CATÓLICA LISBON
BUSINESS & ECONOMICS

Market Volatility: Can Machine Learning Methods Enhance Volatility Forecasting?

Afonso Batista

Dissertation written under the supervision of Professor José Faias

Dissertation submitted in partial fulfilment of requirements for the MSc in Finance, at Universidade Católica Portuguesa, April 2023.

Abstract

This dissertation aims to test whether the use of machine learning (ML) techniques can improve volatility forecasting accuracy. More specifically, if it can beat the best econometric model, the Heterogeneous Autoregressive model of Realized Volatility (HAR-RV). Using S&P 500 Index data from May-2007 to August-2022, the superiority of the HAR-RV was tested and attested against competing econometric models EWMA and GARCH(1,1). Next, the performance of the ML Artificial Neural Network algorithms Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are compared to the performance of the econometric models. Five different variable sets are tested for the ML models. It is found that while both ML models are able to beat the EWMA and GARCH(1,1) models by a significant margin, the HAR-RV model still outperforms LSTM and GRU.

Moreover, an analysis is conduced on the models' predictions on the period corresponding to the Covid-19 crisis. The results did not show any evidence suggesting that ML methods have a particular advantage at predicting during high volatility events.

Finally, a plausible cause that could undermine the remarkable qualities of the ML methods in the aim of volatility forecasting is discussed. It is found that the rigorous set of conditions needed to be met for the proper setup of ML models are very difficult to be met using financial data, which hinders the aptitude of ML for this purpose.

Keywords: Volatility Forecasting; Heterogeneous AutoRegressive model; Machine Learning; Artificial Neural Networks; Long Short-Term Memory; Gated Recurrent Unit.

Title: Market Volatility: Can Machine Learning Methods Enhance Volatility Forecasting?

Author: Afonso Maria Nabeto Valentim Xavier Batista

Resumo

Esta tese visa testar se o uso de técnicas de Machine Learning (ML) pode melhorar a precisão da previsão da volatilidade. Mais especificamente, se estes algoritmos conseguem superar o melhor modelo econométrico, o Heterogeneous Autoregressive model of Realized Volatility (HAR-RV). Usando dados do Índice S&P 500 de Maio-2007 a Agosto-2022, a superioridade do HAR-RV perante os modelos econométricos concorrentes EWMA e GARCH(1,1), foi testada e confirmada. Em seguida, o desempenho dos algoritmos ML de redes neurais artificiais de Long Short-Term Memory (LSTM) e Gated Recurrent Unit (GRU) são comparados com o desempenho dos modelos econométricos tradicionais. Cinco conjuntos diferentes de variáveis são testados para os modelos ML. Verifica-se que enquanto ambos os modelos ML são capazes de superar os modelos EWMA e GARCH(1,1) por uma margem significante, o modelo HAR-RV ainda tem um desempenho superior ao LSTM e ao GRU.

É ainda feita uma análise das previsões dos modelos durante o período correspondente à crise do Covid-19. Os resultados não mostram qualquer evidência que sugira que os métodos ML têm uma particular vantagem durante eventos de alta volatilidade.

Finalmente, é discutida uma possível causa que poderá debilitar as sofisticadas qualidades dos métodos ML para a finalidade de previsão de volatilidade. Verifica-se que o conjunto rigoroso de condições necessárias para a correcta configuração dos modelos ML é muito difícil de se cumprir utilizando series temporais de volatilidade de mercado, o que prejudica a aptidão dos modelos ML para esta finalidade.

Palavras-chave: Previsão de Volatilidade; Modelos Heterogéneos AutoRegressivos; Machine Learning; Redes Neurais Artificiais; Long Short-Term Memory; Gated Recurrent Unit.

Título: Market Volatility: Can Machine Learning Methods Enhance Volatility Forecasting?

Autor: Afonso Maria Nabeto Valentim Xavier Batista

Table of Contents

List of Figures

List of Tables

1. Introduction

Volatility is a topic of major relevance in the financial world. It bears a great importance in many fields in finance such as investments, risk management, security valuation and monetary policy making (Poon, 2003).

Volatility is not synonym to risk however, it can be interpreted as uncertainty. As noted by Campbell et al. (1997), uncertainty is what distinguishes financial economics since, in its absence, all financial economics problems would be reduced to basic microeconomics. Interpreted as uncertainty, volatility is a key component to evaluate investments and manage portfolios. In fact, volatility is at the heart of Markowitz's Modern Portfolio Theory. Portfolios are balanced for set levels of risk, in which for higher uncertainty (volatility), a higher return is expected. Following a similar rationale, volatility is fundamental for the pricing of many financial instruments in which the uncertainty of future values is accounted for, such as option contracts.

In the past decades, the regulatory landscape on financial institutions has tightened and, nowadays, very strict regulations have to be followed. In particular, there are important regulations imposed on capital reserves that lending institutions need to hold. These reserves regulations are most often based on a metric that is nowadays a mainstay of risk exposure control, called Value at Risk (VaR). This metric measures the possible financial loss of value of a product over a period of time, and uses volatility as its main computation input.

Highly volatile financial market periods can spill over to the real economy, resulting in major impacts in people's lives. For this reason, market volatility indexes are of interest to governments and policy makers, to act as a barometer of financial markets vulnerability and support monetary policy decisions.

Given its importance in all these areas of the financial sector and relevance for the economy, it is clear that volatility forecasting is a very valuable and influential topic of research. An accurate volatility forecast adds value to many agents, institutions and governments. Furthermore, even though extensive literature has proved stock market returns to be extremely difficult to forecast, volatility, on the other hand, has showed to be rather predictable, to some extent. This makes the research of the topic more worthwhile, as there is a higher change of achieving more meaningful and impactful progress.

So far, great progress has been made in the development of econometric models to forecast volatility. However, econometric methods tend to struggle with nonlinearity in data,

which is present in volatility timeseries. Machine learning methods, that have been becoming increasingly more prominent in financial applications, have the characteristic of exceling at assimilating and modelling nonlinear relationships. Given machine learning techniques seemingly great abilities and high rate of innovation and improvements, these constitute a great prospect that is worthy of exploring as they could lead to an immense advance on the quest for the best volatility forecasting model. Ergo, in the most recent decades some machine learning approaches have been proposed to forecast volatility.

In this context, the main objective of this research is to ascertain whether machine learning models can beat the current best econometric models, by making a thorough analysis of best prospective ML models for volatility forecasting and comparing its features and performance to the best and most famous traditional econometric approaches. This work sought to answer the following research questions:

1) Can machine learning methods achieve high accuracy in forecasting volatility?
2) Can machine learning methods outperform the best traditional econometric approaches?
3) Can machine learning methods better predict and forecast volatility in times of crisis?

Firstly, to infer on econometric model's performance and confirm which is the best one, the HAR-RV, EWMA and GARCH(1,1) are tested. Using S&P 500 Index data from May-2007 to August-2022 and a one year rolling window, we confirm that, as expected from the literature (Padovani et al., 2016), HAR-RV is the far superior model with an out-of-sample RSME of 3,97%.

Advancing for the ML models, the study focused on the Artificial Neural Networks algorithms: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). Using the same dataset, the two models are setup using a 5-fold time-series cross-validation technique to split the dataset, and a grid search for hyperparameter optimization. Given ML models' ability of taking in multiple inputs, multiple exogenous variables were tested to improve performance, resulting in five different input sets, including variables such as daily log returns, daily RV, weekly and monthly aggregated RV, VIX and the HAR-RV prediction. The best performing variable set, which was the kitchen sink including all the mentioned variables, scored an out-of-sample RMSE of 5,37% and 5,41% for GRU and LSTM, respectively. These results, though clearly better than GARCH(1,1) and EWMA, are inferior to HAR-RV. The two machine learning algorithms showed to be competent at forecasting volatility, achieving results that are

up to par with other traditional techniques. However, they were not able to beat the best econometric model, the HAR-RV.

Machine leaning models are famously known for their ability to capture complex non-linear relationships and uncover unknown trends in data. To test if these methods are indeed able to uncover any hidden patterns that precede high volatility periods, the performances of the models are analysed during the Covid-19 pandemic crisis period, from the 1st of February 2020 to the 1st of May 2020. The results for this period show a similar performance hierarchy as before. No evidence is found that suggests that ML methods have a particular advantage during high volatility events.

This paper is structured into four main sections. First, Chapter 2, focuses on establishing a foundation about volatility: its definition, measures and stylized facts. After that, Chapter 3, has a literature review of the volatility forecasting methods using econometric approaches, an explanation of the theory behind the HAR-RV and two competing econometric models, and an application of the models to test and attest HAR-RV's superiority. Chapter 4 addresses the literature of machine learning approaches, presents the mechanisms of the LSTM and GRU algorithms, establishes the experimental setup and tests the use of the two ML models for volatility forecasting. The results obtained are presented and discussed, comparing them to those of HAR-RV. Lastly, Chapter 5 concludes on the main findings, answers the three research questions proposed and addresses possible limitations and suggestions for further research.

2. Volatility

2.1 Defining and Measuring Volatility

Volatility, usually denoted by σ, is a statistical measure of dispersion of a security's returns from their mean. It expresses the rate at which a security's price rises or falls for a given set of returns. There are three types of volatility: historical, realized and implied.

In finance, the simplest measure is **historical volatility**, which is measured by the standard deviation of logarithmic returns. For a given set of observations,

$$\sigma = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(r_t - \mu)^2} \tag{2.1}$$

where $r_t = \log\left(\frac{S_t}{S_{t-1}}\right)$ is the log returns at time t, S_t is the price at time t, μ is the mean return for the set of observations, and N is the number of observations. The standard deviation is a meaningful measure of dispersion only if returns follow a gaussian distribution, hence log returns are used instead of daily close prices, as the later do not meet this requirement (Andersen et al., 2001). When computing the historical volatility of a population from a sample, it is important to account for the bias due to finite sample size. Bessel's correction factor helps reducing this bias and can be introduced in equation (2.1) by applying the multiplicative factor N/(N-1) inside the square root, resulting in

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}(r_t - \mu)^2}\,. \tag{2.2}$$

However, even though the Bessel's correction allows for an unbiased estimation of variance (σ^2) it cannot not fully correct its square root, which results in a still biased (although less) estimate of volatility (σ), especially for small samples.

To compute the historical volatility, there is the need of calculating the mean returns. The current process, though unbiased, can be result in very noisy estimates, especially for smaller samples. Moreover, the occurrence of negative average returns goes against previously set non-negativity constraints (Merton, 1980). For these reasons, the realized volatility

technique, which sets the mean returns to zero, is the most often the preferred method of estimating volatility in the literature.

Theoretically, volatility is an instantaneous variable and therefore should be treated in a continuous-time diffusive setting. In such setting, over a given interval [0, T], we have

$$ds(t) = \mu(t)dt + \sigma(t)dW(t) \qquad ,0 \leq t \leq T \tag{2.3}$$

where $s(t)$ is the logarithmic asset price at time t, $\mu(t)$ is the drift term, W is a standard Brownian motion process, σ(t) is the instantaneous volatility that inflates the change in price relative to dW(t). The terms μ(t) and σ(t) can also be interpreted as the instantaneous conditional mean and volatility of the return. The continuously compounded return (r) over the interval $[t - k,\ t]$, with $0 < k \leq t$, is therefore

$$r(t,k) = s(t) - s(t - k) = \int_{t-k}^{t} \mu(\tau)d\tau + \int_{t-k}^{t} \mu(\tau)dW(\tau)\ . \tag{2.4}$$

The best way of modelling volatility would be an integral over time, which would result in the *integrated volatility* however, due to the discrete nature of the financial system this is not possible. Instead, a good approximation can be achieved using quadratic variation (QV) theory, which following the previous setting and applying it to the returns defined in (2.4), we have

$$QV\ (t,k) = \int_{t-k}^{t} \sigma^2(\tau)\, d\tau \tag{2.5}$$

In order to make this applicable to the financial market, the interval $[t - k,\ t]$ is partitioned in the discrete series $\left\{t - k + \frac{j}{n},\ with\ j = 1,\ \ldots\ n \cdot k\right\}$. Then the **realized volatility** (RV) is

$$RV\ (t,k;\ n) = \sqrt{\sum_{j=1}^{n \cdot k} r^2\left(t - k + \frac{j}{n}, \frac{1}{n}\right)} \tag{2.6}$$

Because semimartingale theory applies, the square of the realized volatility measure converges in probability to the return quadratic variation QV when the sampling frequency n increases:

$$RV^2(t,\delta;n) \to QV(t,\delta), \qquad as\ n \to \infty \tag{2.7}$$

A very high sampling frequency allows for the better approximation of the continuous price over time, however with high frequency the data is very noisy due to many market microstructures. Hence, when choosing sampling frequency, there is a trade-off between continuous prices and less noisy data. In practice, the choice of n is many times influenced and restricted by the liquidity of the market and availability of data, however literature suggests 5 to 15-minute intervals to achieve the best balance (Andersen et al., 2001). The huge improvement in data access has had a big impact on the volatility measurement literature. The availability of high-frequency data for most financial products has led the realized volatility measure to be one of the most prevalent measures of volatility in the literature, as it produces a very accurate proxy of actual volatility using a model-free approach (Barndorff-Nielsen et al., 2002).

