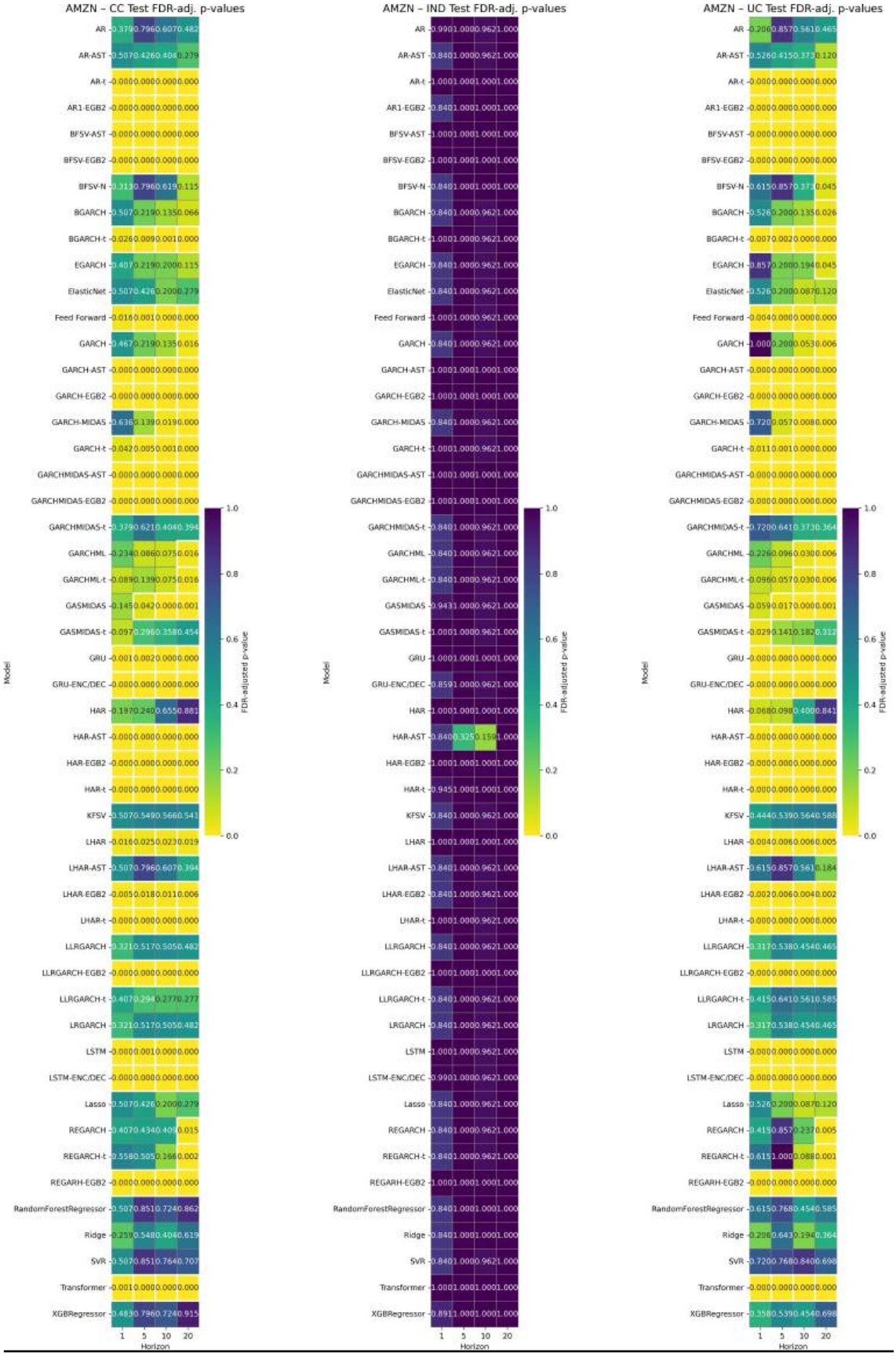
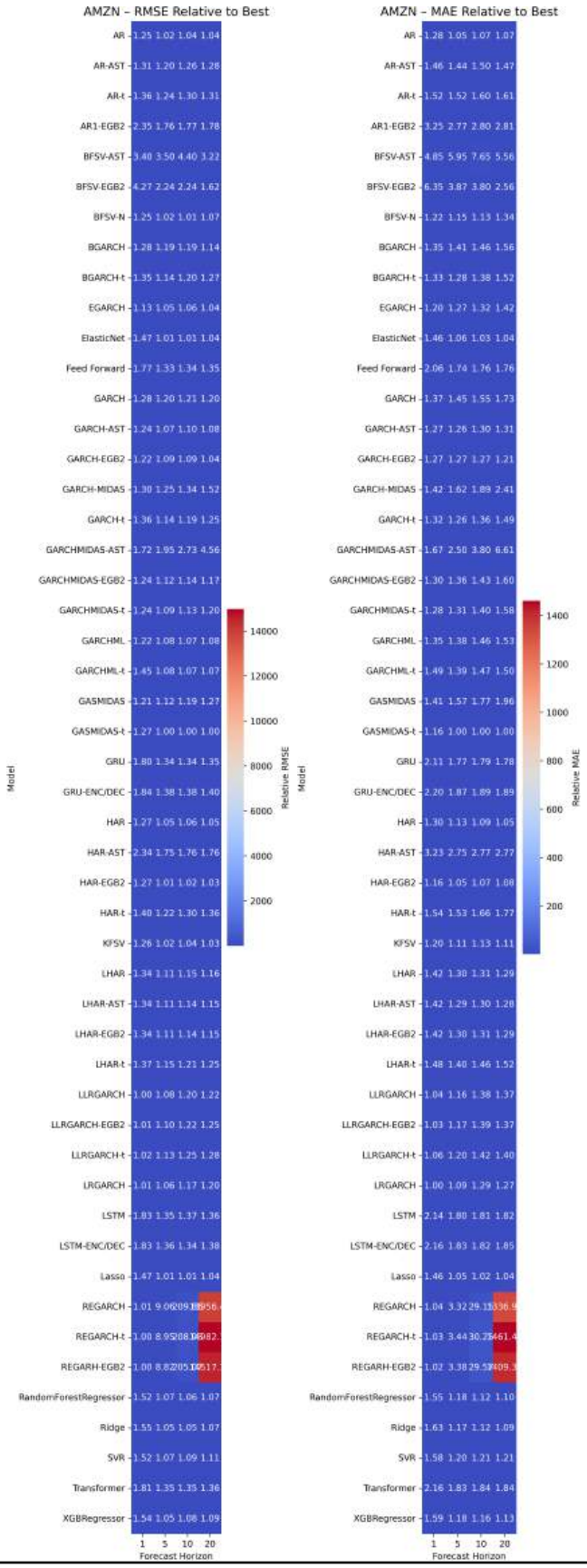


Figure E.16: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for Amazon—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


AMZN – CC Test FDR-adj. p-values
AMZN – IND Test FDR-adj. p-values
AMZN – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value
AR
AR-AST
AR-t
AR1-EGB2
BFSV-AST
BFSV-EGB2
BFSV-N
BGARCH
BGARCH-t
EGARCH
ElasticNet
Feed Forward
GARCH
GARCH-AST
GARCH-EGB2
GARCH-MIDAS
GARCH-t
GARCHMIDAS-AST
GARCHMIDAS-EGB2
GARCHMIDAS-t
GARCHML
GARCHML-t
GASMIDAS
GASMIDAS-t
GRU
GRU-ENC/DEC
HAR
HAR-AST
HAR-EGB2
HAR-t
KFSV
LHAR
LHAR-AST
LHAR-EGB2
LHAR-t
LLRGARCH
LLRGARCH-EGB2
LLRGARCH-t
LRGARCH
LSTM
LSTM-ENC/DEC
Lasso
REGARCH
REGARCH-t
REGARH-EGB2
RandomForestRegressor
Ridge
SVR
Transformer
XGBRegressor


AMZN – RMSE Relative to Best
AMZN – MAE Relative to Best
Model
Relative RMSE
Relative MAE
Forecast Horizon
1 5 10 20
AR 1.25 1.02 1.04 1.04
AR-AST 1.31 1.20 1.26 1.28
AR-t 1.36 1.24 1.30 1.31
AR1-EGB2 2.35 1.76 1.77 1.78
BFSV-AST 3.40 3.50 4.40 3.22
BFSV-EGB2 4.27 2.24 2.24 1.62
BFSV-N 1.25 1.02 1.01 1.07
BGARCH 1.28 1.19 1.19 1.14
BGARCH-t 1.35 1.14 1.20 1.27
EGARCH 1.13 1.05 1.06 1.04
ElasticNet 1.47 1.01 1.01 1.04
Feed Forward 1.77 1.33 1.34 1.35
GARCH 1.28 1.20 1.21 1.20
GARCH-AST 1.24 1.07 1.10 1.08
GARCH-EGB2 1.22 1.09 1.09 1.04
GARCH-MIDAS 1.30 1.25 1.34 1.52
GARCH-t 1.36 1.14 1.19 1.25
GARCHMIDAS-AST 1.72 1.95 2.73 4.56
GARCHMIDAS-EGB2 1.24 1.12 1.14 1.17
GARCHMIDAS-t 1.24 1.09 1.13 1.20
GARCHML 1.22 1.08 1.07 1.08
GARCHML-t 1.45 1.08 1.07 1.07
GASMIDAS 1.21 1.12 1.19 1.27
GASMIDAS-t 1.27 1.00 1.00 1.00
GRU 1.80 1.34 1.34 1.35
GRU-ENC/DEC 1.84 1.38 1.38 1.40
HAR 1.27 1.05 1.06 1.05
HAR-AST 2.34 1.75 1.76 1.76
HAR-EGB2 1.27 1.01 1.02 1.03
HAR-t 1.40 1.22 1.30 1.36
KFSV 1.26 1.02 1.04 1.03
LHAR 1.34 1.11 1.15 1.16
LHAR-AST 1.34 1.11 1.14 1.15
LHAR-EGB2 1.34 1.11 1.14 1.15
LHAR-t 1.37 1.15 1.21 1.25
LLRGARCH 1.00 1.08 1.20 1.22
LLRGARCH-EGB2 1.01 1.10 1.22 1.25
LLRGARCH-t 1.02 1.13 1.25 1.28
LRGARCH 1.01 1.06 1.17 1.20
LSTM 1.83 1.35 1.37 1.36
LSTM-ENC/DEC 1.83 1.36 1.34 1.38
Lasso 1.47 1.01 1.01 1.04
RandomForestRegressor 1.52 1.07 1.06 1.07
Ridge 1.55 1.05 1.05 1.07
SVR 1.52 1.07 1.09 1.11
Transformer 1.81 1.35 1.35 1.36
XGBRegressor 1.54 1.05 1.08 1.09
AR 1.28 1.05 1.07 1.07
AR-AST 1.46 1.44 1.50 1.47
AR-t 1.52 1.52 1.60 1.61
AR1-EGB2 3.25 2.77 2.80 2.81
BFSV-AST 4.85 5.95 7.65 5.56
BFSV-EGB2 6.35 3.87 3.80 2.56
BFSV-N 1.22 1.15 1.13 1.34
BGARCH 1.35 1.41 1.46 1.56
BGARCH-t 1.33 1.28 1.38 1.52
EGARCH 1.20 1.27 1.32 1.42
ElasticNet 1.46 1.06 1.03 1.04
Feed Forward 2.06 1.74 1.76 1.76
GARCH 1.37 1.45 1.55 1.73
GARCH-AST 1.27 1.26 1.30 1.31
GARCH-EGB2 1.27 1.27 1.27 1.21
GARCH-MIDAS 1.42 1.62 1.89 2.41
GARCH-t 1.32 1.26 1.36 1.49
GARCHMIDAS-AST 1.67 2.50 3.80 6.61
GARCHMIDAS-EGB2 1.30 1.36 1.43 1.60
GARCHMIDAS-t 1.28 1.31 1.40 1.58
GARCHML 1.35 1.38 1.46 1.53
GARCHML-t 1.49 1.39 1.47 1.50
GASMIDAS 1.41 1.57 1.77 1.96
GASMIDAS-t 1.16 1.00 1.00 1.00
GRU 2.11 1.77 1.79 1.78
GRU-ENC/DEC 2.20 1.87 1.89 1.89
HAR 1.30 1.13 1.09 1.05
HAR-AST 3.23 2.75 2.77 2.77
HAR-EGB2 1.16 1.05 1.07 1.08
HAR-t 1.54 1.53 1.66 1.77
KFSV 1.20 1.11 1.13 1.11
LHAR 1.42 1.30 1.31 1.29
LHAR-AST 1.42 1.29 1.30 1.28
LHAR-EGB2 1.42 1.30 1.31 1.29
LHAR-t 1.48 1.40 1.46 1.52
LLRGARCH 1.04 1.16 1.38 1.37
LLRGARCH-EGB2 1.03 1.17 1.39 1.37
LLRGARCH-t 1.06 1.20 1.42 1.40
LRGARCH 1.00 1.09 1.29 1.27
LSTM 2.14 1.80 1.81 1.82
LSTM-ENC/DEC 2.16 1.83 1.82 1.85
Lasso 1.46 1.05 1.02 1.04
RandomForestRegressor 1.55 1.18 1.12 1.10
Ridge 1.63 1.17 1.12 1.09
SVR 1.58 1.20 1.21 1.21
Transformer 2.16 1.83 1.84 1.84
XGBRegressor 1.59 1.18 1.16 1.13

