Volatility model against Deep Learning Techniques

Avoiding the one – size – fit – all model trap for predicting future financial volatility

Department of Economics and Finance

Master's Degree in Finance

Chair of Econometric Theory

Supervisor:
Prof. Paolo Santucci de Magistris

Co-supervisor:
Prof. Giacomo Morelli

Candidate:
Aurelio Mazzocco
ID: 743241

Academic Year 2022/2023

Abstract

We perform both in-sample and out-of-sample empirical analyses to model and predict the volatility of the SP500. The in-sample analysis helps us determine the model parameters, while the out-of-sample analysis assesses the model's accuracy. We evaluate the model's performance on both datasets using metrics such as MSE, RMSE, and MAE. The best-performing model is then employed to make forecasts, which aids in devising a strategy to capture the mean-reverting behavior of daily realized volatility. The in-sample dataset consists of 20,282 daily observations from 04/01/1928 to 09/10/2008, with the last 3,579 observations reserved as a validation set for the model. The out-of-sample dataset encompasses 3,580 daily observations spanning from 10/10/2008 to 29/12/2022. Our empirical findings suggest that machine learning models excel in predicting realized volatility for both the training and validation sets, primarily because of their capability to detect nonlinearity in the data.

Keywords: realized volatility, GARCH models, RNN models, stock market volatility.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to extend my heartfelt gratitude to Professor Paolo Santucci de Magistris, whose guidance, support, and unwavering availability were instrumental in my journey. Your dedication to my progress and success will always be remembered.

A special thanks to Pierluigi for his constant support and his valuable advice about everything. I'm really grateful that I met you.

To Michele e Rina, for being the people who always believed in me, supported and helped me and taught me what love is.

To Samuele, to be my greatest friend and the person I can always trust.

To Antonio, for being the big brother I did not have and who always looked after me. I love you infinitely.

To Vittorio e Mattia, to be the people I can most trust and rely on. Never change, always be that way.

To my Father, for allowing me to do what I always wanted to do and for supporting me as long as you could. I miss you every day, you should have been here.

1 Introduction

One of the most studied and used topics in the financial literature is volatility. Its use is extremely widespread and is considered in multiple contexts: risk management; asset allocation and portfolio optimisation; pricing derivatives and options and trading decisions are just a few of the myriad fields in which the study and analysis of volatility is fundamental.

First, it is important to point out that volatility is one of the main inputs used in the creation of portfolios. Forecasting the volatility of assets in a portfolio becomes crucial for assessing investment risk. Another key area in which volatility is one of the parameters that most heavily scrutinise is option pricing: markets, indeed, use volatility to price options. Another important area in which volatility is used is in all risk management activities. Since 1996 with the conclusion of the Basel Accords, volatility and its forecasting have assumed a fundamental role for all financial institutions worldwide. Hence the indispensability that market volatility has on the economy in general and the repercussions it can have as a whole. Policy makers often use market volatility as a barometer to determine whether the economy is vulnerable and in danger.

Given these factors, the importance of predicting volatility naturally arises. Despite the of extensive existence literature and various proposed models, there has been a recent increase in the utilization of machine learning techniques across diverse industries. Therefore, the objective of this thesis is to establish a connection between traditional financial literature and machine learning models in the context of volatility forecasting.

In the second part, we will provide the reader with an overview of volatility along with some stylised facts about the S&P 500 index, providing both descriptive statistics and evidence of the index under investigation, the ultimate goal of which will be to predict realized volatility. The third and fourth parts will examine the hypotheses that this work will attempt to prove together with a description of the models that will be implemented in the last part.

In order to understand the main object of our analysis, we have to define what realized volatility is. It refers to the measure of actual price movement of an asset over a specified period of time. Unlike implied volatility, which forecasts future volatility based on prices of options and is derived from theoretical models, realized volatility is calculated directly from historical price data.

Mathematically, realized volatility can often be calculated using the standard deviation of logarithmic returns. For a daily realized volatility, you'd look at the daily returns over a given period and compute the standard deviation of those returns.

In the context of financial markets, tracking realized volatility is essential for portfolio management, risk management, and derivative pricing ad mentioned before.

Ultimately, this work wants to to understand which of the models that will be implemented later is best suited to analysing and forecasting realised volatility itself.The

literature has shown that market volatility is predictable and time varying (Schwert and Seguin 1990 and French et al. 1987).

In some of the early work on volatility, Hsieh 1991 explores the attractiveness of chaotic dynamics is its ability to generate large movements which appear to be random providing one of the first estimate of the daily return constructed from a dataset of intra - day S&P500 returns not exploring the relation with the concept of quadratic variation. Moreover, Zhou 1996, utilizing high-frequency data, noted that it offers benefit of estimating volatility, as it enables its prediction for any desired time period promptly. The frequency of the sampling can affect the autocorrelation of the data. He also proposes a method to correct the bias created. Furthermore, Andersen et al. 2003 documented the impact on volatility of shocks due to macroeconomic news announcement, and the long - run dependence in RV time series. Finally, Barndorff-Nielsen and Shephard 2001 provided one of the first empirical analysis of realized volatility using non-Gaussian processes of Ornstein-Uhlenbeck type which offer the possibility of capturing important distributional deviations from Gaussianity and for flexible modelling of dependence structures.