One different approach to volatility is **implied volatility** (IV). While the aforementioned approaches look to measure the volatility that happened in a past period of time given the security's price history, the implied volatility looks to find the market outlook of future volatility. Implied volatility uses the options pricing model Black-Scholes to calculate backwards the volatility that is implied in an options contract price. This volatility can be considered an efficient expectation of future volatility by the market given the fact that these option prices are set by the market, which in turn operates according to the efficient market hypothesis.

2.2 Volatility Stylized Facts

To understand what makes up a good volatility forecasting model, one should first understand very well volatility and its behaviour. Volatility, as a measure of dispersion of a security's market returns, is very hard to predict given the uncertainty and unpredictability linked with the stock market. Although it is very hard to fully comprehend, there are some tendencies that can be detected and used to summarize broad characteristics of volatility behaviour. These tendencies are often called stylized facts. Extensive literature has proved stock market returns to be extremely difficult to forecast however, on the other hand, volatility has showed to have characteristics that are predictable to some extent. Many stylized facts have been discovered and make it reasonable to believe it is possible to model volatility's behavioural dynamics. The graphs, tables and calculations used to illustrate and exemplify the stylized facts below, are built using S&P 500 daily realized volatility data from 30-04-2007 to 12-08-2022.

First of all, volatility is not constant. This is a long-known fact with much evidence of its veracity, such as Fama (1965), Officer (1973), among others. Though it is not constant, it displays a clear mean-reverting behaviour (Fouque et al. 2000). It can be seen that volatility always tends to slowly go back to its mean. Even when it suffers large shocks that make it rise to abnormally high values, in the long run volatility always reverts back to its mean. Figure 2.1 below shows the one-month evolution of the daily volatility mean of the observations that are on the first and tenth deciles of volatility, as well as its mean, where the mean-reverting behaviour is clearly seen.

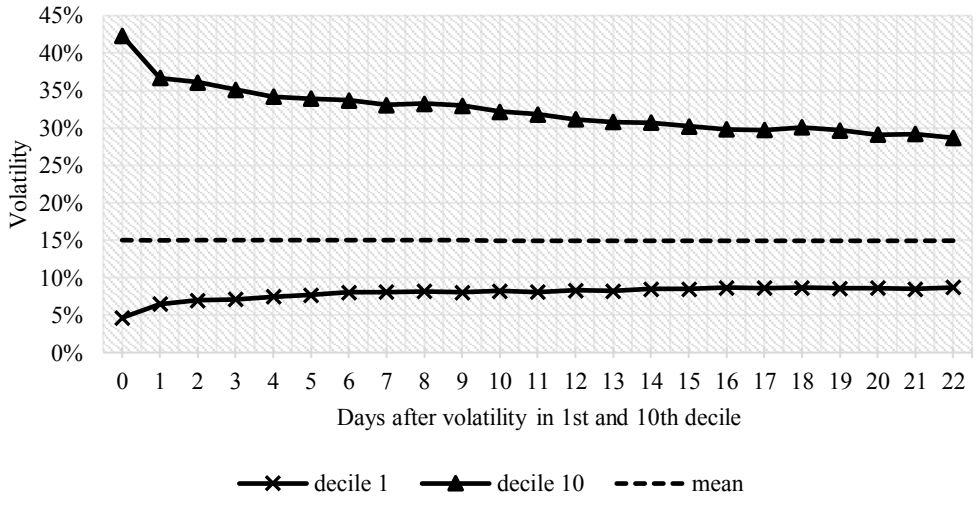

Figure 2.2.1 - Volatility mean-reverting behaviour graph

Another stylized fact is that volatility tends to cluster. This fact was first noted by Mandelbrot (1963) and it refers to the fact that high volatility is likely to be followed by high volatility and low volatility is likely to be followed by low volatility.

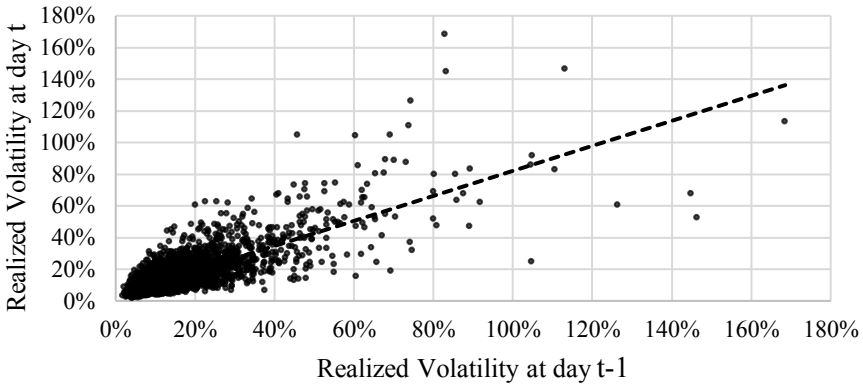

Figure 2.2 - Volatility Clustering behaviour graph.

Moreover, volatility exhibits serial autocorrelation (LeBaron, 1992) which is in line with volatility clustering. This autocorrelation decays for farther lags, but still holds for very long lags, thus the reason why volatility is said to have long-term memory. Because of this characteristic, shocks in volatility tend to have very long-lasting impact.

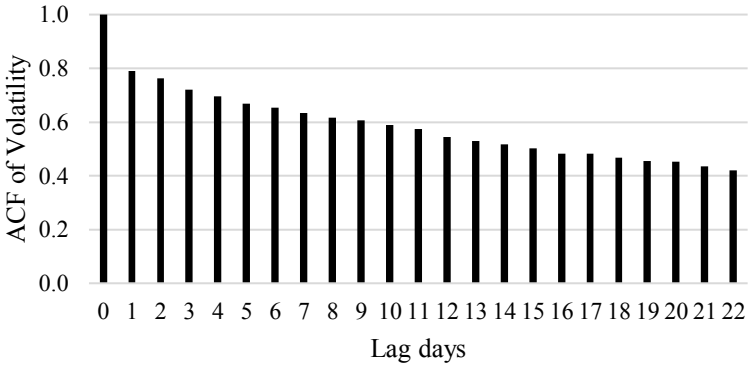

Figure 2.3 – Realized Volatility Correlogram.

Analysing the volatility distribution and comparing it with a normal distribution, one can see that it is heavily skewed to the right, as it can easily be observed in Figure 2.4. This

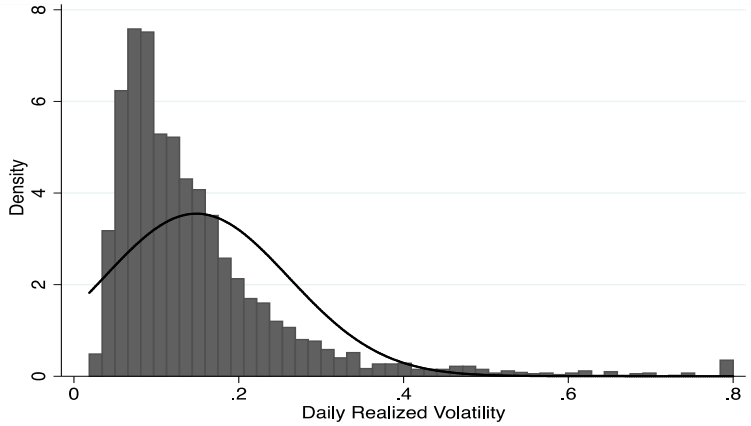

Figure 2.4 - Daily realized volatility distribution.
Obs.: for visualization purposes, values above 0.8 were grouped in the same bin.

means that periods of high volatility are more likely than what would be expected from a normal distribution.

Another stylized fact, that is often times challenging for models to capture, is the Leverage Effect. This effect refers to the tendency of a volatility to be negatively correlated with the underlying asset's returns. This effect, also known as the asymmetric volatility phenomenon, was first studied by Black (1976) which suggested that the explanation for this behaviour was the fact that when the market price of a firm decreases, the firm's relative value of debt increases relative to its equity and, because the company becomes more leveraged, it is perceived as risker and thus is more volatile. This relationship is in fact asymmetric, everything else constant, when returns are negative, volatility increases rapidly, but when returns are positive, volatility decreases but to a much lesser extent. Overall, negative shocks tend to have a bigger impact on conditional volatility than positive ones.

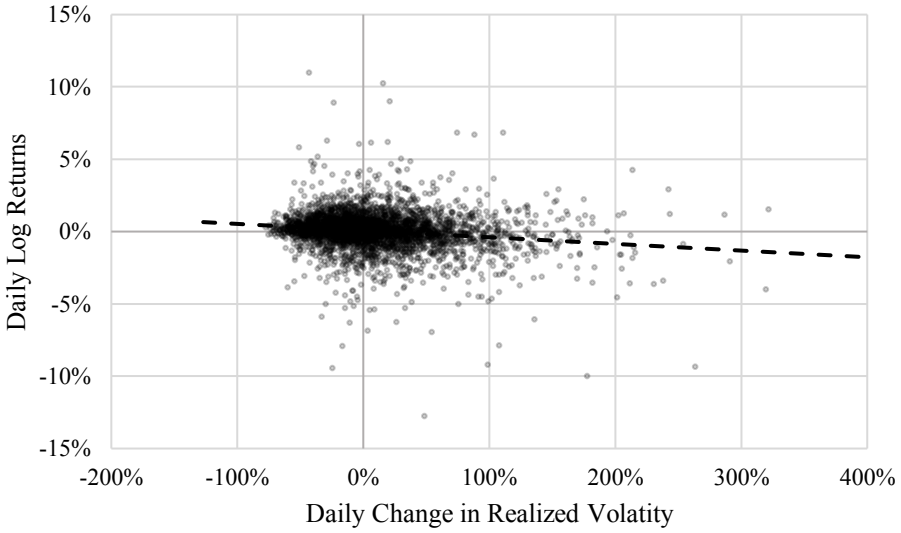

Figure 2.5 – Negative correlation between realized volatility and returns.

Lastly, it has also been shown that volatility measured at different frequencies, has different information content. Müller et al. (1997) discovered that the "diversity of agents in a heterogeneous market makes volatilities of different time resolutions behave differently". Changes in low-frequency volatility have a higher impact in subsequent high-frequency volatility than vice-versa. The rationale is that some market participants have different investment horizons, but regardless if they are short, medium or long-term, they are all influenced by the long-term market movements.

3. Econometric Models

3.1. Early Econometric Approaches Literature Review

Given the importance of volatility in financial markets, literature on volatility forecasting dates back to over half a century ago. In the last few decades, the topic of volatility has been one of the most active in the economic forecasting field and has achieved great progress in this period. Many approaches and tentative models have been put forward to formulate volatility forecasts using historical information or even option contracts information. Before machine learning was ever introduced, econometric models were proposed using both parametric and non-parametric methods. In this literature review, non-parametric methods will not be considered as they have been proved to perform poorly out-of-sample (Pagan et al., 1990).

Time series models make up the bulk of the volatility forecasting literature. Pure time series models are not based on economic theory, instead they seek to capture the stock returns volatility features observed in the actual market. The most rudimentary form of a time series model is the Random Walk model. According to this model, in every period, volatility takes a random step away from its previous value. These steps are independently and identically distributed in size. However, this model is obviously very deficient at taking into account any of the known stylized facts. Expanding on the idea of using historical volatility to predict future volatility, there is the Historical Average (HA) method which always predicts volatility in the next period to be the average of all of the historical data available. An upgrade of the previous, is the simple Moving Average (MA), which takes the average of a set rolling window behind the current value of volatility. This makes the estimate more current and relevant by disposing of the older values and reflects a trade-off between using more data in the estimate and using data closer to time t. To make recent values more prominent, one can use an exponential function to place greater weights on the more recent volatility values. Following this exponential logic, similarly to the HA and MA methods, there is the Exponential Smoothing method (ES), which takes all historical data, and the Exponential Weighted Moving Average (EWMA), which uses a rolling window. The later model (EWMA) was made popular by the famous J.P. Morgan's set of techniques to measure market risks - RiskMetrics^{TM} model (1996).