E.5 ASML

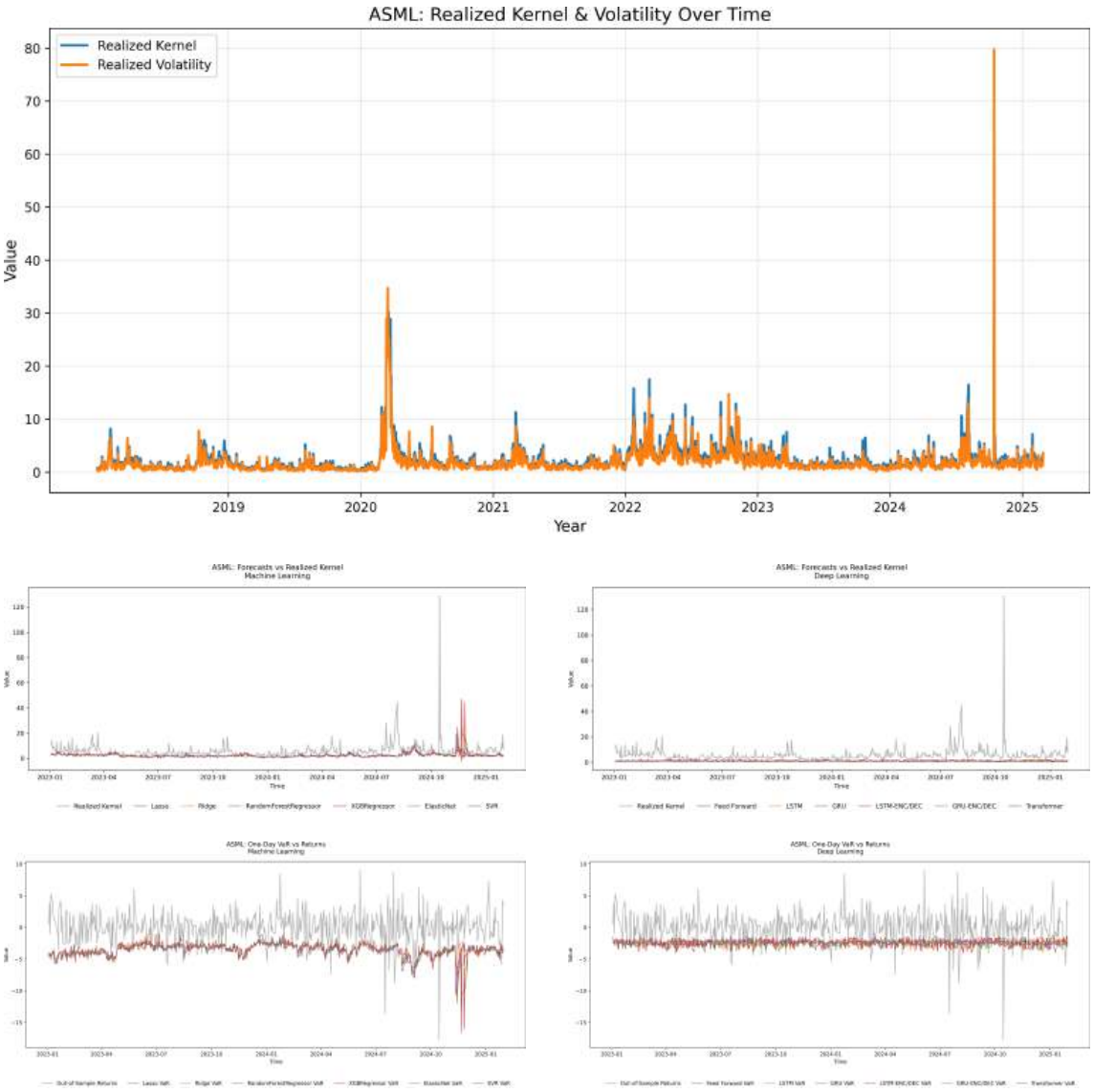
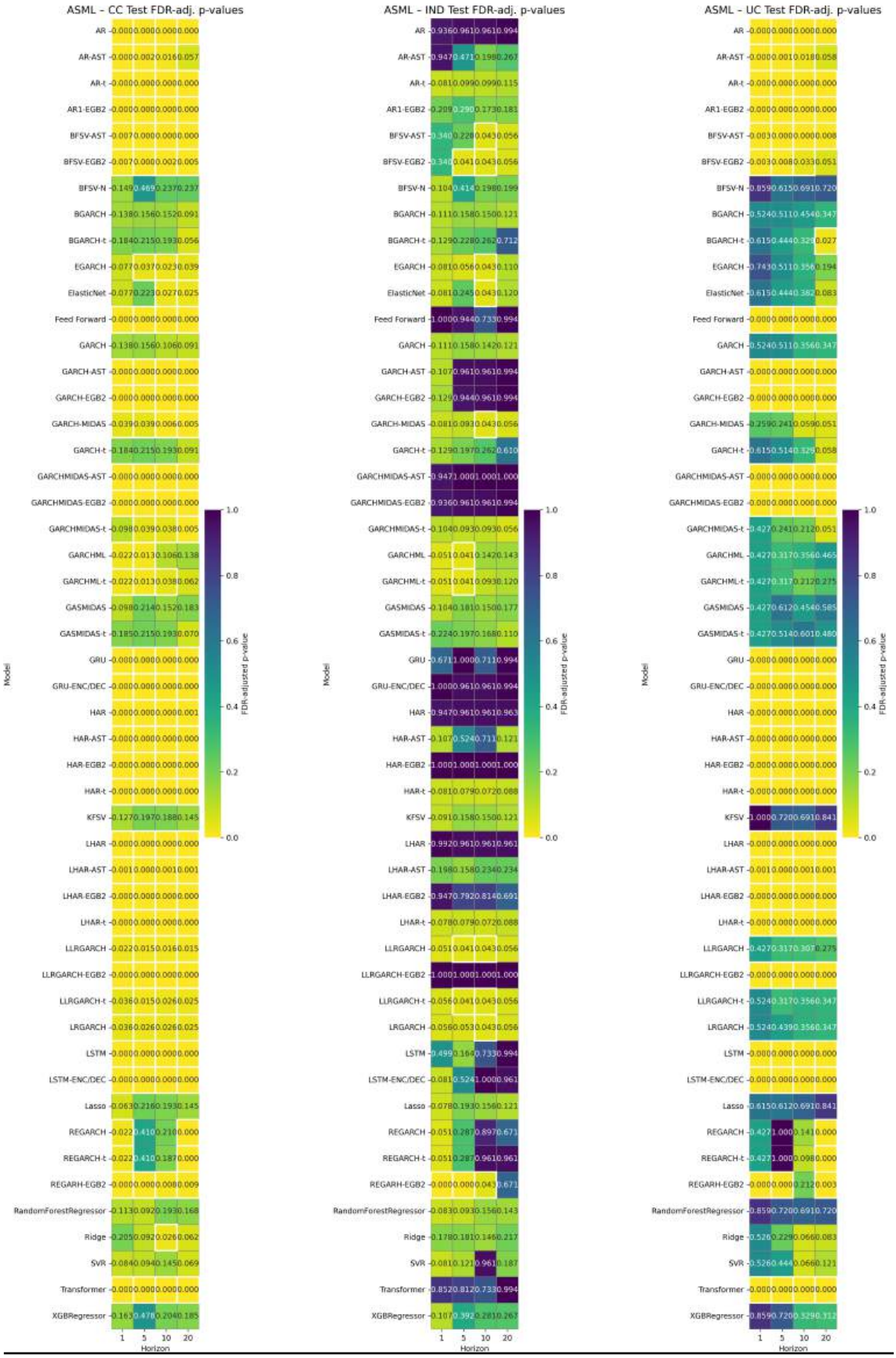
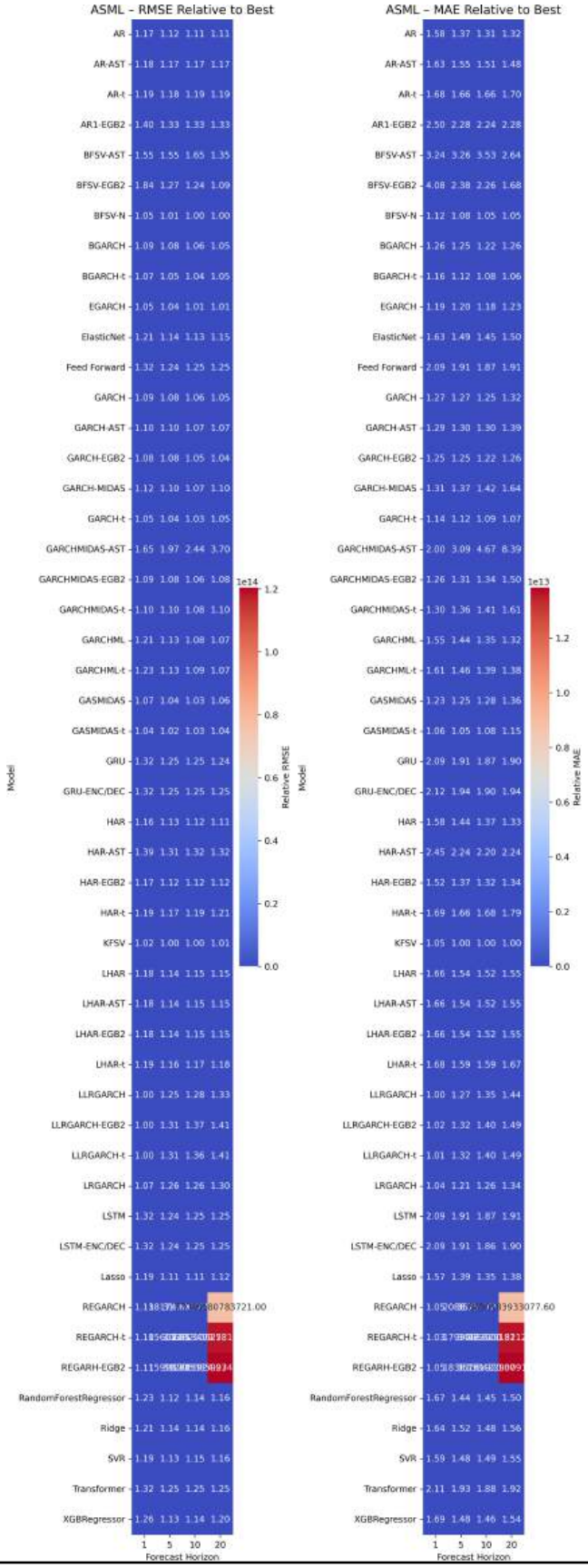


Figure E.19: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for ASML—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


ASML – CC Test FDR-adj. p-values
ASML – IND Test FDR-adj. p-values
ASML – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value


ASML – RMSE Relative to Best
ASML – MAE Relative to Best
Model
Forecast Horizon
Relative RMSE
Relative MAE
1e14
1e13