Another branch of methods to model volatility are the autoregressive (AR) models, which use a simple regression to express volatility as a function of past volatility and an error term. If the autoregressive model also includes past volatility errors, then it is an Autoregressive

Moving Average (ARMA) model. This is, ARMA tries to model how unexpected external factors that caused past impacts can still be affecting the current volatility. A significant downfall of these models is that they do not take under consideration that volatility has long term memory. A good way of caring for the long-term memory is the use of an integrated process (Nielsen et al., 2005). In 1976, Box and Jenkins published the Autoregressive Integrated Moving Average (ARIMA) model for the first time which added an Integrated factor, i.e. a differencing factor, to the ARMA model. The basic ARIMA has a differencing factor of order one, if the factor is of order smaller than one it is an Autoregressive Fractionally Integrated Moving Average (ARFIMA) model (Granger et al., 1980).

A more sophisticated family of time series models offered a new toolset to the volatility forecast issue. Proposed by Engle, in 1982, the Autoregressive Conditional Heteroskedasticity (ARCH) model can capture the volatility clustering and long-term mean reversion characteristics. The ARCH model, does not use past standard deviations, instead it formulates conditional variance as a linear function of the past squared errors, with a set number of lags. This model, however, lacks in modelling the persistence of volatility shocks and usually presents very "bursty" results. To address this shortfall, the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) (Bollerslev, 1986) was developed, in which conditional variance does not only depend on squared errors but also on a set number of lags of past conditional variance. The GARCH model is then able to account for long term memory, volatility clustering and fat tails returns.

The GARCH model is an influential landmark method and inspired many other variations and adaptations in a quest to overcome weaknesses and make improvements on the GARCH model. One of these GARCH-family models is the Exponential GARCH (EGARCH) model (Nelson, 1991). This model formulates conditional variance logarithmically and thus ensures that volatility is never negative without the need of imposing estimation constraints. Furthermore, EGARCH captures the size and sign of past residuals and because of that is able to model the "leverage effect" stylized fact. Other approaches also seek to incorporate the leverage effect in the model, one of them is the GJR-GARCH (Glosten et al., 1993), which uses a dummy variable to incorporate the information on whether shocks are positive or negative. Using a similar approach to the GJR-GARCH, the Threshold GARCH (TGARCH) (Zakoian, 1994) also models the leverage effect, tough it uses standard deviations instead of the variance. The GARCH model's volatility persistence is also not perfect, this is because the autocorrelation of the conditional variance is exponentially decaying, unlike the serial correlations of securities' realized volatility that have a very slow decay. To improve

GARCH's long-term memory, the Integrated GARCH (IGARCH) model (Engle et al., 1986) and subsequently the Fractionally Integrated GARCH (FIGARCH) (Bailie et al., 1996) models were proposed. Similar to what ARIMA and ARFIMA had done with and ARMA, these models added an Integrated factor to ARCH which represented volatility persistence more accurately. However, it has been showed that a positive Integrated process results in a positive drift which is not true in reality (Hwang et al., 1998), which is a big theoretical shortcoming of the FIGARCH. The GARCH-type models' family is very numerous. In order to see if the all the extensions of GARCH were useful, Hansen and Lund compared 330 ARCH-type models in terms of their performance at modelling conditional variance (Hansen et al., 2005). They found that, to forecast exchange rates volatility, no model outperformed the simple GARCH(1,1), but that using an equity security returns (IBM stock) the GARCH(1,1) was outperformed by models that could accommodate for the leverage effect.

Long-memory volatility models usually use fractional integration to parsimoniously achieve long memory. However, though it is a clever mathematical technique, fractional integration lacks economical interpretability and is thought to create an artificial mixing of the long and short-term volatility features (Comte et al., 1998). Inspired by the asymmetric propagation of volatility in short and long horizons, Corsi proposed the Heterogeneous Autoregressive model of Realized Volatility (HAR-RV) (Corsi, 2009). The HAR-RV is an additive cascade model that combines different volatility components at different frequencies with the aim of capturing the different types of market participants. The model, that results in a simple AR-type model, performs incredibly well at modelling volatility persistence and other known volatility stylized facts, even though it is not formally a long-memory model (Corsi, 2009). Nowadays, the HAR-RV model is often regarded as the best econometric models for realized volatility forecasting.

3.2. Heterogeneous Autoregressive for Realized Volatility (HAR-RV)

The Heterogeneous Autoregressive for Realized Volatility (Corsi, 2009) model builds on the heterogeneous market hypothesis and asymmetric propagation of volatility in short and long horizons. To capture the different types of market participants, the model combines realized volatility at three different frequencies, i.e., is computed using three different time horizons: daily, weekly (5 trading days) and monthly (22 trading days). The main formulation of the HAR-RV model can be expressed as

$$RV_t^{(d)} = c + \beta^{(d)} RV_{t-1}^{(d)} + \beta^{(w)} RV_{t-1}^{(w)} + \beta^{(m)} RV_{t-1}^{(m)} + u^{(d)} \tag{3.1}$$

where $u^{(d)}$ accounts for estimation errors and latent volatility, $RV_t^{(d)}$ is the daily realized volatility being forecasted and $RV_{t-1}^{(d)}$, $RV_{t-1}^{(w)}$, $RV_{t-1}^{(m)}$ are the last day, week and month's realized volatility, respectively. Given its simplicity, this model can easily be estimated by an Ordinary Least Squares (OLS) regression. In order to allow direct comparison among the volatilities defined over various time horizons, the $RV_{t-1}^{(w)}$ and $RV_{t-1}^{(m)}$ are normalized as one-period realized volatilities by a simple average of the daily volatilities expressed as

$$RV_t^{(w)} = \frac{1}{5}\left(\sum_{i=1}^{5} RV_{t-i}^{(d)}\right) \tag{3.2}$$

$$RV_t^{(m)} = \frac{1}{22}\left(\sum_{i=1}^{22} RV_{t-i}^{(d)}\right) \tag{3.3}$$

Therefore, to forecast the one day ahead volatility, the HAR-RV model needs the last 22 days' daily realized volatilities.

The HAR-RV model, with the use of the three different time horizons, is able to reproduce the volatility's long memory, fat tails and time persistence characteristics, even though it is not a true long-memory model. Also, it is able to do it in a very simple and parsimonious way while, at the same time, achieving an exceptionally good forecasting performance.

3.3. Competing Econometric Models

To use as benchmark to compare HAR-RV's performance, we use two of the most prominent econometric models for volatility forecasting: the EWMA and the GARCH.

3.3.1. Exponentially Weighted Moving Average (EWMA)

The Exponentially Weighted Moving Average, originally outlined in Riskmetrics Technical Document (1996), is a model that explores the volatility tendency to cluster. As previously discussed, volatility exhibits high and significant autocorrelation, that is decaying the more distant the lags are. With this in mind, the EWMA model makes use of the past squared returns to forecast variance, using an exponential weighting to attribute more weight to the most recent returns, thus implying that older returns are less informative than more recent ones. According to this model, conditional variance at time t is given by

$$\sigma_t^2 = (1-\lambda)\sum_{i=1}^{\infty} \lambda^{i-1}\epsilon_{t-i}^2 \qquad ,0 < \lambda < 1 \tag{3.4}$$

where $\epsilon_{t-1} = r_{t-1} - \mu_{t-1}$, r_t is the return at time t, μ_t is the mean return at time t and λ is the decay factor. Additionally, for the reasons previously explained in section 2.1, mean returns are set to zero and thus we get $\epsilon_t = r_t$. With this in mind, it is possible to rearrange equation (3.4) into a much simpler formula which is usually the most commonly used.

$$\sigma_t^2 = (1-\lambda)\, r_{t-1}^2 + \lambda\sigma_{t-1}^2 \qquad ,0 < \lambda < 1 \tag{3.5}$$

This equation is indeed very appealing as, to forecast volatility, one only needs two values: the current estimate of variance and the latest market return.

The decay factor (λ) determines the depth of memory of the EWMA. The lower the decay factor, the higher the importance the model gives to recent observations and the lower it gives to farther observations, thus putting more emphasis on the more informative recent returns. On the other hand, the higher the decay factor, the lower the importance of the most recent observations and higher the importance of the older ones, which means that the EWMA will have a deeper memory by giving significant weight to the more distant returns.

In Figure 3.1, one can see that using a high factor of $\lambda = 0.99$, EWMA assigns low weights to the first lags but that it lasts very long in time, with a very slow decay, only reaching

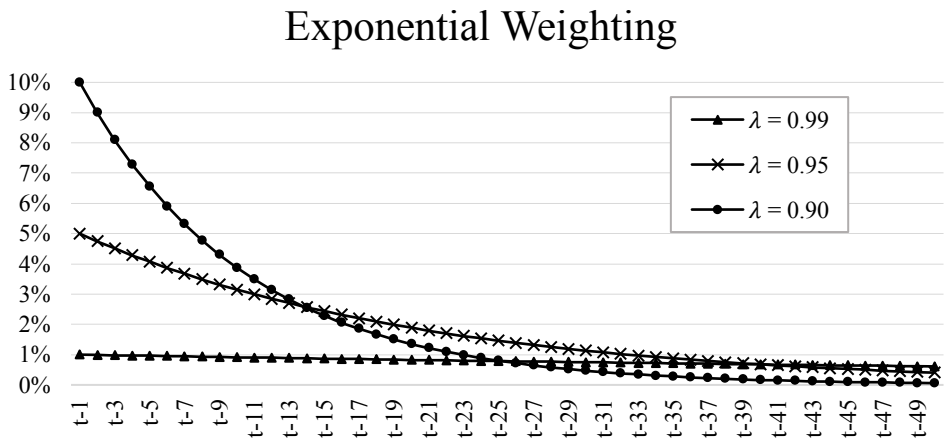

Figure 3.1 - Exponential weights conditional to decay factor.

a weight of under 0.5% on the 70th lag. Contrarily, a factor of $\lambda = 0.90$ values highly the first lags, but weights quickly diminish, reaching a weight of under 0.5% on the 30th lag.

The RiskMetrics 1996 document by JP Morgan, which popularized the EWMA method, tested different values of λ over several financial products and suggested an optimal decay factor of 0.94 for daily data and 0.97 for monthly data.

3.3.2. Generalized Autoregressive Conditional Heteroskedasticity (GARCH)

The Generalized Autoregressive Conditional Heteroskedasticity (GARCH) (Bollerslev, 1986) method was derived from an extension of the ARCH model. The ARCH model, besides also being able to capture the volatility clustering, it is able to model the volatility's mean reversion characteristic. The model does this by formulating conditional variance as a linear function of past squared errors. The ARCH(q) model can be expressed as

$$\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i r_{t-i}^2 \quad , \quad \omega, \alpha_i > 0 \tag{3.6}$$

where q is the number of past returns included, i.e., the number of lagged returns, and $\sum_{i=1}^{q} \alpha_i < 1$. However, to achieve a good fit, the ARCH model requires a relatively high number of lags and therefore the estimation of a high number of parameters. Moreover, the model has a weak capacity to model the persistence of shocks and the asymmetric volatility phenomenon.

The GARCH model brings a solution to the difficulty of modelling the persistence of shocks by introducing the lagged variance in the conditional variance equation. The GARCH(p,q) model can be expressed as

$$\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i r_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 \quad , \quad \omega, \alpha_i, \beta_i > 0 \tag{3.7}$$

following the same assumptions made for the ARCH(q) model, where *p* is the number of lagged variances included and $\sum_{i=1}^{q} \beta_i < 1$. With the addition of the lagged variance, the model can better prolongate volatility shocks in time, thus losing the "bursty" aspect present in ARCH.
In fact, the particular setting of GARCH(1,1) can be shown to be equivalent to the ARCH(∞), i.e., the ARCH model with infinite lags. The ARCH(∞) needs infinite estimates, while the GARCH(1,1) only needs three parameters to be estimated. This showcases one of the main advantages of GARCH, this model needs very few parameters to be estimated. In fact, the GARCH(1,1) is usually enough for fitting financial data (Hansen et al., 2005), thus we will use this setting of the GARCH model.