E.6 IBM

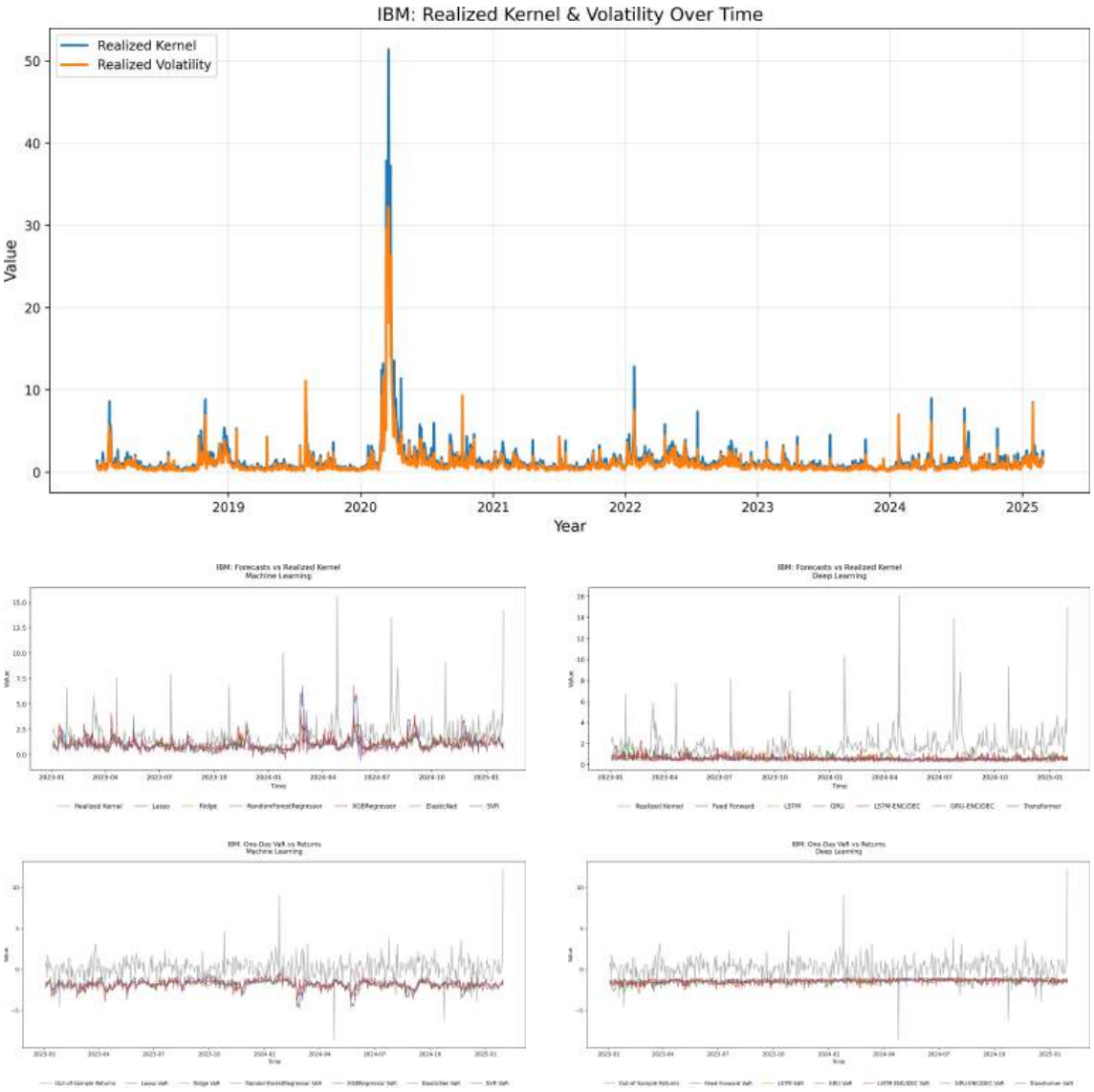
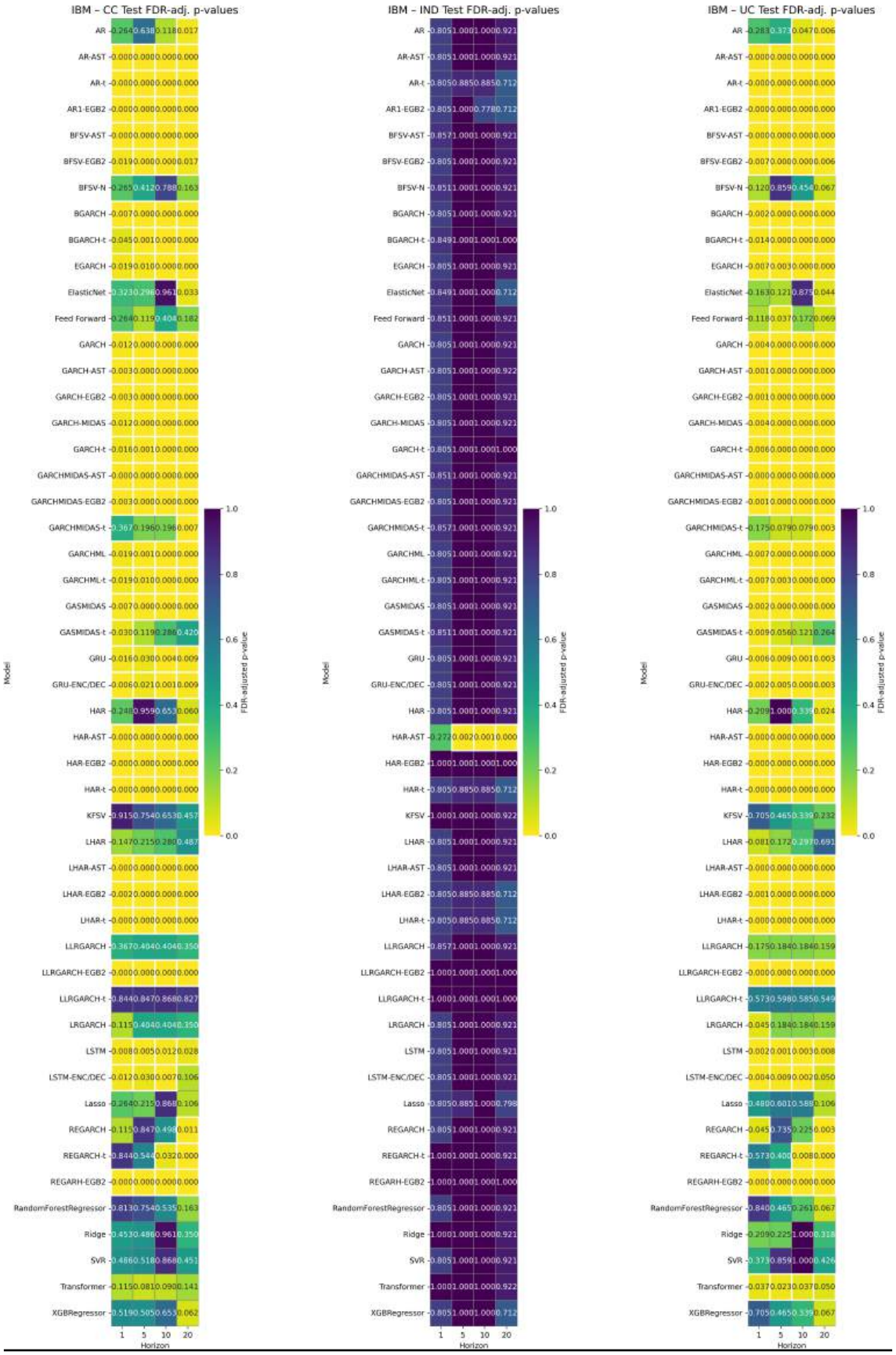
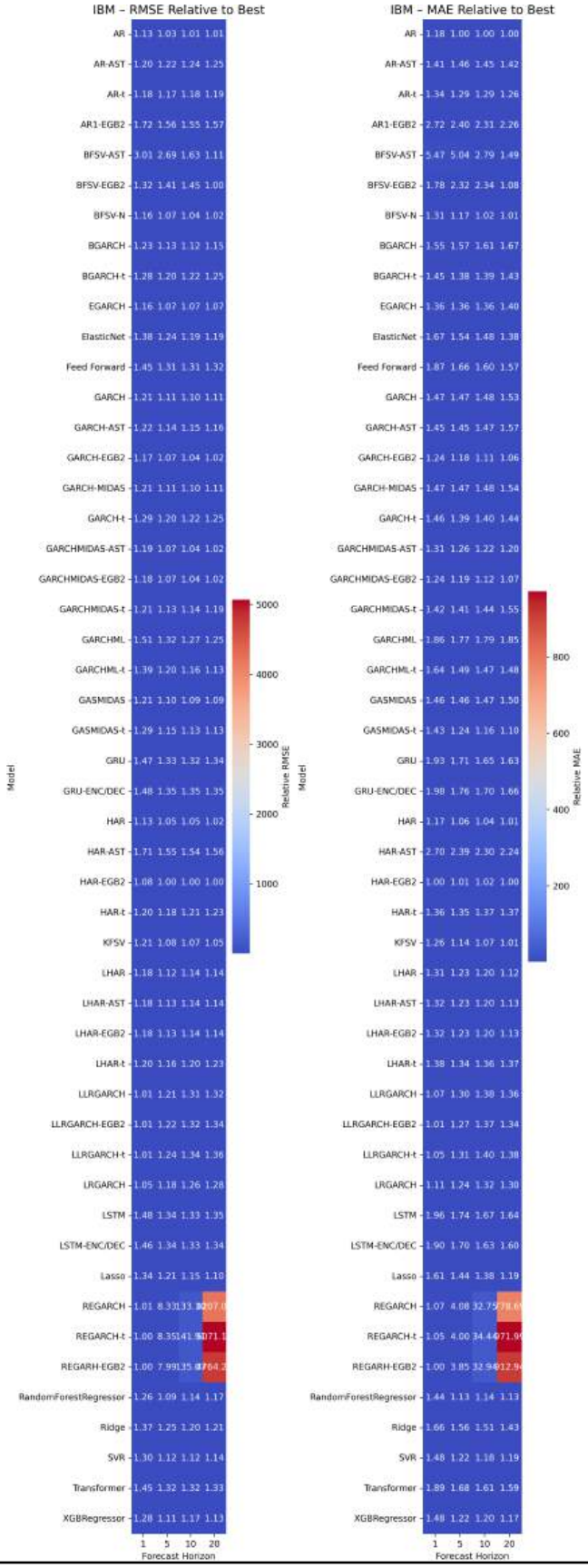


Figure E.22: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for IBM—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


IBM – CC Test FDR-adj. p-values
IBM – IND Test FDR-adj. p-values
IBM – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value


IBM – RMSE Relative to Best
IBM – MAE Relative to Best
Model
Relative RMSE
Relative MAE
Forecast Horizon
1
5
10
20
1000
2000
3000
4000
5000
200
400
600
800

E.7 Intel

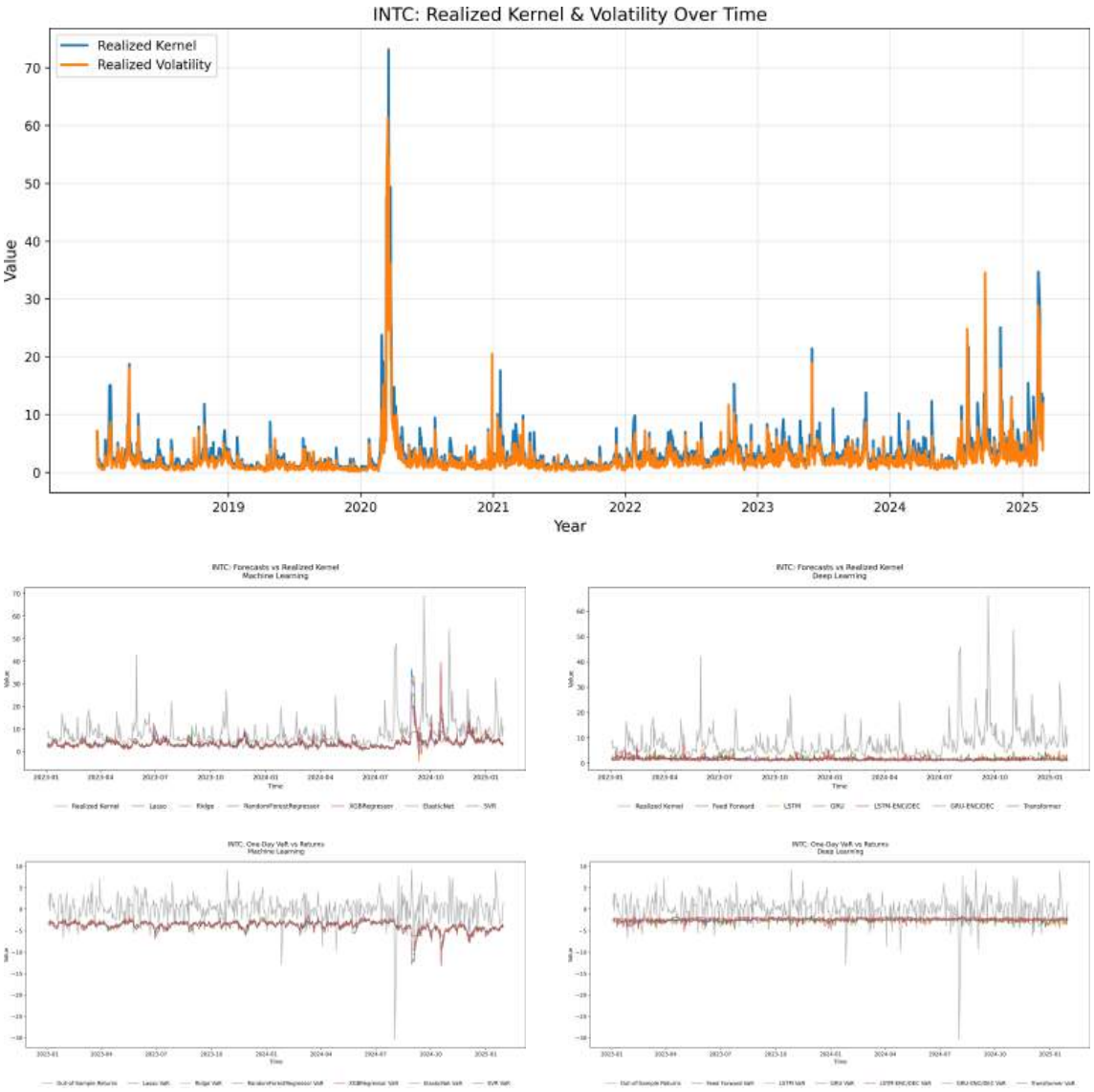
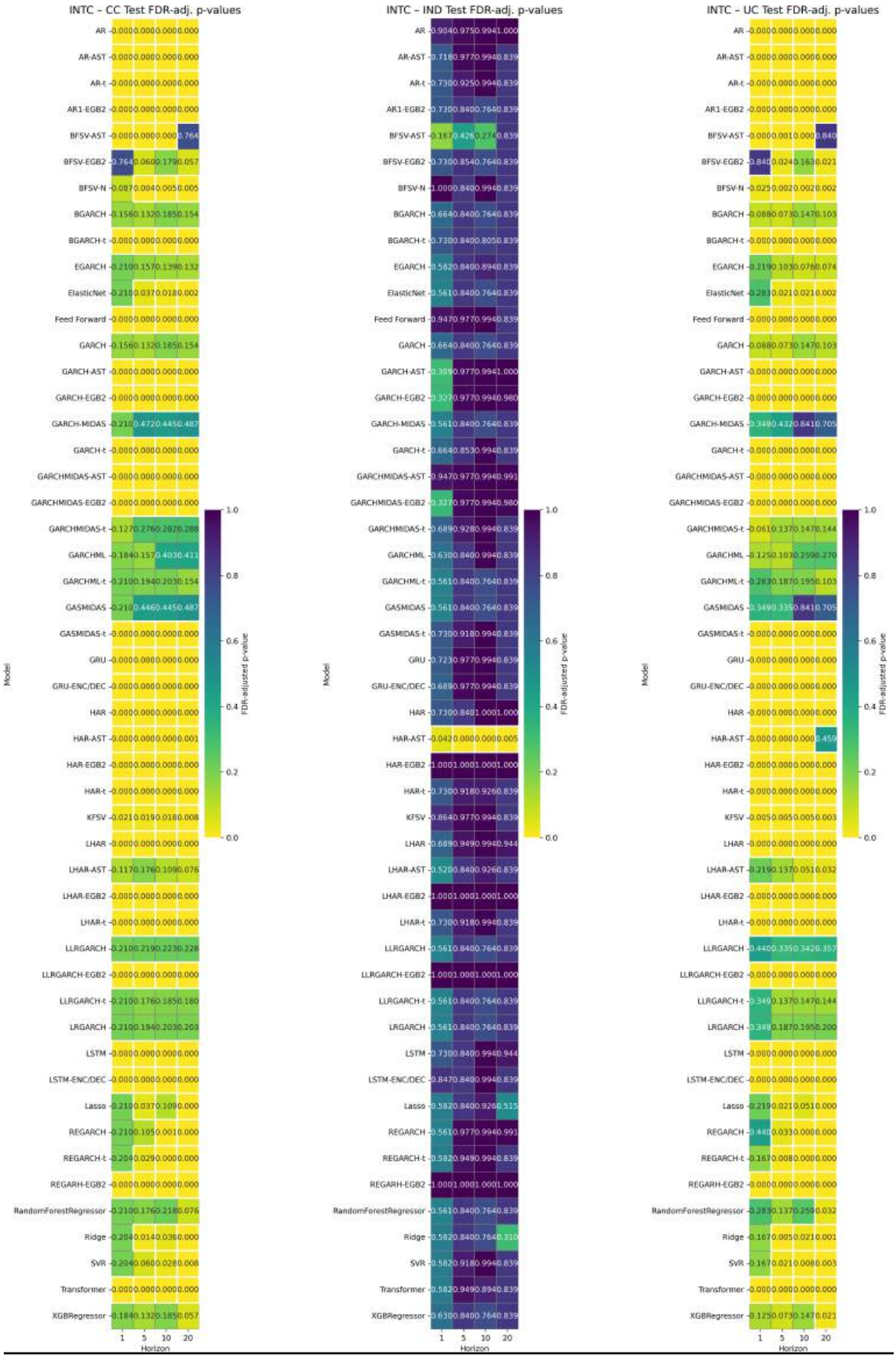
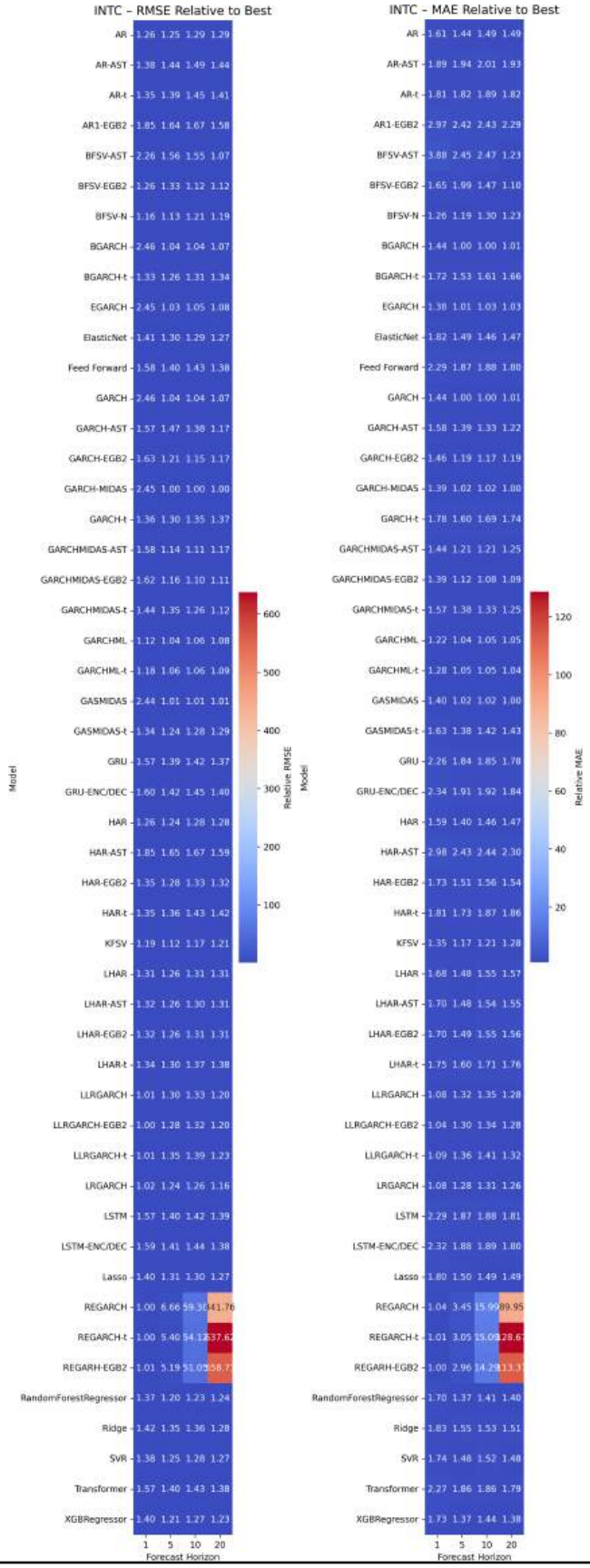


Figure E.25: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for Intel—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


INTC – CC Test FDR-adj. p-values
INTC – IND Test FDR-adj. p-values
INTC – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value


INTC – RMSE Relative to Best
INTC – MAE Relative to Best
Model
Relative RMSE
Relative MAE
Forecast Horizon
1
5
10
20
AR
AR-AST
AR-t
AR1-EGB2
BFSV-AST
BFSV-EGB2
BFSV-N
BGARCH
BGARCH-t
EGARCH
ElasticNet
Feed Forward
GARCH
GARCH-AST
GARCH-EGB2
GARCH-MIDAS
GARCH-t
GARCHMIDAS-AST
GARCHMIDAS-EGB2
GARCHMIDAS-t
GARCHML
GARCHML-t
GASMIDAS
GASMIDAS-t
GRU
GRU-ENC/DEC
HAR
HAR-AST
HAR-EGB2
HAR-t
KFSV
LHAR
LHAR-AST
LHAR-EGB2
LHAR-t
LLRGARCH
LLRGARCH-EGB2
LLRGARCH-t
LRGARCH
LSTM
LSTM-ENC/DEC
Lasso
REGARCH
REGARCH-t
REGARH-EGB2
RandomForestRegressor
Ridge
SVR
Transformer
XGBRegressor

E.8 Microsoft

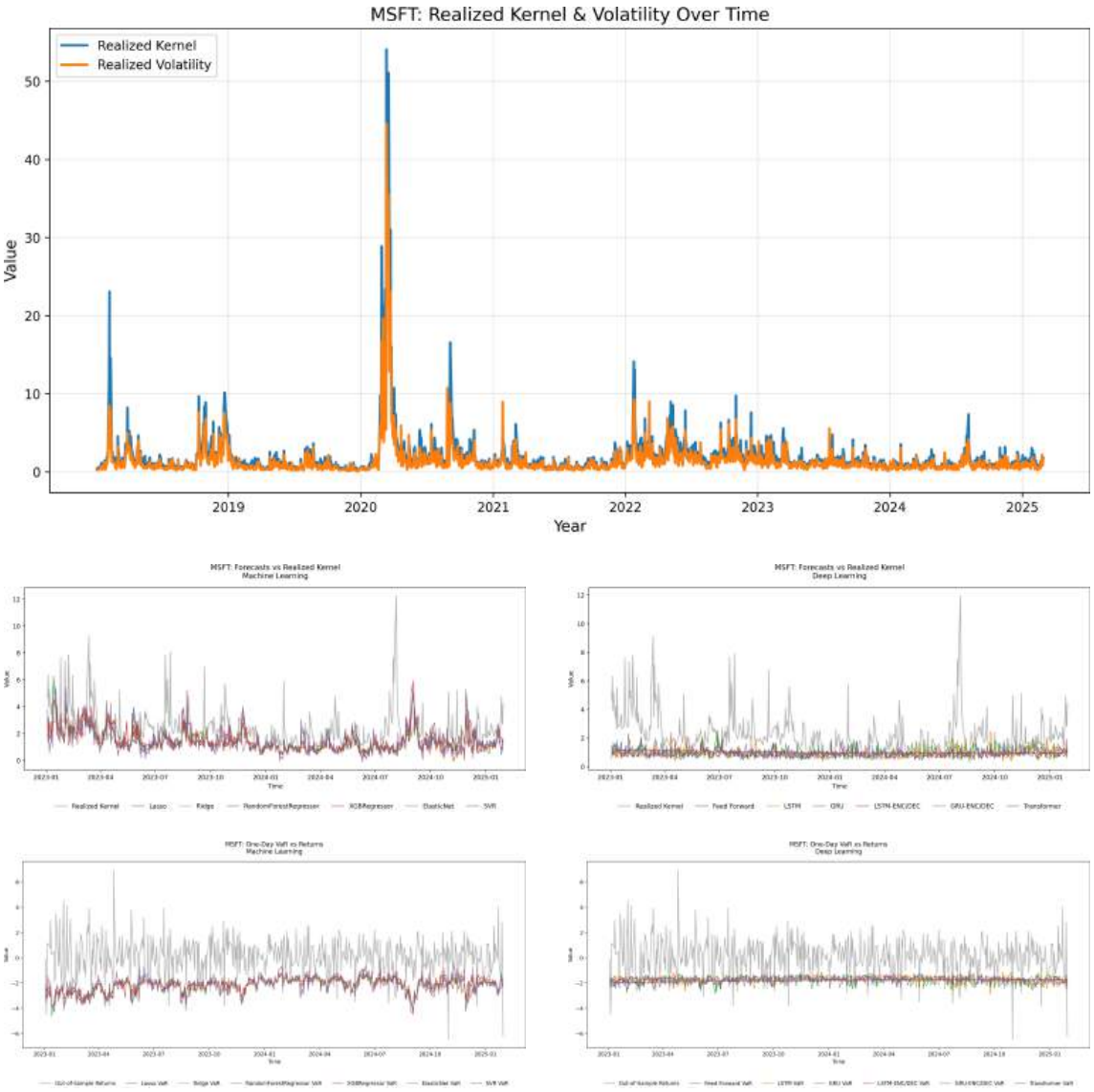
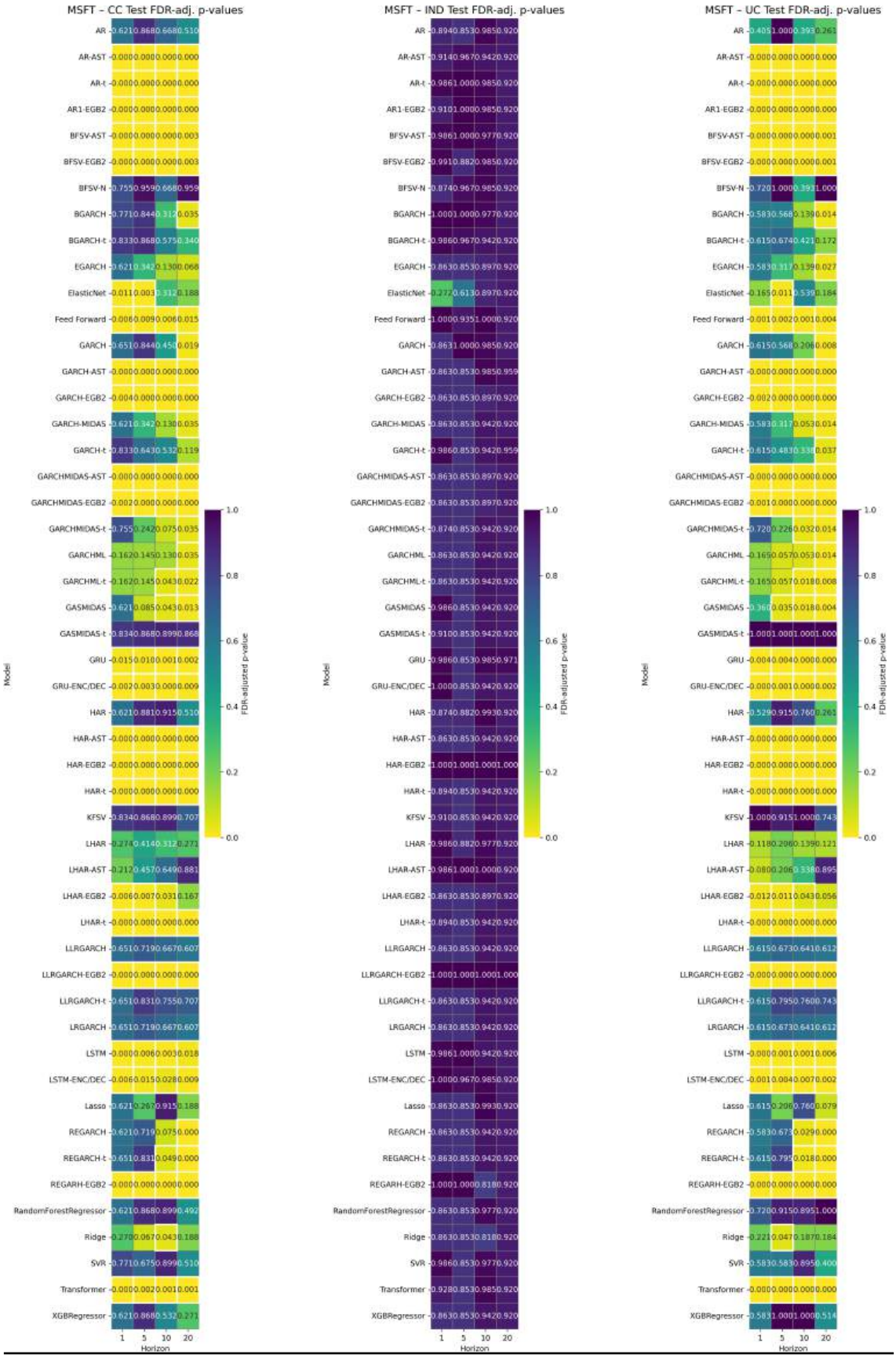
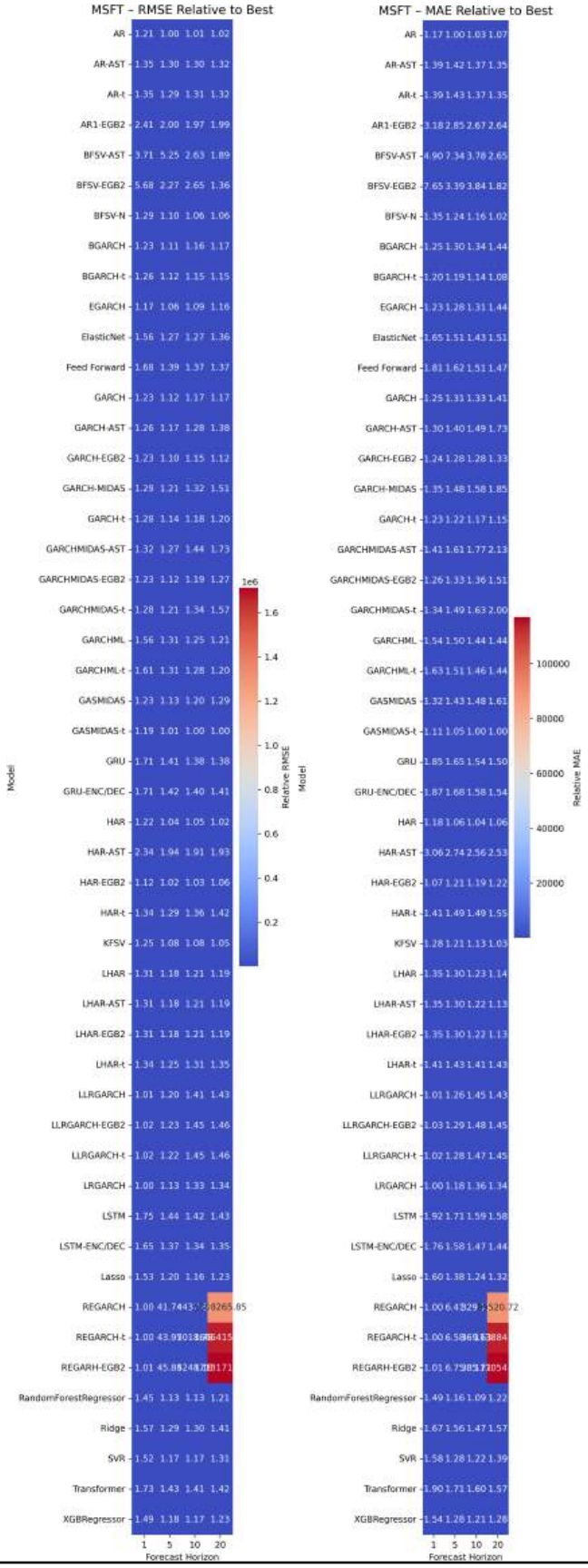