3.4. Experimental Setup

3.4.1. Data

The main objective of this research is to test whether the use of ML techniques can improve volatility forecasting accuracy and beat the best econometric models. For that purpose, it is necessary to collect market price data on a security. In this research, we use high frequency 5-min market price data for the S&P 500 Index, from the 30th of April 2007 to the 12th of August 2022. The dataset was purchased from "FirstRate Data", a leading provider of high-resolution intraday stock market, crypto, futures and FX data. This timeline was chosen in order to include a long enough period to conduct a thorough analysis. This timeline includes periods of both high and low volatility in the US markets, which the S&P 500 Index aims to track. It is important to expose the predictive models to different market conditions to demonstrate its resilience. Following the analysis of Andersen et al. (1997), the daily realized volatility is going to be computed based on 5-minute returns. This frequency is chosen because the 5-minutes horizon reveals to be short enough that "the accuracy of the continuous record asymptotics underlying our realized volatility measures work well", and long enough that the "confounding influences from market microstructure frictions are not overwhelming" (Anderson et al. 2001).

The returns and realized volatility at day *t*, r_t and RV_t respectively, were computed following the formulas:

$$r_t = \sum_{i=1}^{n} r_{i,t} \tag{3.8}$$

$$RV_t = \sqrt{252 * \sum_{i=1}^{n} r_{i,t}^2} \tag{3.9}$$

where $r_{i,t} = \log\left(r_{i,t}/r_{i-1,t}\right)$ is the i^{th} 5-minute logarithmic return of day *t*, and *n* is the number of 5-minute intraday intervals within that day.

After computing all the variables, from a total of 307.042 intraday high-frequency S&P 500 market prices, the final dataset is composed of 3.849 daily returns and realized volatilities. The following table presents the descriptive statistics of the dataset.

	5-minute Close Price	*5-minute log Returns*	*Daily RV*	*Daily log Returns*
Count	307042	307041	3849	3849
Mean	2151,09	0,0003%	14,98%	0,03%
Median	1980,88	0,0007%	11,64%	0,07%
St. Deviation	991,13	0,1361%	12,16%	1,31%
Minimum	667,04	-8,49%	1,86%	-12,77%
Maximum	4817,67	6,00%	168,55%	10,96%
Kurtosis	-0,08	215,71	24,79	12,15
Skewness	0,87	-2,00	3,76	-0,54
Aug. Dickey-Fuller	0,52	-58,93*	-7,367*	-15,06*

* indicates significance at the 1% level

Table 3.1 - Dataset descriptive statistics

A notable observation is the fact that daily realized volatility and daily log returns do not have a unit root. Unit roots can cause unpredictable results in a time series analysis, therefore variables containing a unit root have to be differenced prior to analysis. Since the Augmented Dickey-Fuller Test for daily RV and daily log returns rejects the presence of unit root at the 1% level, we can proceed without differencing.

3.4.2. Evaluation Metrics

Model evaluation is very important as it helps to understand the performance of a model and provide a comparable metric to contrast with other models. Following Corsi (2009), for this research, the two main metrics chosen to evaluate forecast accuracy in and out-of-sample were the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE).

The MAE is an absolute measure of the goodness of fit of the model. As the name suggests, it calculates the equal-weighted average of the absolute errors between actual and predicted values.

$$MAE = \frac{1}{n} * \sum_{i=1}^{n} \left| RV_i - \widehat{RV_i} \right| \tag{3.10}$$

The Mean Square Error (MSE) measure also takes the average error, however, instead of using the absolute value of errors it takes the squared value of errors. Taking the square of errors will affect the weights attributed to each error and thus giving and higher penalisation

for larger prediction errors. Due to MSE values sometimes being too large for proper comparison and not easy for interpretation because of errors being squared, more commonly the RMSE is used. The RMSE is nothing more than the square root of the MSE, keeping the same properties relative to penalising more heavily larger errors.

$$RSME = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} \left(RV_i - \widehat{RV}_i\right)^2} \tag{3.11}$$

Besides the two metrics of error magnitude, a directional criterion was also used. This criterion measures, in percentage, how many times the direction (first difference) of the predicted values match the direction of the actual realized volatility. The directional criterion can be defined as:

$$Direction = \frac{1}{n} * \sum_{i=1}^{n} Direction_i \tag{3.12}$$

$$\text{with} \quad Direction_i = \begin{cases} 1, & if\ sign(\Delta RV_i) = sign\left(\Delta \widehat{RV}_i\right) \\ 0, & otherwise \end{cases}$$

where $\Delta RV_i = RV_i - RV_{i-1}$ and $\Delta \widehat{RV}_i = \widehat{RV}_i - \widehat{RV}_{i-1}$.

3.5. Results

In the literature, HAR is often considered as one of the best models there is for volatility forecasting. In this research, after computing the HAR, GARCH and EWMA, the results can now be analysed and the actuality of HAR's superiority attested. All the models were estimated using a 1 year rolling window, resulting in estimates for the interval between 30-05-2008 and 11-08-2022. All the computations to setup and optimize all the models were performed in Python. The EWMA and HAR-RV were built using tools from the python library Statsmodels and the GARCH(1,1) using the library Garch.

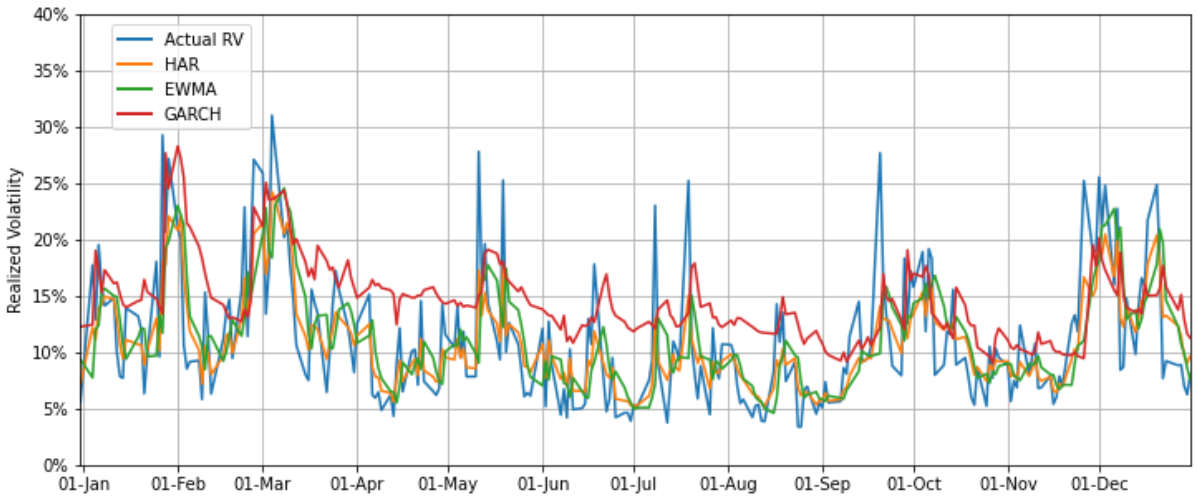

Figure 3.2 - HAR vs. EWMA vs. GARCH prediction plot against Actual RV for the year of 2021

The results show a very clear performance hierarchy, that is in consonance among all evaluation metrics, in and out-of-sample. As expected, the results indicate a much higher prediction accuracy for the HAR model. The GARCH(1,1) model had the worst performance with an out-of-sample RMSE of 7,86%, followed by EWMA with an out-of-sample RMSE of 6,96%. Clearly set apart, for the period analysed, the HAR-RV model scored an out-of-sample RMSE of only 3,97%, thus displaying a 42.96% improvement over the second-best model (GARCH). In regards to the directional criterion, HAR-RV also stands out with over double the directional accuracy of the second-best model. The long-memory capacity given by the aggregation of RV at different horizons' seems to really differentiate HAR from the competing econometric models that do not have long-memory.

	RMSE in-sample	*RMSE out-of-sample*	*MAE in-sample*	*MAE. out-of-sample*	*Direction in-sample*	*Direction out-of-sample*
EWMA (1Yr-RW)	*6,88%*	*6,96%*	*4,18%*	*4,21%*	*32,14%*	*32,34%*
GARCH (1Yr-RW)	*7,67%*	*7,86%*	*5,19%*	*5,36%*	*42,94%*	*43,27%*
HAR (1Yr-RW)	*3,90%*	*3,97%*	*2,31%*	*2,41%*	*90,99%*	*89,76%*

Table 3.2 – Econometric models performance metrics results.

Despite performance differences, none of the three models show signs of overfitting with the highest relative difference between in- and out-of-sample results being 2.38%, corresponding to the GARCH model.

4. Machine Learning Models

4.1. Machine Learning Approaches Literature Review

The literature of classical econometric models for volatility forecasting is very extensive, dates back many decades and has achieved great progress in dealing with the challenging dynamics of volatility. However, market volatility has intrinsic properties such as high autocorrelation, low signal-to-noise ratio and nonlinearity of the underlying structure that will always be problematic for many econometric models. Machine learning techniques are able to overcome all of these properties (Varian, 2014). Machine leaning techniques are famously know to excel at capturing complex non-linear relationships and uncovering unknown trends in data. Finance is undoubtedly a key area in which these techniques can be applied, especially with the huge upsurge of access to big amounts of data. However, the literature on the application of ML on the forecasting of volatility is quite limited. It was only in the past decade that more approaches have been proposed and explored, with some families of ML algorithms being the most popular for this purpose. For the sake of clarity, this literature review will be organized by ML algorithm family, rather than chronologically.

One ML approach explored was the component-wise boosting method, by Mittnik et al. (2015). This method is capable of building a parsimonious model that can take a high number of predictors and, unlike other models, allows for the straightforward interpretation of its parameter estimates. Considering a wide range of potential risk drivers, the authors show that the component-wise boosting method is able to substantially improve the out-of-sample volatility forecast relative to the GARCH and EGARCH methods, for both the short and long-term.

Many different ML approaches were studied and some with relative success, however, when it comes to using a ML method to forecast volatility, the Artificial Neural Networks (ANNs) methods dominates the literature. Neural Networks are some of the best machine learning models to uncover and model complex latent interactions between variables, as these are able to approximate rather well many linear and non-linear functions without knowing the generating process behind the data (Bucci, 2020). Indeed, ANNs have been discovered to be very suitable to forecast volatile financial products that present a nonlinear dependence, such as stock prices, realized volatility or exchange rates (Donaldson et al., 1996). Many authors used Neural Networks in combination with the known econometric models. Donaldson et al. (1997), combined simple ANNs with the GARCH model and was able to demonstrate that the

ANN model captures dynamics overlooked by other GARCH-family models, as well as it outperforms their out-of-sample volatility forecasts. Similarly, Hajizadeh et al. (2012) reached the same conclusion using EGARCH estimates to feedforward to a neural network complemented with explanatory variables. More studies were published supporting ANN-based GARCH-like models as opposed to the basic parametric GARCH, such as Schittenkopf et al. (2000), Taylor (2000), Dunis et al. (2002) and Kristjanpoller et al. (2014).

Nevertheless, the traditional ANN algorithm has some shortcomings, it needs a very substantial number of controlling parameters, it struggles at reaching a final solution and is prone to overfitting (Tay et al., 2001). Therefore, some researchers employed a Support-Vector Machines (SVM) method, a model with a similar concept to ANNs, though are not classified as a neural network. The SVM model can keep the same advantages of the ANN, but is able to easily reach a unique global solution using quadratic programming and implements structural, rather than empirical, risk minimization principle which helps to prevent overfitting, hence SVM should result in better forecasts (Gunn, 1998). Chen et al. (2010) constructed and tested an SVM-GARCH model for volatility forecasting. The SVM-GARCH model significantly outperformed the simple GARCH, EGARCH and ANN-GARCH model, thus confirming the theoretical advantage of SVM.

Inside the ANNs, depending on the neurons organization, the ANNs can be Feedforward Neural Networks (FNNs) or Recurrent Neural Networks (RNNs). In FNNs the information moves in only one direction and, from inputs nodes to hidden nodes to output nodes, without ever forming a cycle between nodes. The RNNs allow neural connections with neurons in the same our previous layers, thus enabling cycles of information and the incorporation of temporal dynamic behaviours. However, the vanilla RNN can usually only hold short-term information (Hochreiter et al., 2001). Withing the RNNs, more advanced extensions were developed that were competent at holding long-term information, being the main one the Long Short-Term Memory (LSTM) network (Hochreiter et al., 1997). Recently, Xiong et al. (2015) used a LSTM network incorporated with Google domestic trends as explanatory variables to forecast S&P 500 volatility, being capable to clearly outperform the linear Ridge/Lasso and GARCH benchmarks. Bucci (2020) compared the predictive performance of FNNs, the LSTM network and the NARX network (another RNN) with advanced econometric ARFIMA-family approaches. Bucci's study shows that both RNNs (LSTM and NARX) are able to outperform the econometric methods, as well as that the long memory seems to also enhance the forecasting accuracy in highly volatile conditions.