Figure E.28: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for Microsoft—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


MSFT – CC Test FDR-adj. p-values
MSFT – IND Test FDR-adj. p-values
MSFT – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value


MSFT – RMSE Relative to Best
MSFT – MAE Relative to Best
Model
Forecast Horizon
Relative RMSE
Relative MAE
1e6

E.9 NVIDIA

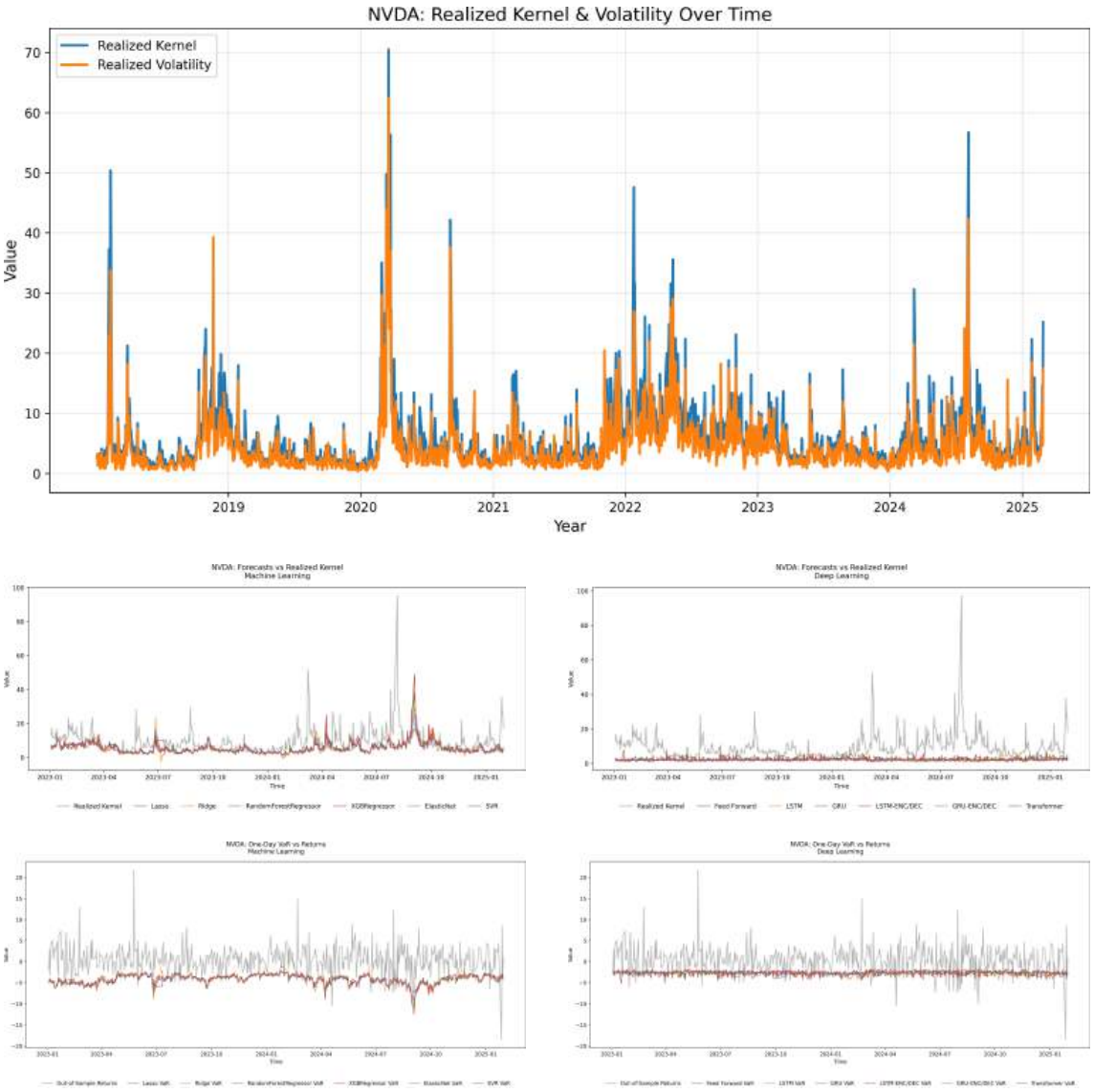
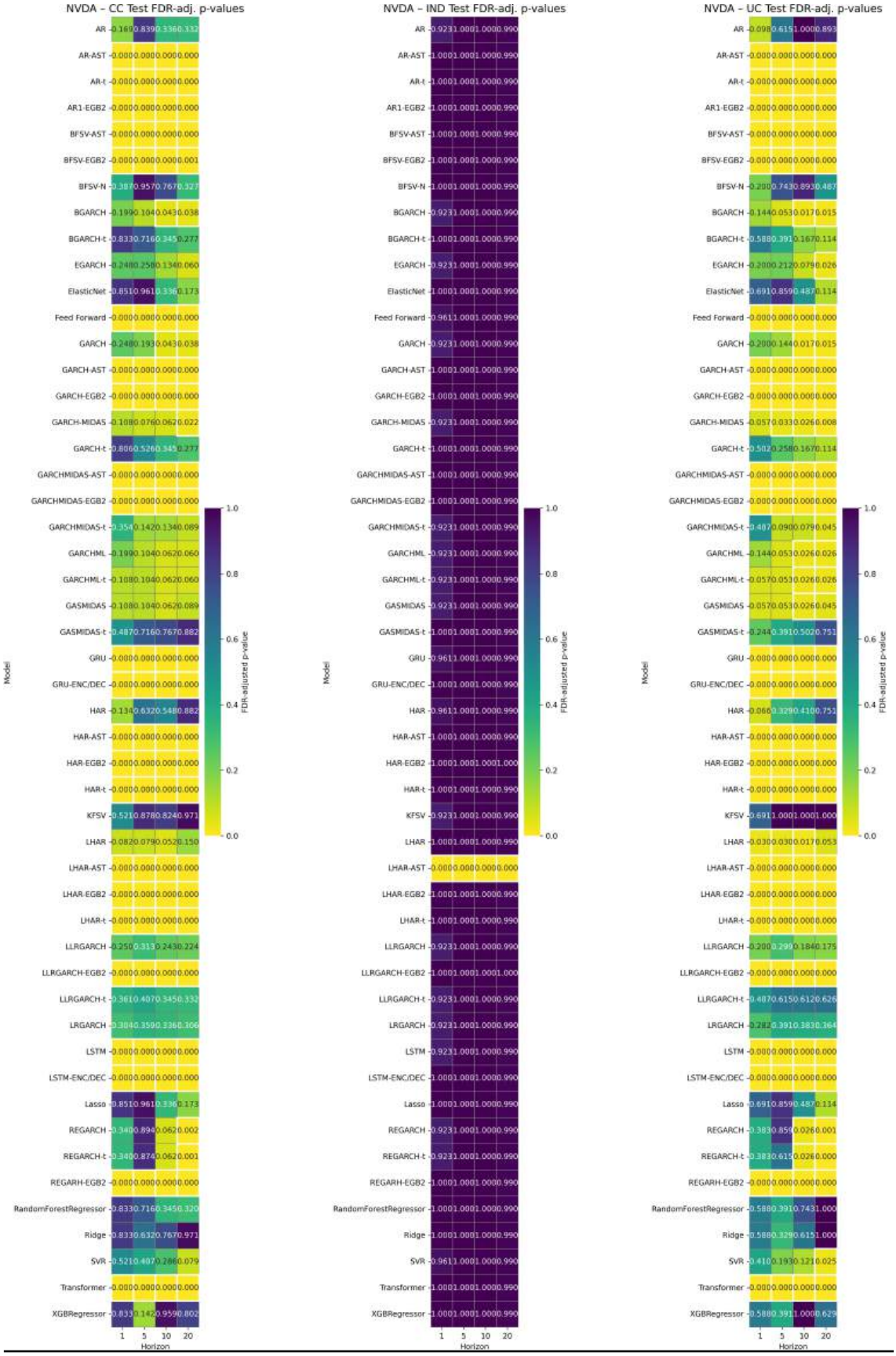
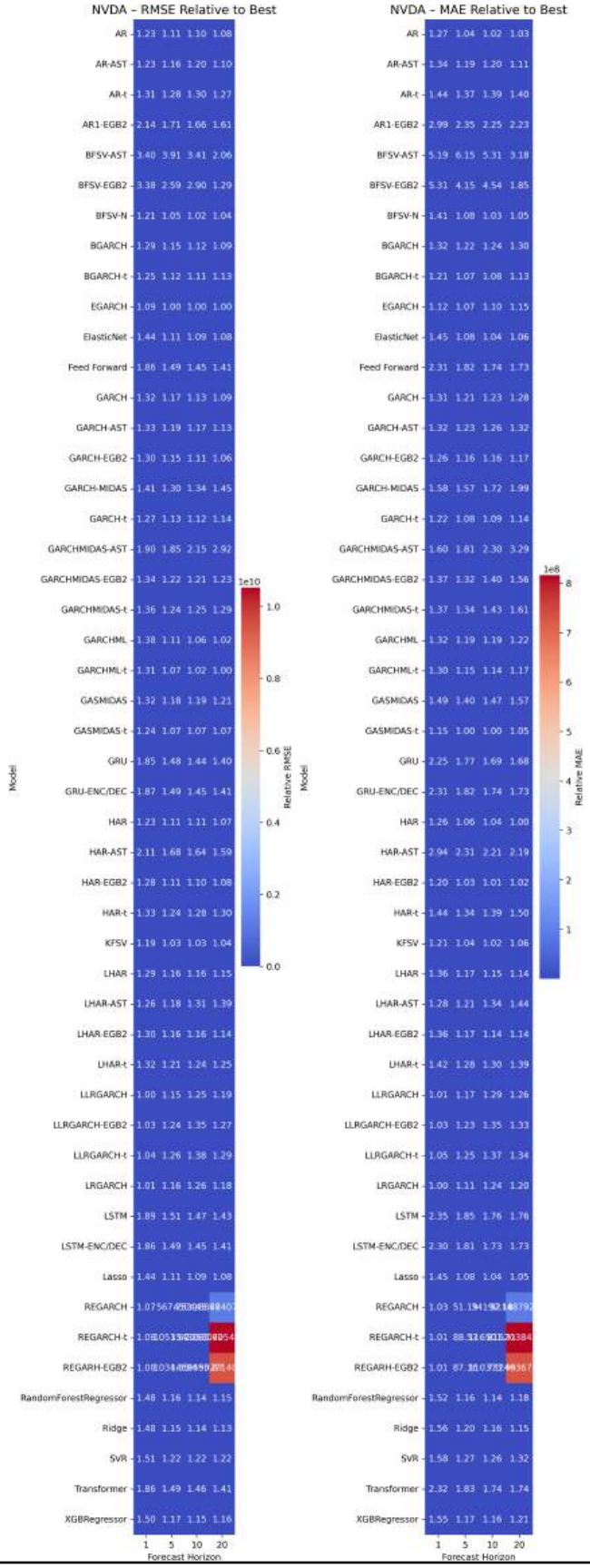