4.2. Machine Learning Theory

4.2.1. Artificial Neural Networks (ANN)

Artificial Neural Networks (ANN), commonly called just Neural Networks, are mathematical models inspired in the way information is processed in the biological neural networks of the brain. The Neural Network system has been a matter of study for almost 80 years (McCulloch et al., 1943), but it is its application in machine learning that have made them a very exciting topic from a technological perspective.

ANNs are made up of artificial neurons which are organized in layers. Artificial neurons are the elementary processing unit of ANNs, and are interconnected between them to create a network. Neurons receive information from one or more inputs and turn them into a signal. The inputs of a neuron can be external data or output signals from other neurons. Neurons transform the input information by computing a weighted sum of all inputs using the weights assigned to each input connection. Following this, a bias term of constant value equal to 1 is added to the result of the weighted sum, acting like the constant term in a regression and ensuring that even if all inputs are zeros, the neuron is going to be activated. After, the resulting value will be run through an activation function which will introduce non-linearity into the neural network. The activation function will transform the information in such a way that it

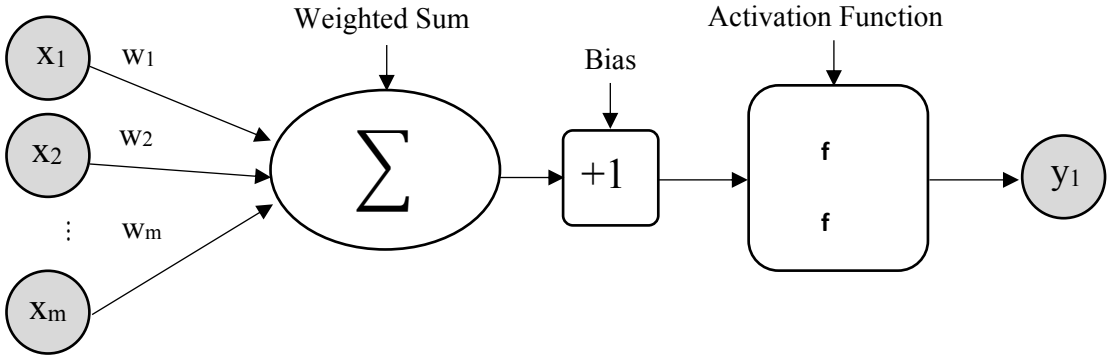

Figure 4.1 - Perceptron - representation how a single neuron works.

will decide if the information in that neuron is relevant or not to the process. There are many different activation functions that can be used, but the three most prominent non-linear functions are the Sigmoid function, the Tanh function and the Rectified Linear Unit (ReLU) function, which are chosen depending on the problem being modelled. The final output of the neuron is then passed onwards to the next neuron.

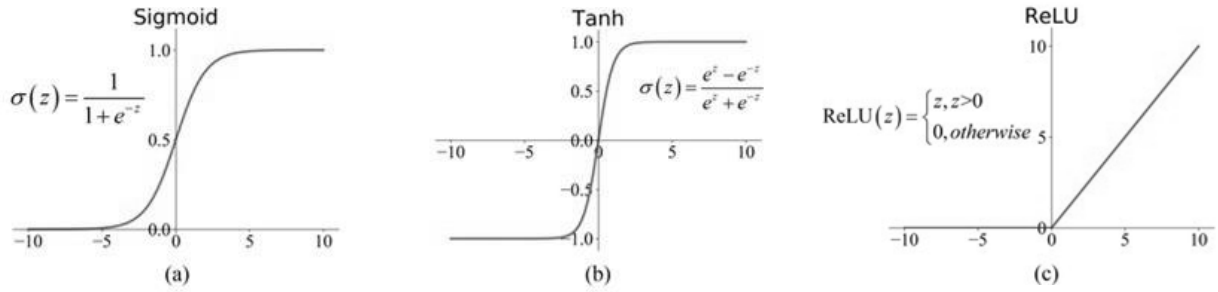

Figure 4.3 - Commonly used activation functions: (a) Sigmoid, (b) Tanh and (c) ReLU.

Neurons can be grouped into three main types of layers: input layers, hidden layers and output layers. Input layers takes the raw input and directs the information to the hidden layer, without performing any calculation. The hidden layer, as the name suggests, is not an exposed layer in the ANN's architecture. This type of layer is responsible for all the computations done on the features introduced through the input layer. The output layer is the last layer of the ANN, it groups all the information that was computed throughout the network and delivers a final result value.

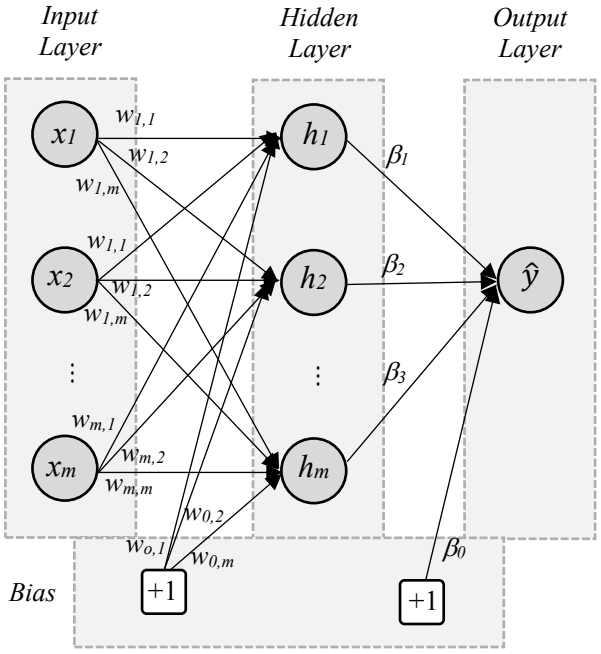

Figure 4.2 - Single hidden layer Feedforward Neural Network diagram.

Artificial Neural Networks can have a structure with only one or, more commonly, multiple hidden layers in which case it is be considered a Deep Neural Network (DNN). Neurons of one layer can only connect to neurons of the layers adjacent to it, i.e., the layer immediately before and after itself. Depending on the organization of the ANN, the flow of information along the network can be different from one model to the other. If the information flows in a single direction from input layer, to hidden layers, to output layer, without ever

looping between two hidden layers, the network has a forward architecture. These types of ANNs, in which a neuron only feeds information to the layer after itself, are called Feed-Forward Neural Networks (FNNs).

However, these ANNs are not very useful for time-series forecasting as they make decision based only on current inputs and therefore have no memory. For example, imagine an ANN that was trained to predict tomorrow's volatility, it would use the information available today and tune its setting to try to make the best prediction. If tomorrow we wanted the model to predict volatility of the day after, it would once again use the data available tomorrow and re-tune itself all over again, without keeping any insights from the previous prediction and without the sense of a sequential order. To solve these shortcomings, Elman (1990) proposed the Recurrent Neural Network (RNN) architecture. These neural networks allow information to flow not only from input to output, but also between hidden layers of previous experiences.

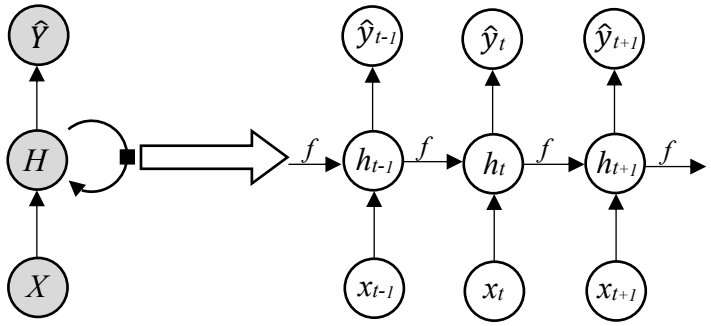

Figure 4.4 - Recurrent Neural Network mechanism diagram.

This way RNNs are capable of remembering what they have learned in the past and use it for future predictions, thus they are very fit for timeseries forecasting. In this model, inputs are fed to the model organized in a vector of sequential data. Sequential data points inputted run through the network, each resulting in an output. The hidden layer(s), usually called hidden state in the context of RNNs, are fed back to the model at the passing of each datapoint, thus allowing the model to keep memory. Each hidden state can be understood by the following equation

$$h_t = f_w(h_{t-1}, X_t) \tag{4.1}$$

where h_t is the current hidden state, f_w is some function with parameters *w,* h_{t-1} is the old hidden state and X_t is the datapoint of the input vector for time *t*.

RNNs are oftentimes trained using the Gradient Descent and the Backpropagation optimization algorithms, which aim to adjust the neural network weights (depicted in Figure 4.2) such that errors are minimized. However, with deep RNN these algorithms tend to be very

faulty and often suffer from vanishing gradients in which the weights become very small and the model stops learning, or exploding gradients in which the weights increase a lot making the model very unstable. These complications, prevent long-term dependencies from being captured by RNNs, limiting the model to a very short-term memory. In a timeseries setting, this is a major flaw of the model. It is critical that RNNs are able to keep long-term memory for a better timeseries forecasting ability.

4.2.2. Long Short-Term Memory (LSTM)

Given these limitations, Hochreiter et al. (1997) proposed the Long Short-Term Memory (LSTM) model, a more advanced type of RNN. LSTMs are renowned for being very effective at modelling both short and long-term dependencies, thus being more appropriate for volatility forecasting since volatility has shown to exhibit long memory itself. To achieve this long-term memory the model uses a group of gate systems to monitor the information stored in the hidden state. At each time-step of the model training, it is able to choose which past information to forget, which new one to add and what result to output. The system function with three gates: the forget gate, the input gate and the output gate.

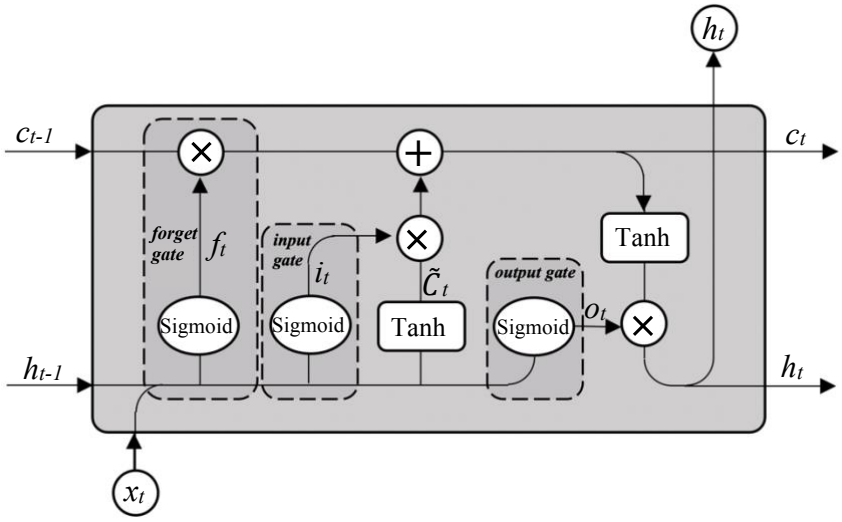

Figure 4.5 - LSTM mechanism diagram.

To better understand the mechanism of the LSTM, lets refer to Figure 4.5. The main component of the LSTM model is the cell state, represented above by the top horizontal straight line. It acts as a conveyor belt where all the relevant information flows. The first action by the model is choosing which information to forget using the forget gate. The forget gate takes x_n (n^{th} value of the input vector, corresponding to time t) and h_{t-1} (*t-1*'s hidden state) and using a sigmoid function outputs a number (f_t) between 0 and 1 for each value in the cell state, representing the fraction of information to be kept. The f_t factor is applied by multiplying it by

the old state (C_{t-1}). The next action is selecting which new information to add to the cell state using the input gate. Using a sigmoid function, the gate first decides on the information it will want to keep, in a similar manner to the forget gate using x_n and h_{t-1}. This decision is kept in i_n. Then, using a tanh function, the gate creates a vector ($\hat{C}_t$) with all the new potential values from x_n and h_{t-1} that could be added to the cell state. After that, both i_n and $\hat{C}_t$ are combined by multiplying both, creating an update factor with all the values scaled by the amount by which the sigmoid function decided to update each state value. This factor is then added to the cell state. Lastly, the LSTM model will select its output using the output gate. This gate will first take x_n and h_{t-1} and apply them a sigmoid function, resulting in a factor (o_t) containing the information on which information to output. Then, it takes the cell state already updated with the forget and input gates, runs it through a tanh function, to force the values between -1 and 1, and multiplies the result by the o_t , resulting in only the portion of information it chose to output.

4.2.3. Gated Recurrent Unit (GRU)

The LSTM model is indeed well qualified to model long term dependencies, thus the reason why it has been one of the most used models for that purpose. Nonetheless, more

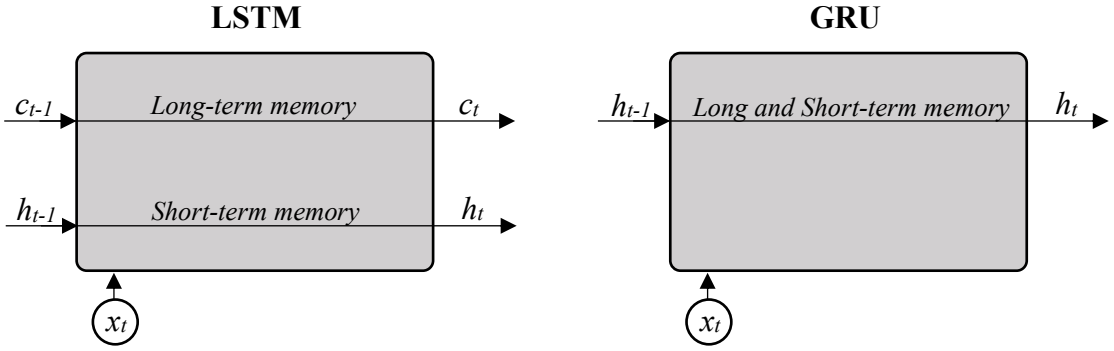

Figure 4.6 - LSTM vs. GRU architecture diagrams.

recently, a variation of this model was proposed by Chung et al. (2014), the Gated Recurrent Unit (GRU). This model, that has been gaining increasing popularity, presents a simpler version of LSTM in which the forget and input gates are combined, the cell state and hidden state are merged together, and that can produce a performance comparable to the LSTM.
By having only two gates and combining long and short-term memory in its hidden state, the GRU is computationally lighter to run than LSTM.

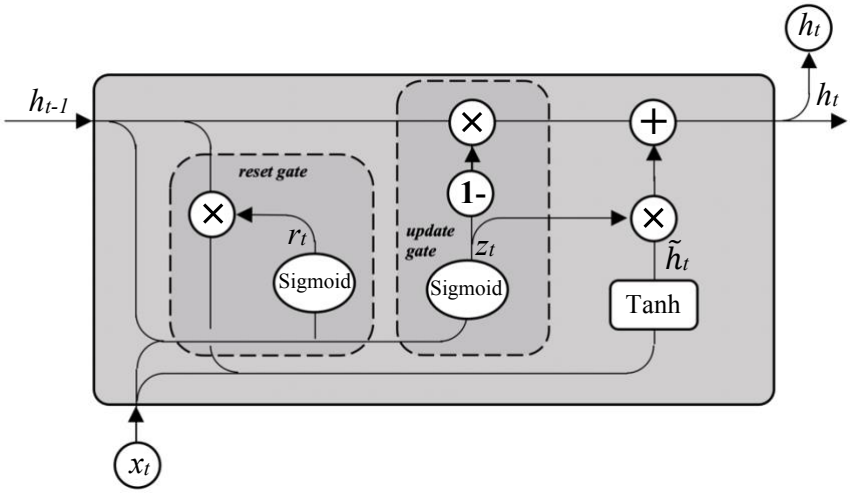

Figure 4.7 - GRU mechanism diagram.

Let's use the graphical representation in Figure 4.7 to guide us through the GRU mechanism. First x_n and h_{t-1} are fed to both of its gates: the reset gate and the update gate. Similarly to what happened in LSTM, the reset gate takes the x_n and h_{t-1} values and feeds them to a sigmoid function, getting r_t containing the information which should be kept from the previous hidden state. The r_t will then be multiplied by h_{t-1} to have control on how much of the hidden state from the previous step should be fed in to make the prediction. Since we want to use this information to predict an actual output, the piece resulting from this multiplication will then be ran through a tanh function, creating a hidden state candidate ($\tilde{h}_t$). The update gate can be seen as a replacement for the duties of the forget and input gate on LSTM. This gate also feeds x_n and h_{t-1} to a sigmoid function getting the z_t value determining how much information is taken in from the new input. This value is used for two purposes hence we see it following two branches. First z_t passes through the block represented with the "1-" where one minus z_t represents the amount we do not want to keep. This 1- z_t argument is then used to update h_{t-1} by multiplication, forming what can be seen as the "past component" of the hidden state computations. Secondly, z_t is also used to multiply by the "hidden state candidate" ($\tilde{h}_t$) to produce the "new component" of the hidden state. Lastly, the new and old components are added to make up the hidden state to be used in the next prediction.

4.3. Experimental Setup

For the setup and application of the machine learning models, the dataset and evaluation metrics used are the same as for the previous models.

4.3.1. Dataset Split

In order to train and evaluate machine learning models, the dataset needs to be split into training, validation and testing sets. The training set is used by the neural network to learn the patterns present in the data and the validation set is used to tune the model's hyperparameters (parameters that define the model's architecture, more on this ahead) in order to optimise them. The testing set is used to test the optimised model's performance on data it has never seen before. These three subsets should be homogenous between them and representative of the whole dataset, which can be very challenging considering a 15-year horizon. There would be a great risk of the testing set being characterized by one type of market, e.g. a very volatile/stable period. Evaluating the performance of a model on a non-representative test set will introduce bias and not accurately depict the models' real prediction capacity. To tackle this issue, we use a 5-fold time-series cross-validation technique, in which almost all of the dataset is used as a testing set at least once. In a timeseries context, the cross-validation is setup in a time-ordered way with an expanding training set, as represented in the diagram below.

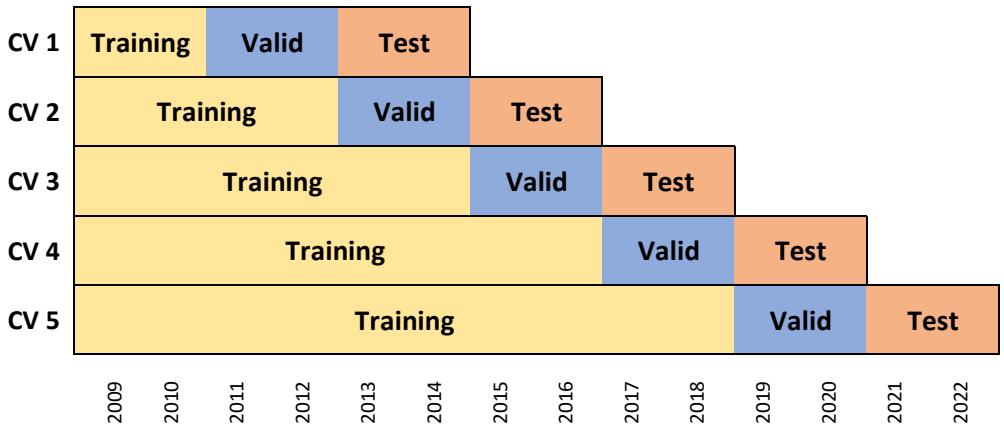

Figure 4.8 - Cross-validation diagram

For each fold, the models are trained and tuned on the train and validation sets and the prediction error evaluated on the test set. By averaging the prediction errors achieved on each fold it is possible to obtain an "almost unbiased estimate of the error" (Varma and Simon, 2006), and therefore a good estimate of the overall performance of the model.

Machine learning models' hyperparameters are not meant to be re-optimized on a rolling basis just like the previous econometric models were. ML models are built to learn from past data and be able adapt to make predictions in any scenario. Nonetheless, there could be models that are better in low volatility scenarios and others in high volatility scenarios. The 5-fold cross validation also allows for testing this hypothesis, given that it allows for some differentiation between periods, resulting in 5 different test sets with different levels of volatility. The table below summarizes the descriptive statistics on the realized volatility of each testing set.

	Daily Realized Volatility			
	mean	*stdev*	*min*	*max*
CV 1	9,87%	4,29%	2,07%	31,32%
CV 2	12,10%	7,84%	3,02%	104,65%
CV 3	8,39%	5,13%	2,24%	43,76%
CV 4	17,13%	17,81%	1,86%	168,55%
CV 5	15,30%	8,50%	3,39%	60,58%

Table 4.1 - Realized volatility descriptive statistics for each cross-validation test set.

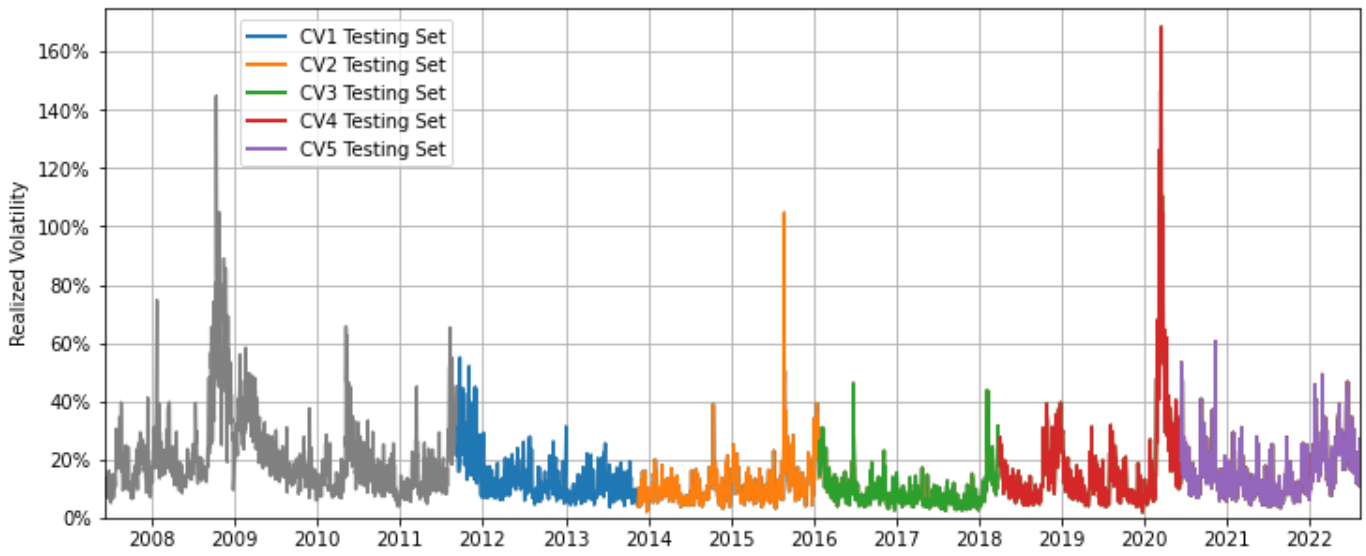

Figure 4.9 - Cross Validation testing sets plot

4.3.2. Model Optimization & Hyperparameter Tuning

To setup the LSTM and GRU models, one needs to perform an hyperparameter tuning. Hyperparameters are the parameters which define the ML models architecture. So before fitting the model to the data, one needs to first set the hyperparameters that define the model. Because the LSTM and GRU models differ only in their internal architecture, but the great majority of its hyperparameters are the same for both models, the tuning process and values used were the same for both models. The hyperparameters relative to functions were not left to be tuned, instead the appropriate value was chosen for each. As activation functions, the tanh was chosen for the hidden layers and a linear function was chosen for the output layer. The model optimizer was set to be the Adaptive Moment Estimation method. Because volatility sometimes have big jumps and thus big forecasting errors are especially undesirable, MSE was set as the models' loss function. MSE is used over RMSE also for reasons of computational efficiency of the machine learning models. To optimally tune the remaining hyperparameters for volatility prediction, a grid search technique was used. The grid search technique consists of choosing a set of values for each hyperparameter, testing every possible combination of the set values, and evaluating their performance on the validation set, in order to find the optimal hyperparameters. The table below contains the ranges of values chosen for each of the hyperparameter left to tune.

Window size	[2,5,11,22]
# Hidden layers	[1,2]
# Neurons per layer	[4,6,10]
Batch size	[8, 32 ,128]
# Epochs	[0 - 2000]

Table 4.2 – Table of LSTM and GRU grid search hyperparameters.