Figure E.31: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for NVIDIA—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


NVDA – CC Test FDR-adj. p-values
NVDA – IND Test FDR-adj. p-values
NVDA – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value


NVDA – RMSE Relative to Best
NVDA – MAE Relative to Best
Model
Forecast Horizon
Relative RMSE
Relative MAE
1e10
1e8

E.10 Tesla

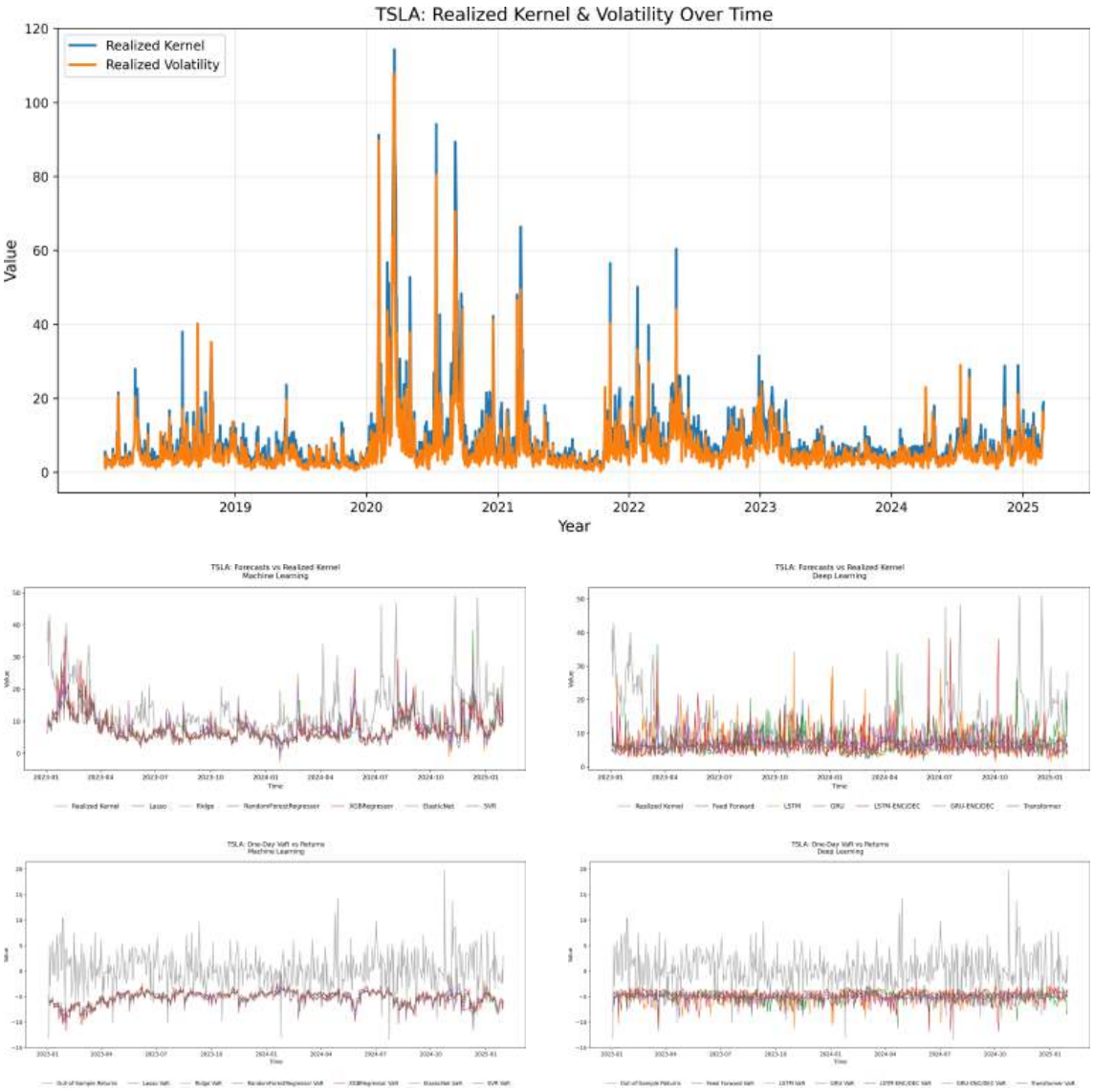
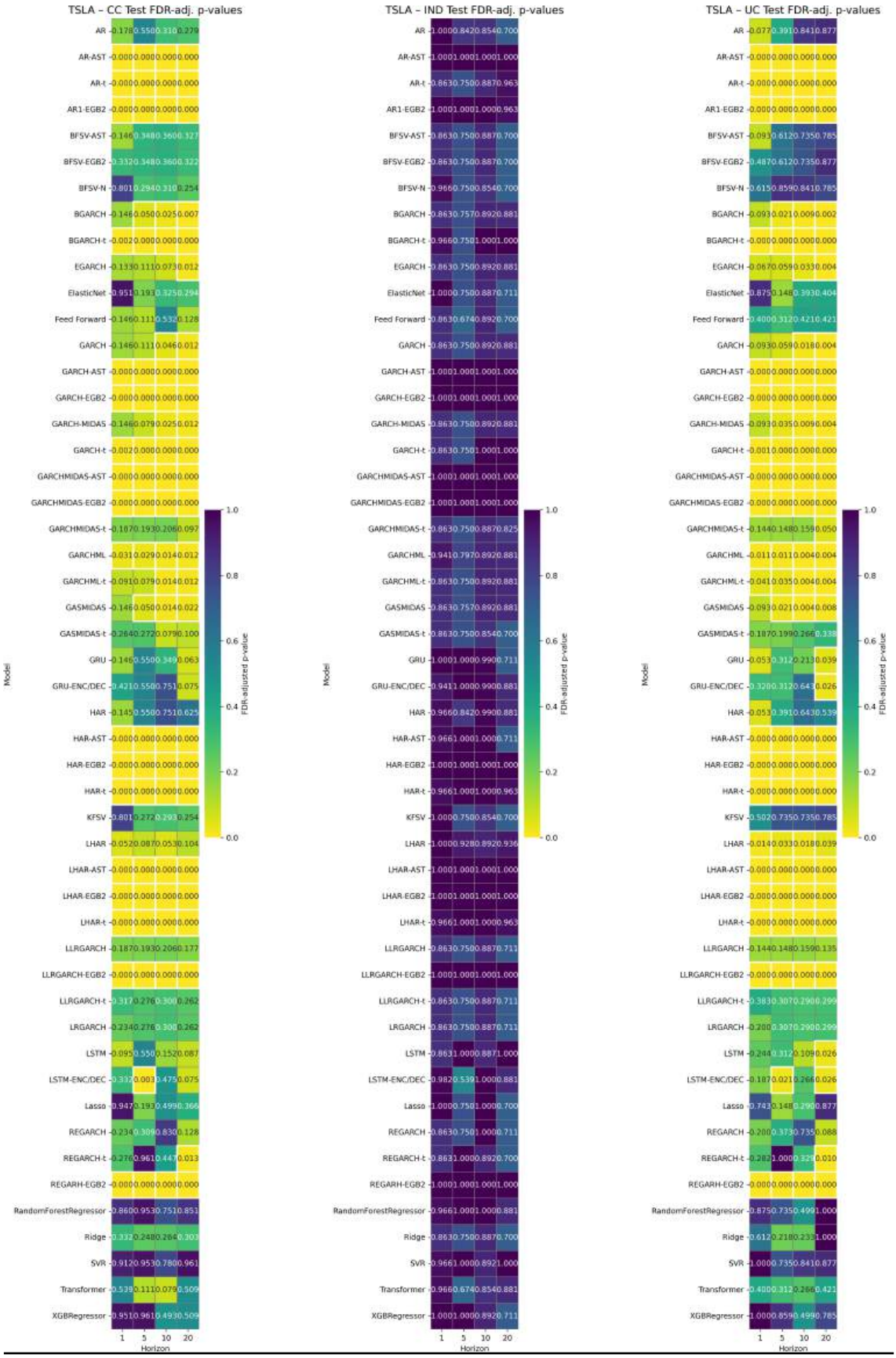
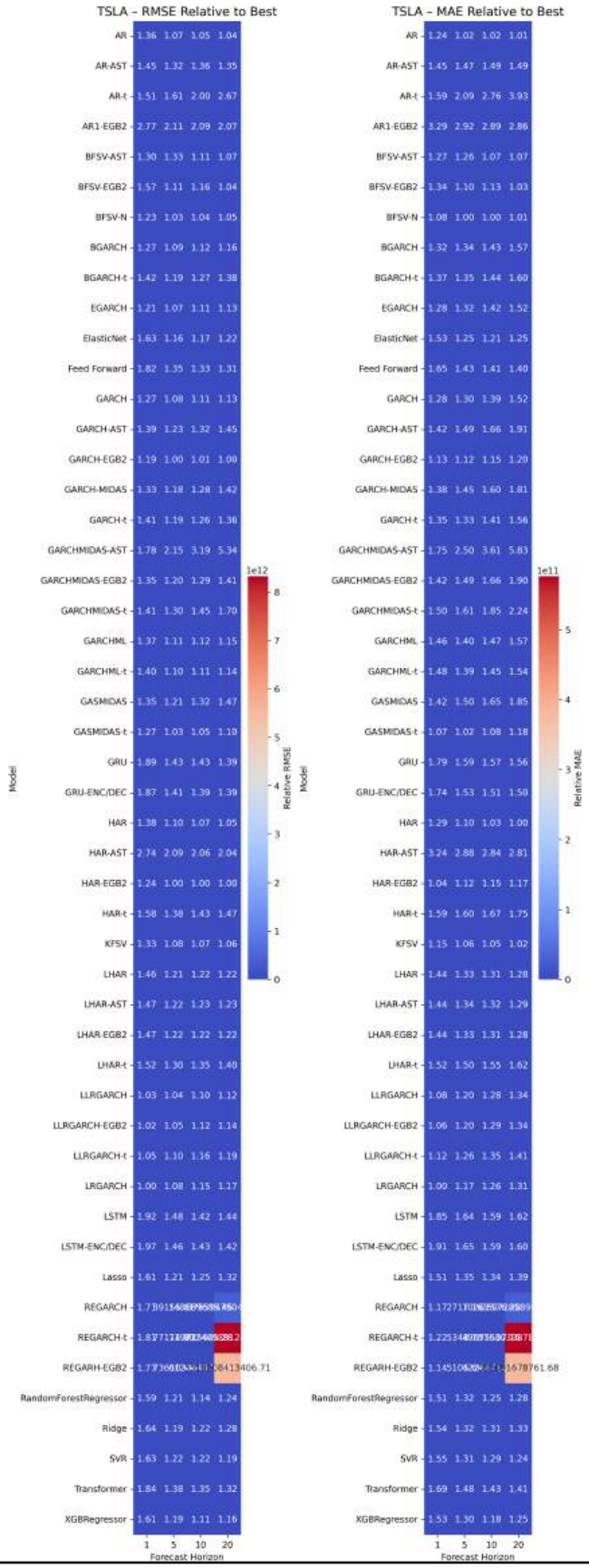


Figure E.34: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for Tesla—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


TSLA – CC Test FDR-adj. p-values
TSLA – IND Test FDR-adj. p-values
TSLA – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value
1
5
10
20
0.0
0.2
0.4
0.6
0.8
1.0
AR
AR-AST
AR-t
AR1-EGB2
BFSV-AST
BFSV-EGB2
BFSV-N
BGARCH
BGARCH-t
EGARCH
ElasticNet
Feed Forward
GARCH
GARCH-AST
GARCH-EGB2
GARCH-MIDAS
GARCH-t
GARCHMIDAS-AST
GARCHMIDAS-EGB2
GARCHMIDAS-t
GARCHML
GARCHML-t
GASMIDAS
GASMIDAS-t
GRU
GRU-ENC/DEC
HAR
HAR-AST
HAR-EGB2
HAR-t
KFSV
LHAR
LHAR-AST
LHAR-EGB2
LHAR-t
LLRGARCH
LLRGARCH-EGB2
LLRGARCH-t
LRGARCH
LSTM
LSTM-ENC/DEC
Lasso
REGARCH
REGARCH-t
REGARH-EGB2
RandomForestRegressor
Ridge
SVR
Transformer
XGBRegressor


TSLA – RMSE Relative to Best
TSLA – MAE Relative to Best
Model
Forecast Horizon
Relative RMSE
Relative MAE
1e12
1e11

E.11 TSCM

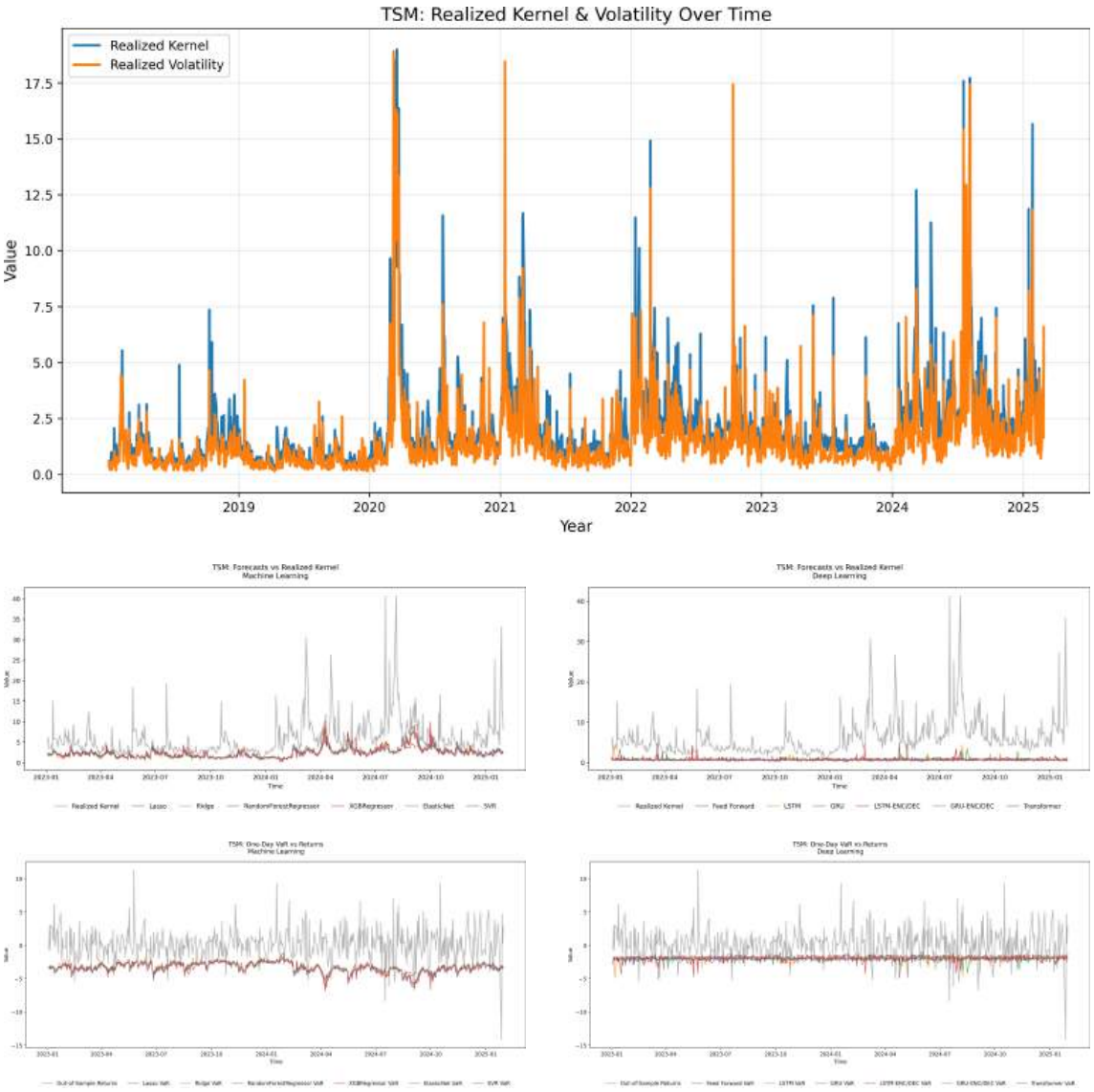
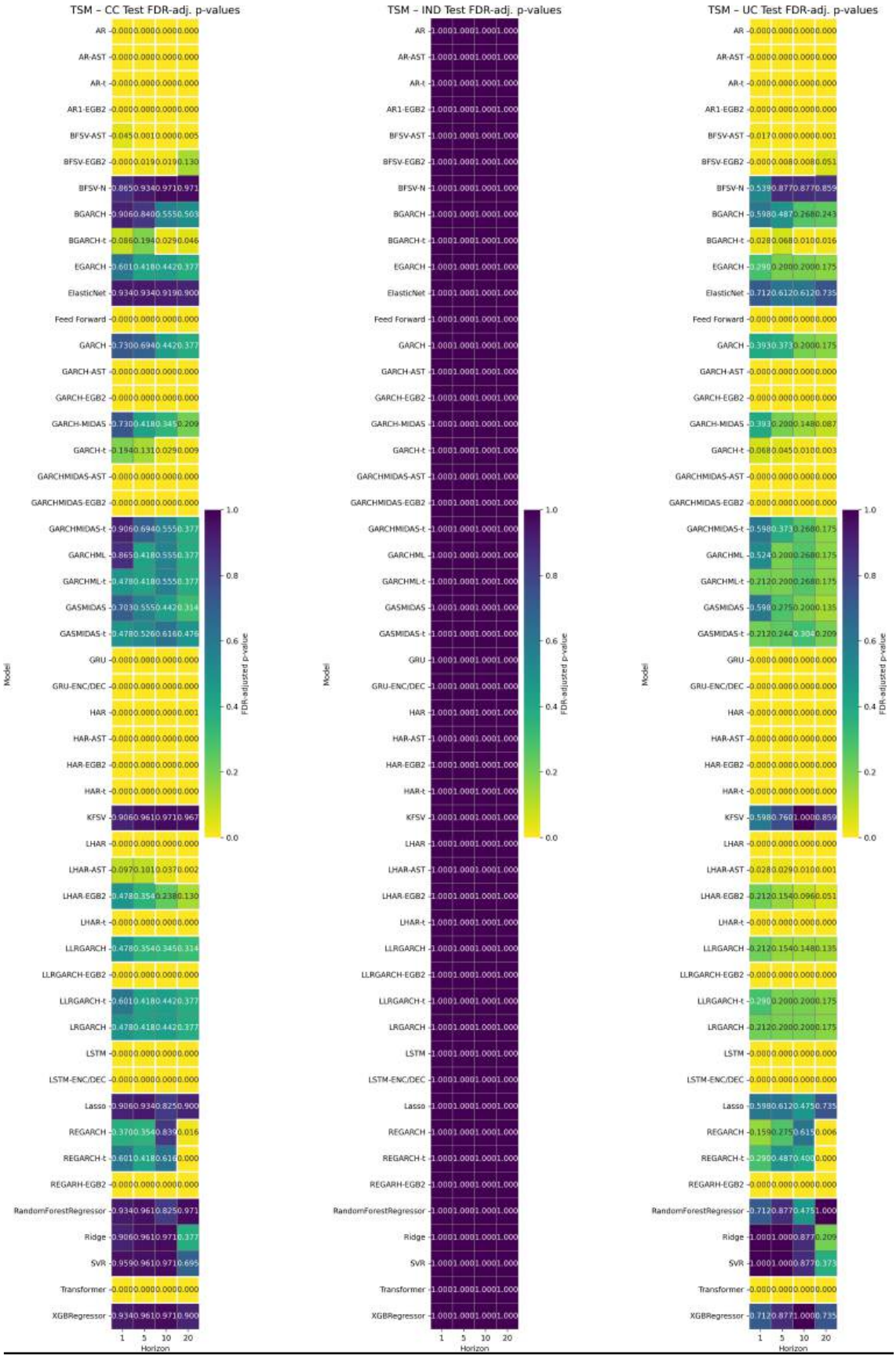
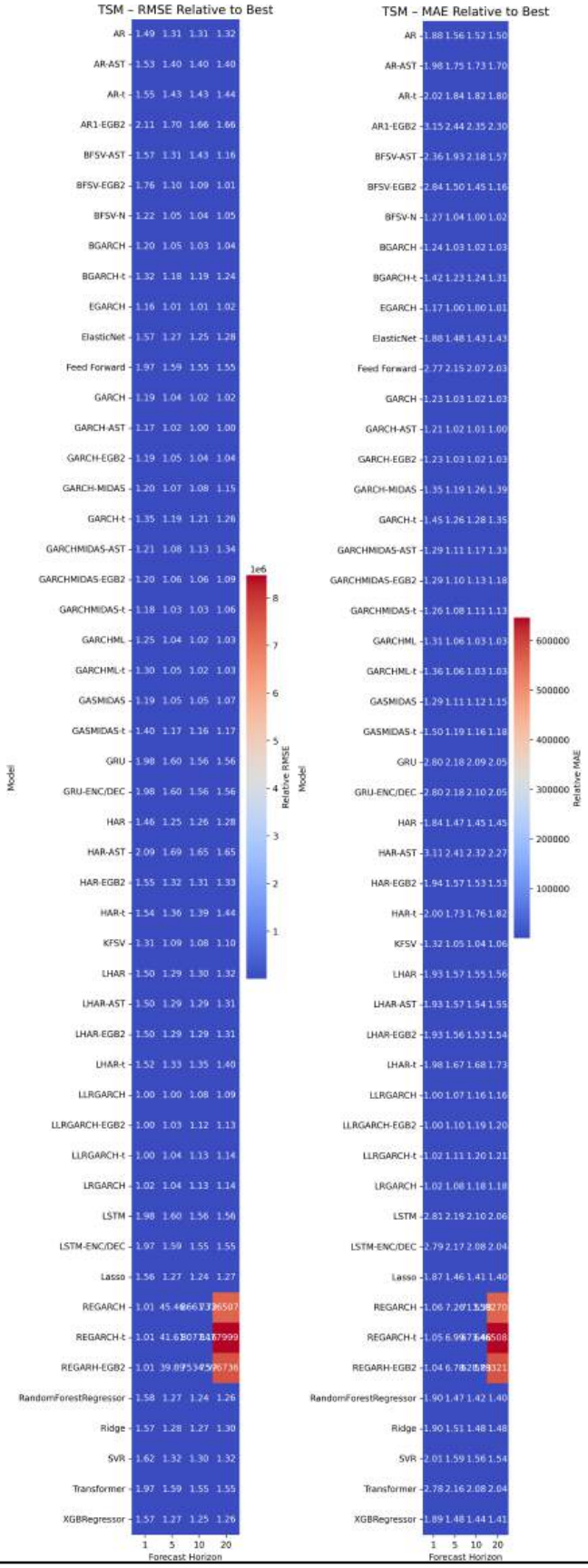


Figure E.37: Top: realized variance (Eq. (9)) vs. realized kernel (Sec. 4). Middle row: one-step-ahead forecasts for TSCM—machine-learning (left) vs. deep-learning (right). Bottom row: corresponding Value-at-Risk forecasts.


TSM – CC Test FDR-adj. p-values
TSM – IND Test FDR-adj. p-values
TSM – UC Test FDR-adj. p-values
Model
Horizon
FDR-adjusted p-value


TSM – RMSE Relative to Best
TSM – MAE Relative to Best
Model
Forecast Horizon
Relative RMSE
Relative MAE
1e6
AR
AR-AST
AR-t
AR1-EGB2
BFSV-AST
BFSV-EGB2
BFSV-N
BGARCH
BGARCH-t
EGARCH
ElasticNet
Feed Forward
GARCH
GARCH-AST
GARCH-EGB2
GARCH-MIDAS
GARCH-t
GARCHMIDAS-AST
GARCHMIDAS-EGB2
GARCHMIDAS-t
GARCHML
GARCHML-t
GASMIDAS
GASMIDAS-t
GRU
GRU-ENC/DEC
HAR
HAR-AST
HAR-EGB2
HAR-t
KFSV
LHAR
LHAR-AST
LHAR-EGB2
LHAR-t
LLRGARCH
LLRGARCH-EGB2
LLRGARCH-t
LRGARCH
LSTM
LSTM-ENC/DEC
Lasso
REGARCH
REGARCH-t
REGARH-EGB2
RandomForestRegressor
Ridge
SVR
Transformer
XGBRegressor

References

Hirotugu Akaike. Information theory and an extension of the maximum likelihood principle. *Proceedings of the 2nd International Symposium on Information Theory*, 1:267–281, 1973.