Theoretically, with neural networks, one hidden layer should be capable to approximate any continuous function. Kaastra and Boyd (1996) found that, for financial timeseries, one should not use more than 2 hidden-layers as both theory and empirical work show that a higher number of layers will not improve results. For this reason, the hidden layers were set to take the values of 1 or 2. The number of neurons per layer, despite its relevance, can only be found

by experimentation (Kaastra and Boyd, 1996) and thus were left to tune between a sensible range of values which are common practice in literature. The windows size, i.e., the number of lags to include are a very important parameter to be tuned to achieve a good accuracy. To give the models the chance of getting as much information as the HAR model that takes into account the past month data, we let this hyperparameter take the values of 2-, 5-, 11- and 22-day lags.

Lastly, there is the batch size and the number of epochs. Even though a little more complex to explain, the batch size can be thought as the number of days of the timeseries data (inputs + realized volatility) that the model would work through before updating the internal parameters, while the number of epochs is the number of times the algorithm would run through the entire training dataset. The batch size was set to the most common values used in the literature. To tune the number of epochs, we set every model to allow a maximum of 2000 epochs, but with an early stopping mechanism to stop the learning process if the model does not improve for 100 consecutive epochs.

4.4. Models and Results

4.4.1. Model Performance Analysis and Comparison

After the experimental settings are setup, models can now be tested and evaluated. The ML models were implemented using the python library Keras, with a seed set for reproducibility purposes. Firstly, the LSTM and GRU models were tested using the same base input variables that were used for the econometric models - the daily log returns and daily realized volatility. The results achieved are summarized in the table below.

	RMSE in-sample	*RMSE out-of-sample*	*MAE in-sample*	*MAE. out-of-sample*	*Direction in-sample*	*Direction out-of-sample*
EWMA (1Yr-RW)	*6,88%*	*6,96%*	*4,18%*	*4,21%*	*32,14%*	*32,34%*
GARCH (1Yr-RW)	*7,67%*	*7,86%*	*5,19%*	*5,36%*	*42,94%*	*43,27%*
HAR (1Yr-RW)	*3,90%*	*3,97%*	*2,31%*	*2,41%*	*90,99%*	*89,76%*
LSTM (LogRet & RV)	*6,08%*	*5,63%*	*4,07%*	*3,56%*	*48,14%*	*45,12%*
GRU (LogRet & RV)	*6,08%*	*5,70%*	*4,06%*	*3,57%*	*48,51%*	*45,82%*

Table 4.3 – LSTM and GRU models performance metrics results vs. econometric models.

The machine learning models were able to beat the EWMA and GARCH models by a clear margin. However, the LSTM and GRU out-of-sample RMSE scores of 5.63% and 5.70%,

respectively, still fell short of HAR-RV's error of only 3.97%. The HAR shows to be superior not only in terms of error magnitude, but also in the prediction direction accuracy.

One aspect that distinguishes HAR-RV from the two other econometric models is the aggregation of volatility over different time horizons. Indeed, the explanatory power of these aggregated variables could be what gives HAR the hedge over the others. For that reason, a second variable set was tested, adding the weekly RV and monthly RV to the previous variable set. The addition of the aggregated variables only marginally improved the ML models' performances, with out-of-sample (oos) RMSE scores of 5,61% and 5,66%, for LSTM and GRU, respectively.

In financial literature, when trying to achieve better models, it is often common practice to combine different models with the ambition of the combined model yielding a better performance than any of the constituent ones. In fact, it has been shown that "forecast accuracy can be substantially improved through the combination of multiple individual forecasts" (Clemen, 1989). Adapting this approach to our problem, we added the HAR-RV prediction as an input variable to the ML models, in addition to the base variables daily Log Returns and daily RV. The combined model performed well and was able to outperform the previous ML models, but not the HAR. The LSTM achieved and oos RSME of 5,43%, while the GRU 5,60%.

As mentioned in part 2.1., Implied Volatility (IV) is derived from option prices and encapsulates investor's expectations of future economic conditions. It can be interpreted as the market's forecast of future volatility and thus can contain information content that may be very useful in predicting volatility. Indeed, it is argued that, if options markets are efficient, IV should be the only variable needed to predict future volatility (Christensen et al., 1998). Although, in practice, IV is not a competitive standalone predictor, some evidence suggests that its incorporation into other volatility models improves the accuracy of forecasts (Frijns et al., 2010; Kambouroudis et al., 2016; Blair et al., 2001). In the pursuit of improving the ML models' performances, the Chicago Board Options Exchange's Volatility Index (VIX) – a measure of the S&P 500's IV for the next 30 days – was tested as an input variable with the aim of benefiting from IV's information content. The use of VIX in addition to the base variable set (daily log returns and daily RV), improved the ML models accuracy relative to their performance with the base variable set alone. However, it couldn't match the performance of variable set including HAR prediction, let alone the performance of the HAR-RV model itself.

Lastly, we tried a "kitchen sink" approach to the ML models, i.e., we ran the models with all the variables discussed so far. This technique is usually used when the goal is simply

predictive modelling, without much concern about which particular predictors go into the model as long as the final model yields the best possible predictions. This variable set produced the most accurate results so far. The "kitchen sink" LSTM and GRU scored an oos RMSE of 5,41% and 5,37%, respectively. However, once again, even though it is better than the previously tried ML iterations, these cannot reach the accuracy of the simple HAR-RV model.

	RMSE in-sample	*RMSE out-of-sample*	*MAE in-sample*	*MAE. out-of-sample*	*Direction in-sample*	*Direction out-of-sample*
EWMA (1Yr-RW)	*6,88%*	*6,96%*	*4,18%*	*4,21%*	*32,14%*	*32,34%*
GARCH (1Yr-RW)	*7,67%*	*7,86%*	*5,19%*	*5,36%*	*42,94%*	*43,27%*
HAR (1Yr-RW)	*3,90%*	*3,97%*	*2,31%*	*2,41%*	*90,99%*	*89,76%*
LSTM (LogRet & RV)	*6,08%*	*5,63%*	*4,07%*	*3,56%*	*48,14%*	*45,12%*
GRU (LogRet & RV)	*6,08%*	*5,70%*	*4,06%*	*3,57%*	*48,51%*	*45,82%*
LSTM (Aggregeted RVs)	*6,19%*	*5,61%*	*4,03%*	*3,43%*	*48,75%*	*46,79%*
GRU (Aggregeted RVs)	*6,01%*	*5,66%*	*3,93%*	*3,45%*	*49,28%*	*46,50%*
LSTM (HAR)	*6,12%*	*5,43%*	*3,93%*	*3,36%*	*47,06%*	*44,85%*
GRU (HAR)	*5,96%*	*5,60%*	*3,87%*	*3,46%*	*50,34%*	*47,40%*
LSTM (VIX)	*5,95%*	*5,62%*	*3,85%*	*3,41%*	*49,77%*	*48,30%*
GRU (VIX)	*6,17%*	*5,58%*	*3,97%*	*3,44%*	*49,59%*	*48,58%*
LSTM (Kitchen Sink)	*5,77%*	*5,41%*	*3,76%*	*3,38%*	*50,43%*	*48,41%*
GRU (Kitchen Sink)	*6,00%*	*5,37%*	*3,88%*	*3,36%*	*50,85%*	*49,45%*

Table 4.4 – Performance Metrics (In- and Out-of-Sample)

	Variable List
LogRet & RV	*LogRet, Daily RV*
Aggregated RVs	*LogRet, Daily RV, Weekly RV, Monthly RV*
HAR	*LogRet, Daily RV, HAR-RV Prediction*
VIX	*LogRet, Daily RV, VIX*
Kitchen Sink	*LogRet, Daily RV, Weekly RV, Monthly RV, HAR Prediction, VIX*

Table 4.5 - List of variables for each variable set.

4.4.2. Crisis Scenario

Machine leaning models are famously known for their ability to capture complex non-linear relationships and uncover unknown trends in data. As proposed, it is valuable to test if these methods are indeed able to uncover any hidden patterns that precede high volatility periods such as crisis. For this purpose, the performance metrics are computed for the period

of the Covid-19 pandemic crisis. We use as a crisis window the period from the 1st of February 2020 to the 1st of May 2020.

	RMSE out-of-sample	*MAE out-of-sample*	*Direction out-of-sample*
EWMA (1Yr-RW)	*22,85%*	*15,25%*	*38,71%*
GARCH (1Yr-RW)	*25,02%*	*17,38%*	*48,39%*
HAR (1Yr-RW)	*15,51%*	*9,95%*	*79,03%*
LSTM (LogRet & RV)	*20,58%*	*13,24%*	*46,77%*
GRU (LogRet & RV)	*21,35%*	*13,51%*	*41,94%*
LSTM (Aggregeted RVs)	*21,75%*	*13,79%*	*40,32%*
GRU (Aggregeted RVs)	*23,14%*	*14,61%*	*46,77%*
LSTM (HAR)	*20,89%*	*12,88%*	*48,39%*
GRU (HAR)	*22,83%*	*14,04%*	*43,55%*
LSTM (VIX)	*21,90%*	*13,38%*	*50,00%*
GRU (VIX)	*21,75%*	*13,69%*	*45,16%*
LSTM (Kitchen Sink)	*23,17%*	*15,25%*	*45,16%*
GRU (Kitchen Sink)	*22,21%*	*14,05%*	*40,32%*

Table 4.6 – Covid-19 Period Performance Metrics (Out-of-Sample)

The ML approaches ranked similarly, relative to the econometric ones, in their accuracy at predicting volatility during the Covid-19 crisis period. Most ML models outperformed EWMA and GARCH, and HAR-RV clearly beat any other model by a considerable margin. No evidence is found that suggests that ML methods have a particular edge during high volatility events. Below we can see the plot of the predictions of HAR-RV and the best performing LSTM and GRU models, which during covid are, for both, the simpler variable set with daily log returns and daily RV.

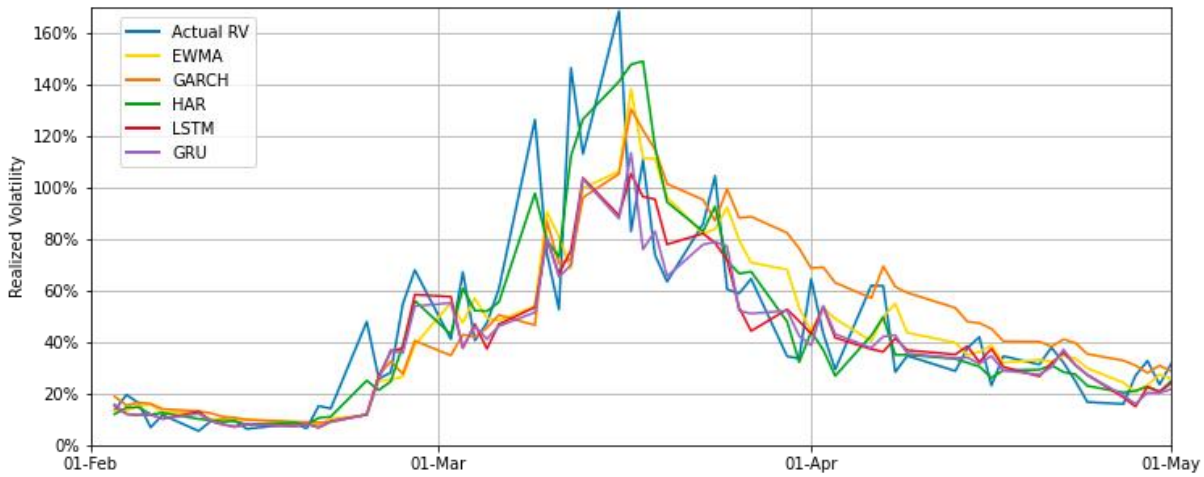

Figure 4.10 – Covid-19 Period HAR-RV, LSTM and GRU Predicted Volatility vs. Actual Realized Volatility.

4.4.3. Results Discussion

The machine leaning methods were able to beat the GARCH and EWMA by a good margin, but have performed considerably worse than the HAR-RV model. This relative performance hierarchy is constant for every variable set across all performance metrics used.

Both machine learning models performed similarly and improved with the introduction of more variables. The two algorithms achieved their best oos performance with the same variable set – the kitchen sink-, that included all the variables discussed, which may suggest that the models can benefit from being fed more information.

The in-sample performances of the ML models were worse than out-of-sample, however, this can be misleading as they are measured for the whole training and validation set, but the model architecture was optimized for the validation set only. In fact, if instead we look at the in-sample performance only on the validation set, we are able to get a more telling estimate.