Torben G. Andersen, Tim Bollerslev, and Francis X. Diebold. Modeling and forecasting realized volatility. *Econometrica*, 71(2):579–625, 2003. doi: 10.1111/1468-0262.00418.

Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models, 2019. URL `https://arxiv.org/abs/1908.10063`.

Fabrizio Audrino and Paul Offner. Forecasting u.s. equity volatility with finbert-derived sentiment and attention mechanisms. *Quantitative Finance*, 20(5):743–760, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015. URL `https://arxiv.org/abs/1409.0473`.

Andrea Barbaglia, Marco Rossi, and Alan Taylor. Sentiment, machine learning, and realized volatility: An empirical comparison. *Journal of Forecasting*, 42(2):315–338, 2023.

Ole E Barndorff-Nielsen, P Reinhard Hansen, Asger Lunde, and Neil Shephard. Realized kernels in practice: Trades and quotes, 2009.

Ole E. Barndorff-Nielsen and Neil Shephard. Estimating quadratic variation using realised variance. *Journal of Financial Econometrics*, 1(1):1–32, 2002.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973. ISSN 00223808, 1537534X. URL `http://www.jstor.org/stable/1831029`.

S. T. Bodilsen and A. Lunde. Exploiting news analytics for volatility forecasting. *Journal of Applied Econometrics*, 40(1):18–36, 2025.

T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986a.

T. Bollerslev, A. J. Patton, and R. Quaedvlieg. Exploiting the errors: A simple approach for improved volatility forecasting. *Journal of Econometrics*, 192(1):1–18, 2016.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986b.

Jean-Charles de Borda. *Mémoire sur les 'elections au scrutin.* Histoire de l'Académie Royale des Sciences, 1781.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

A. Bucci. Realized volatility forecasting with neural networks. *Journal of Financial Econometrics*, 18(3):502–531, 2020.

Business Insider. Business insider nederland, 2025. URL `https://www.businessinsider.nl/?international=true&r=US`.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. pages 785–794, 2016.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. ACL, 2014.

K. Christensen, M. Siggaard, and B. Veliyev. A machine learning approach to volatility forecasting. *Journal of Financial Econometrics*, 21(5):1680–1727, 2023.

Peter F Christoffersen. Evaluating interval forecasts. *International economic review*, pages 841–862, 1998.

F. Corsi. A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2):174–196, 2009a.

Fulvio Corsi. A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2):174–196, 2009b.

Drew Creal, Siem Jan Koopman, and André Lucas. Generalized autoregressive score models with applications. *Journal of Applied Econometrics*, 28(5):777–795, 2013.

Stavros Degiannakis, Christos Floros, and Anastasia Livada. Forecasting volatility with score-driven models: An empirical comparison. *Journal of International Financial Markets, Institutions and Money*, 36:123–136, 2015.

David A. Dickey and Wayne A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366): 427–431, 1979.

Francis X Diebold and Robert S Mariano. Comparing predictive accuracy. *Journal of Business & economic statistics*, 20(1):134–144, 2002.

Alex Doe, Maria Rossi, and Chen Wang. On the limitations of self-attention in noisy time series. *Journal of Time Series Analysis*, 44(6):875–894, 2023. doi: 10.1111/jtsa.12456.

James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*, volume 38. OUP Oxford, 2012.

J. D. Díaz, E. Hansen, and G. Cabrera. Machine-learning stock market volatility: predictability, drivers, and economic value. *International Review of Financial Analysis*, 94: 103303, 2024.

R. F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of UK inflation. *Econometrica*, 50(4):987–1008, 1982.

R. F. Engle and V. K. Ng. Measuring and testing the impact of news on volatility. *Journal of Finance*, 48(5):1749–1778, 1993.

Robert F. Engle, Eric Ghysels, and Bumjean Sohn. Stock market volatility and macroeconomic fundamentals. *The Review of Economics and Statistics*, 95(3):776–797, 2013. doi: 10.1162/REST_a_00292.

Federal Reserve Bank of St. Louis. Federal Reserve Economic Data (FRED). `https://fred.stlouisfed.org`, 2025.

J. Frank. Forecasting realized volatility in turbulent times using temporal fusion transformers. *FAU Economics Discussion Paper No. 03/2023*, 2023. Erlangen-Nuremberg: FAU.

Carl Friedrich Gauss. Theoria motus corporum coelestum, 1809.

Benyamin Ghojogh and Ali Ghodsi. Attention mechanism, transformers, bert, and gpt: Tutorial and survey. *IEEE Access (preprint)*, 2020. arXiv:2012.00177.

Neha Goel, Derek Wang, and Suman Patel. Timesfm: A foundation transformer model for long-horizon variance forecasting. In *Proceedings of the 2024 Conference on Empirical Methods in Finance*, pages 102–114, 2024.

Fabio A. González. Backpropagation derivation, March 21 2018. Technical report.

Robert E. Goodin. Routledge handbook of public policy. 2005. Section on voting rules.

Paolo Gorgi, Peter R Hansen, Pawel Janus, and Siem J Koopman. Realized Wishart-GARCH: A score-driven multi-asset volatility model. *Journal of Financial Econometrics*, 17(1):1–32, 2019.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL `https://arxiv.org/abs/2312.00752`.

E. S. Gunnarsson, H. R. Isern, A. Kaloudis, M. Risstad, B. Vigdel, and S. Westgaard. Prediction of realized volatility and implied volatility indices using AI and machine learning: A review. *International Review of Financial Analysis*, 87:103221, 2024.

Deepak Gupta, Mahardhika Pratama, Zhifeng Ma, Jie Li, and Madasu Prasad. Financial time series forecasting using twin support vector regression. *PLoS ONE*, 14(3):e0211402, 2019. doi: 10.1371/journal.pone.0211402. URL `https://doi.org/10.1371/journal.pone.0211402`.

Peter R. Hansen, Asger Lunde, and James M. Nason. The model confidence set. *Econometrica*, 79(2):453–497, 2011a.

Peter Reinhard Hansen and Asger Lunde. Forecasting volatility using high frequency data. 2011. URL https://public.econ.duke.edu/~get/browse/courses/201/spr11/DOWNLOADS/VolatilityMeasures/SpecificlPapers/hansen_lunde_forecasting_rv_11.pdf. Section 6.3.1.

Peter Reinhard Hansen, Zhuo Huang, and Howard Howan Shek. Realized GARCH: a joint model for returns and realized measures of volatility. *Journal of Applied Econometrics*, 27(6):877–906, 2011b.

Andrew Harvey, Esther Ruiz, and Neil Shephard. Multivariate stochastic variance models. *The Review of Economic Studies*, 61(2):247–264, 1994.

Hossein Hassani, Eduardo Silva, and Luiz Moraes. Hybrid machine learning and econometric models for volatility forecasting. *International Journal of Forecasting*, 40(1): 150–167, 2024.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer, 2nd edition, 2009. ISBN 978-0-387-84857-0.

Trevor Hastie, Robert Tibshirani, Gareth James, and Jonathan Taylor. *An Introduction to Statistical Learning, With Applications in Python.* Springer, 2023.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Nikolaus Hautsch and Wing S. Hsieh. Realized jumps and realized garch: Forecasting volatility with jumps and microstructure noise. *Journal of Financial Econometrics*, 10 (4):601–639, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Ching-Lai Hwang and Kwangsun Yoon. *Methods for Multiple Attribute Decision Making.* Springer-Verlag, New York, NY, 1981.

IBM. What is lasso regression?, January 2024. URL https://www.ibm.com/think/topics/lasso-regression. [Online; accessed 29 June 2025].

Sungki Kim, Neil Shephard, and Siddhartha Chib. Stochastic volatility: Likelihood inference and comparison with arch models. *Review of Economic Studies*, 65(3):361–393, 1998.

Siem Jan Koopman, Borus Jungbacker, and Eugenie Hol. Forecasting daily variability of the s&p 100 stock index using historical, realised and implied volatility measurements. *Journal of Applied Econometrics*, 2005.

D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988. URL https://books.google.nl/books?id=4rKaGwAACAAJ.

B. Lei, Z. Liu, and Y. Song. On stock volatility forecasting based on text mining and deep learning under high-frequency data. *Journal of Forecasting*, 40(8):1596–1610, 2021.

Mingjie Li, Rui Liu, Guangsi Shi, Mingfei Han, Changling Li, Lina Yao, Xiaojun Chang, and Ling Chen. Mitigating data redundancy to revitalize transformer-based long-term time series forecasting system. *arXiv preprint arXiv:2207.07827v5*, 2022. doi: 10.48550/arXiv.2207.07827. URL https://arxiv.org/abs/2207.07827v5.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, 2015. URL https://arxiv.org/abs/1508.04025.

Pekka Malo, Ankur Sinha, Pyry Takala, Pekka Korhonen, and Jyrki Wallenius. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the American Society for Information Science and Technology*, 04 2014. doi: 10.1002/asi.23062.

Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. Machine learning advances for time series forecasting. *Journal of economic surveys*, 37(1):76–111, 2023.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, pages 14014–14024, 2019.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010.

Daniel B Nelson. Conditional heteroskedasticity in asset returns: A new approach. *Econometrica: Journal of the econometric society*, pages 347–370, 1991.

Harvard NLP. The annotated transformer. http://nlp.seas.harvard.edu/2018/04/03/attention.html, 2018.

NYSE Market Data. Monthly taq user guide v1.3. https://www.nyse.com/publicdocs/nyse/data/Monthly-TAQ-User-Guide-v1.3.pdf, 2011.

Sungwoo Park, Seungmin Jung, Seungwon Jung, Seungmin Rho, and Eenjun Hwang. Sliding window-based lightgbm model for electric load forecasting using anomaly repair. *The Journal of Supercomputing*, 77:1–22, 11 2021. doi: 10.1007/s11227-021-03787-4.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL http://jmlr.org/papers/v12/pedregosa11a.html.

Fotios Petropoulos, Yanfei Kang, Feng Li, and et al. Forecasting: theory and practice. *International Journal of Forecasting (online encyclopedia)*, 2023. URL `https://forecasting-encyclopedia.com/theory.html`. Updated May 5, 2023; comprehensive review of forecasting theory and practice.

Dimitris N. Politis and Joseph P. Romano. The stationary bootstrap. *Journal of the American Statistical Association*, 89(428):1303–1313, 1994.

S.-H. Poon and C. W. J. Granger. Forecasting volatility in financial markets: A review. *Journal of Economic Literature*, 41(2):478–539, 2003.

Z. Qiu, C. Kownatzki, F. Scalzo, and E. S. Cha. Historical perspectives in volatility forecasting methods with machine learning. *Risks*, 13(5):98, 2025.

E. Rahimikia and S.-H. Poon. Big data approach to realized volatility forecasting using har model augmented with limit order book and news. *SSRN Working Paper No. 3684040*, 2020.

Alberto Ramos-Pérez, Jonathan Smith, and Karen Lee. Multi-transformer models for one-day-ahead s&p 500 realized volatility. *Journal of Financial Econometrics*, 19(3): 567–589, 2021.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978.

C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Dustin Stansbury. Derivation: Error backpropagation & gradient descent for neural networks. `https://dustinstansbury.github.io/theclevermachine/derivation-backpropagation`, June 29 2020.

Cole Stryker and Dave Bergmann. What is a transformer model? IBM Think, Mar 2025. URL `https://www.ibm.com/think/topics/transformer-model`.

Student. The Probable Error of a Mean. *Biometrika*, 6(1):1–25, 1908.

Nasos Thanasoulas. MTHEC RU master thesis. Erasmus School of Economics, student ID: s1002576, 2023.

Shuairu Tian and Shigeyuki Hamori. Modeling interest rate volatility: A realized garch approach. *Journal of Banking & Finance*, 61:158–171, 2015. doi: 10.1016/j.jbankfin.2015.09.008.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Allan Timmermann and Yinchu Zhu. Comparing forecasting performance with panel data. April 29 2019. Working paper.

Wesley van Driel. Breaking down stock market volatility: A component model analysis. July 2019.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995. ISBN 978-0-387-98780-4.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017. URL `https://arxiv.org/abs/1706.03762`.

Waloddi Weibull. A statistical theory of strength of materials. *IVB-Handl.*, 1939.

S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang. Adversarial sparse transformer for time series forecasting. In *Advances in Neural Information Processing Systems*, NeurIPS '20, 2020.

Liang Xiao and Wenhu Zhang. Performance of the realized-garch model against other garch-based models in cryptocurrency volatility forecasting. *Risks*, 11(12):211, 2023.

Z. Yang, W. Yan, X. Huang, and L. Mei. Adaptive temporal-frequency network for time-series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(4):1576–1587, 2022. doi: 10.1109/TKDE.2020.3003420. URL `https://doi.org/10.1109/TKDE.2020.3003420`.

G. Udny Yule. On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A*, 226:267–298, 1927.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. `https://D2L.ai`.

Dongming Zhu and John W. Galbraith. A generalized asymmetric student-t distribution with application to financial econometrics. *Journal of Econometrics*, 157(2):297–305, August 2010a. doi: 10.1016/j.jeconom.2010.01.013.

Dongming Zhu and John W Galbraith. A generalized asymmetric student-t distribution with application to financial econometrics. *Journal of Econometrics*, 157(2):297–305, 2010b.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Walter Zucchini, L. MacDonald, Ian and Roland Langrock. *Hidden Markov Models for Time Series: An Introduction Using R*. Chapman & Hall/CRC, 2016.