	RMSE validation	*RMSE out-of-sample*	*MAE validation*	*MAE. out-of-sample*	*Direction validation*	*Direction out-of-sample*
EWMA (1Yr-RW)	*6,88%*	*6,96%*	*4,18%*	*4,21%*	*32,14%*	*32,34%*
GARCH (1Yr-RW)	*7,67%*	*7,86%*	*5,19%*	*5,36%*	*42,94%*	*43,27%*
HAR (1Yr-RW)	*3,90%*	*3,97%*	*2,31%*	*2,41%*	*90,99%*	*89,76%*
LSTM (LogRet & RV)	*4,80%*	*5,63%*	*3,26%*	*3,56%*	*47,85%*	*45,12%*
GRU (LogRet & RV)	*5,02%*	*5,70%*	*3,36%*	*3,57%*	*46,42%*	*45,82%*
LSTM (Aggregeted RVs)	*4,95%*	*5,61%*	*3,18%*	*3,43%*	*47,61%*	*46,79%*
GRU (Aggregeted RVs)	*5,18%*	*5,66%*	*3,31%*	*3,45%*	*46,58%*	*46,50%*
LSTM (HAR)	*4,89%*	*5,43%*	*3,21%*	*3,36%*	*48,65%*	*44,85%*
GRU (HAR)	*5,01%*	*5,60%*	*3,19%*	*3,46%*	*48,11%*	*47,40%*
LSTM (VIX)	*4,88%*	*5,62%*	*3,22%*	*3,41%*	*49,20%*	*48,30%*
GRU (VIX)	*4,97%*	*5,58%*	*3,24%*	*3,44%*	*45,19%*	*48,58%*
LSTM (Kitchen Sink)	*4,77%*	*5,41%*	*3,18%*	*3,38%*	*49,57%*	*48,41%*
GRU (Kitchen Sink)	*4,63%*	*5,37%*	*3,08%*	*3,36%*	*50,26%*	*49,45%*

Table 4.7 - Performance metrics with validation-only in-sample performances.

Indeed, looking at the in-sample performances for the validation set only, we see that they are better than the out-of-sample ones. Furthermore, they display some overfitting with performances approximately 12% and 14% worse out-of-sample than for the in-sample validation periods, for the kitchen sink LSTM and GRU respectively. For comparison, the same difference for the HAR model is less than 2%. This overfitting can be caused by a training-validation-testing splits that results in heterogeneous periods of volatility.

One plausible reason for the advanced features and capabilities of the machine learning methods not being able to accomplish a better performance are the possible non-optimal conditions to setup the model. As mentioned above, the training-validation-testing split is a very important step that heavily influences how the models is going to learn. A split in the validation dataset in which the three resulting datasets are not accurately homogeneous will hinder the model's learning ability or even incite the model to be optimized for the characteristics of a non-representative period. This is very difficult to achieve in the real word, using timeseries financial data, which represents a substantial pitfall of the application of machine leaning methods in the forecast of market volatility.

5. Conclusion

With the purpose of bridging the gap between the literature of machine learning methods for volatility forecasting and the traditional econometric models, this research had the aim to explore the ML methods for volatility forecasting, testing their performance and comparing it to the performance of traditional models. More specifically, this study proposed to answer the question whether: 1) machine leaning methods can achieve high accuracy in forecasting volatility; 2) machine learning methods can outperform the best traditional econometric approaches; 3) machine learning methods can better predict and forecast volatility in times of crisis.

In regard to the first question, the two machine learning algorithms that were tested showed to be competent enough at forecasting volatility. The models achieved results that are up to par with other traditional techniques. More specifically, with the kitchen-sink variable set, the LSTM and GRU models yielded an out-of-sample RMSE of 5,41% and 5,37%, respectively, which can be considered a good accuracy.

Relative to the second question, the answer is no. The ML algorithms could not outperform the econometric model regarded as the best for RV prediction - the HAR-RV. The HAR-RV's ability to reproduce the volatility's long memory, fat tails and time persistence characteristics proved to be very powerful and efficient at forecasting volatility, while being greatly less computationally expensive than the ML models. Nonetheless, tough they could not beat HAR-RV, the ML models were able comfortably outperform the EWMA and GARCH(1,1) models. The best ML algorithm scored an RMSE 29,68% and 46,39% lower than

those of EWMA and GARCH(1,1) respectively, but HAR-RV still scored 26,03% lower than it. During the COVID-19 crisis, the models' relative performance hierarchy were similar to the ones of the whole period. The ML methods did not show the capacity to uncover crisis-preceding trends in data nor to have a particular advantage during high volatility events, thus answering the third research question.

Lastly, it is relevant to reflect on the limitations found and suggestions for further research. The main limitation found was that machine learning methods require that a rigorous set of conditions are met for the proper setup and deployment of the models. In practice, these conditions are very difficult to be met using financial data, which greatly undermines the remarkable qualities of the machine learning methods.

The data splitting procedure should be explored in further researches, with the goal of finding a method to split the data in three representative and homogeneous datasets, for a better learning process. Furthermore, this research focused on directly predicting daily RV from historical daily RV. It could be interesting to explore the use of intraday frequencies to predict intraday volatilities and subsequently aggregate them. High-frequency data tends to be noisy due to market microstructures, however these could be assimilated by the model and actually help the model. It could be that the ML algorithms are better at a high-frequency level. Another field in which improvement is most likely achievable is the input variable set. The algorithms showed to benefit from larger input sets, therefore there is a lot of room for further exploration here that can effectively change the best model verdict.

References

Andersen, T. G., & Benzoni, L. (2008). Realized Volatility, Working Paper 2008-14.

Andersen, T. G., & Bollerslev, T. (1997). Heterogeneous information arrivals and return volatility dynamics: Uncovering the long-run in high frequency returns. The journal of Finance, 52(3), 975-1005.

Andersen, T. G., Bollerslev, T., Diebold, F. X., & Ebens, H. (2001). The distribution of realized stock return volatility. Journal of financial economics, 61(1), 43-76.

Arnerić, J., Poklepović, T., & Aljinović, Z. (2014). GARCH based artificial neural networks in forecasting conditional variance of stock returns. Croatian Operational Research Review, 329-343.

Barndorff-Nielsen, O. E., & Shephard, N. (2002). Econometric analysis of realized volatility and its use in estimating stochastic volatility models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 64(2), 253-280.

Blair, B. J., Poon, S. H., & Taylor, S. J. (2001). Modelling S&P 100 volatility: The information content of stock returns. Journal of banking & finance, 25(9), 1665-1679.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of econometrics, 31(3), 307-327.

Bucci, A. (2020). Realized volatility forecasting with neural networks. Journal of Financial Econometrics, 18(3), 502-531.

Campbell, J. Y., Lo, A. W., MacKinlay, A. C., & Whitelaw, R. F. (1998). The econometrics of financial markets. Macroeconomic Dynamics, 2(4), 559-562.

Chen, S., Härdle, W. K., & Jeong, K. (2010). Forecasting volatility with support vector machine-based GARCH model. Journal of Forecasting, 29(4), 406-433.

Christensen, B. J., & Prabhala, N. R. (1998). The relation between implied and realized volatility. Journal of financial economics, 50(2), 125-150.

Christensen, K., Siggaard, M., & Veliyev, B. (2021). A machine learning approach to volatility forecasting. Available at SSRN.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Clemen, R. T. (1989). Combining forecasts: A review and annotated bibliography. International journal of forecasting, 5(4), 559-583.

Donaldson, R. G., & Kamstra, M. (1996). Forecast combining with neural networks. Journal of Forecasting, 15(1), 49-61.

Donaldson, R. G., & Kamstra, M. (1997). An artificial neural network-GARCH model for international stock return volatility. Journal of Empirical Finance, 4(1), 17-46.

Dunis, C. L., & Huang, X. (2002). Forecasting and trading currency volatility: An application of recurrent neural regression and model combination. Journal of forecasting, 21(5), 317-354.

Elman, J. L. (1990). Finding structure in time. Cognitive science, 14(2), 179-211.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica: Journal of the econometric society, 987-1007.

Fama, E. F. (1965). The behavior of stock-market prices. The journal of Business, 38(1), 34-105. ISO 690

Fouque, J. P., Papanicolaou, G., & Sircar, K. R. (2000). Mean-reverting stochastic volatility. International Journal of theoretical and applied finance, 3(01), 101-142.

Frijns, B., Tallau, C., & Tourani-Rad, A. (2010). The information content of implied volatility: Evidence from Australia. Journal of Futures Markets: Futures, Options, and Other Derivative Products, 30(2), 134-155.

Gunn, S. R. (1998). Support vector machines for classification and regression. ISIS technical report, 14(1), 5-16.

Hajizadeh, E., Seifi, A., Zarandi, M. F., & Turksen, I. B. (2012). A hybrid modeling approach for forecasting the volatility of S&P 500 index return. Expert Systems with Applications, 39(1), 431-436.

Hansen, P. R., & Lunde, A. (2005). A forecast comparison of volatility models: does anything beat a GARCH (1, 1)?. Journal of applied econometrics, 20(7), 873-889.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hunter, J. S. (1986). The exponentially weighted moving average. Journal of quality technology, 18(4), 203-210.

Kaastra, I., & Boyd, M. (1996). Designing a neural network for forecasting financial and economic time series. Neurocomputing, 10(3), 215-236.

Kambouroudis, D. S., McMillan, D. G., & Tsakou, K. (2016). Forecasting stock return volatility: A comparison of GARCH, implied volatility, and realized volatility models. Journal of Futures Markets, 36(12), 1127-1163.

Kristjanpoller, W., Fadic, A., & Minutolo, M. C. (2014). Volatility forecast using hybrid neural network models. Expert Systems with Applications, 41(5), 2437-2442.

LeBaron, B. (1992). Some relations between volatility and serial correlations in stock market returns. Journal of Business, 199-219.

Lu, X., Que, D., & Cao, G. (2016). Volatility forecast based on the hybrid artificial neural network and GARCH-type models. Procedia Computer Science, 91, 1044-1049.

Masset, P. (2011). Volatility stylized facts. Available at SSRN 1804070.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5, 115-133.

Mittnik, S., Robinzonov, N., & Spindler, M. (2015). Stock market volatility: Identifying major drivers and the nature of their impact. Journal of banking & Finance, 58, 1-14.

Müller, U. A., Dacorogna, M. M., Davé, R. D., Olsen, R. B., Pictet, O. V., & Von Weizsäcker, J. E. (1997). Volatilities of different time resolutions—analyzing the dynamics of market components. Journal of Empirical Finance, 4(2-3), 213-239.

Nielsen, M. Ø., & Frederiksen, P. H. (2005). Finite sample comparison of parametric, semiparametric, and wavelet estimators of fractional integration. Econometric Reviews, 24(4), 405-443.

Officer, R. R. (1973). The variability of the market factor of the New York Stock Exchange. the Journal of Business, 46(3), 434-453.

Olah, C. (2015). Understanding LSTM networks. http://colah.github.io/posts/2015- 08-Understanding-LSTMs/

Padovani, B. D. P., & Ballini, R. (2016). Previsão De Volatilidade Realizada: Uma Análise Para O Mercado De Ações.

Pagan, A. R., & Schwert, G. W. (1990). Alternative models for conditional stock volatility. Journal of econometrics, 45(1-2), 267-290.

Poon, S. H., & Granger, C. W. (2003). Forecasting volatility in financial markets: A review. Journal of economic literature, 41(2), 478-539.

Rahimikia, E., & Poon, S. H. (2020). Machine learning for realised volatility forecasting. Available at SSRN, 3707796.

Ruoxuan Xiong, Eric P. Nichols, and Yuan Shen. Deep learning stock volatility with google domestic trends, 2015.

Schittenkopf, C., Dorffner, G., & Dockner, E. J. (2000). Forecasting time-dependent conditional densities: a semi non-parametric neural network approach. Journal of Forecasting, 19(4), 355-374.

Tay, F. E., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. omega, 29(4), 309-317.

Taylor, J. W. (2000). A quantile regression neural network approach to estimating the conditional density of multiperiod returns. Journal of forecasting, 19(4), 299-311.

Varian, H. R. (2014). Big data: New tricks for econometrics. Journal of Economic Perspectives, 28(2), 3-28.

Varma, S., & Simon, R. (2006). Bias in error estimation when using cross-validation for model selection. BMC bioinformatics, 7(1), 1-8.

Xiong, R., Nichols, E. P., & Shen, Y. (2015). Deep learning stock volatility with google domestic trends. arXiv:1512.04916.