RV became the benchmark against which every forecasting model would be compared. Andersen and Bollerslev 1998 highlighted this importance even further, which would later be taken up by Andersen et al. 2004. So far, the focus has been on realised volatility and how ARCH class models can be used to predict volatility itself. Several studies, however, have attempted different approaches to ARCH models in order to increase predictive power. Ghysels et al. 2006 moves away from the aforementioned models by implanting Mixed Data Sampling (MIDAS) regressions using combinations of estimated volatility measures with different time horizons and time frames. The MIDAS regression framework provides a means of examining whether the use of high-frequency data will inevitably result in improved volatility forecasts over different time intervals, offering a high degree of flexibility in estimating the forecast of volatility. Engle and Gallo 2006, instead, implemented a multivariate extension of the multiplicative error model in order to predict multistep volatility. Subsequently, some of the literature focused on trying to understand if and how volatility was plagued by jumps and how to detect them: Andersen et al. 2007 showed how separating the diffusive component and the estimation of the jump half of the Quadratic Variation improves the prediction performance of the model.

One of the main features of volatility is its temporal dependence. In Andersen and Bollerslev 1997, the authors documented how the correlogram for the RV series exhibit a distinct hyperbolic decay from an integrated fractional process. By taking into consideration the long memory dependence of the volatility can improve the forecasting (Corsi 2009) as in Deo et al. 2006 where the author proposes a model to accommodate the behavior of the Realized Volatility using a regression to forecast the k - step ahead quadratic variation using simple OLS estimation using the coefficients to forecast out of sample volatility.

Hand in hand with the development of econometric models, in recent years the economics

literature has given increasingly more space to machine learning models (especially deep learning) with the aim of finding models that could better capture the non-linearity of volatility relationships. Hochreiter and Schmidhuber 1997 introduced a particular and innovative typology of Recurrent Neural Network which is successful in detecting long-term dependencies in the data called Long Short Term Memory (LSTM). LSTM models do not always improve forecasts compared to classical econometric models. Assaf et al. 2022 demonstrates that employing a single feature LSTM for predicting future realized volatility achieves comparable accuracy to GARCH models. Additionally, the results highlight that a stacked LSTM, configured with a multivariate input encompassing multiple assets and a lagging period exceeding one day, significantly enhances volatility prediction accuracy. The stacked multivariate LSTM architecture effectively captures intricate patterns within the time series data of asset prices, offering a superior alternative to statistical models for volatility modeling and prediction. Notably, the proposed multivariate LSTM architecture demonstrates faster and more accurate daily volatility modeling, making it particularly suitable for intra-day modeling, especially in high-frequency trading environments. Fjellström 2022 used a LSTM to predict stock price on the Stockholm OMX30: the LSTM ensemble yields improved average daily returns and greater cumulative returns over time. Furthermore, the LSTM portfolio demonstrates lower volatility, resulting in higher risk-return ratios.

In addition to LSTM models, other Deep Learning models that have been considered in the literature are models called Convolutional Neural Network (CNN). Aradi et al. 2020 revealed that deep learning techniques can effectively leverage the increased volume of data, resulting in improved outcomes for both value and direction forecasts. Furthermore, various measures assessing the quality of predictions also indicated the superior performance of deep learning models. These findings suggest that the broad nature of stock prices enables data expansion, which, in turn, empowers deep learning methods to surpass traditional time series models in the realm of financial forecasting, particularly in short-term scenarios. Moreover, Lu et al. 2021 implemented a CNN-BiLSTM-AM method which consists of convolutional neural networks (CNN), bi-directional short-term memory (BiLSTM) and attention mechanism (AM). CNN is used to extrapolate features from the input data; BiLSTM uses the extracted characteristics to predict the next day's closing price of the stock and AM is used to capture the influence of feature states on the closing price of the stock at different times in the past to improve the accuracy of the prediction. Vidal and Kristjanpoller 2020, instead, implemented a CNN-LSTM hybrid model on the gold volatility which, compared to normal GARCH and LSTM models, improves volatility forecast. the author also implemented a Model Confidence Set (MCS) which determines a significant improvement in the prediction of the previous model. Basing on the theoretical foundations outlined above, I formulate the following hypothesis:

H1 Machine learning models will have a better forecast than traditional econometrics due

to their better ability to capture the non-linearity of data.

H2 By leveraging the forecasted value from the optimal model, we might be able to devise a strategy that surpasses the performance of the S&P 500.

Fleming et al. 2003 showed that using volatility - timing investment decisions, yields substantial economic benefit and that a risk adverse investor is willing to pay 50 to 200 basis points annually to change from a daily - return - based estimate of the conditional variance - covariance matrix to one based on realised volatility. Moreover, Xiong et al. 2022, analyzing the crash risk of factors in Chinese market, built three target volatility strategies finding that using a dynamic target volatility strategy can improve the performance of most US factor portfolios.

2 Data

2.1 Theoretical Framework and previous work

Going forward with the discussion of the topic, we now proceed with the second section of the work: outlining the theorethical framework and the previous works on the matter. In order to understand the theoretical framework of our work, it's important to highlight what is a random variable. A random variable is a measurable function X: $\Omega -> E$ from a probability space Ω as a set of possible outcome to a measurable space E. The probability that X takes on a value in a measurable set $S \subseteq E$ is written as the following:

$$P(X \in S) = P(\{\omega \in \Omega | X(\omega) \in S\})$$

In many cases, the random variable X is real - valued (i.e. $E = \mathbb{R}$. When the image of X is countable, we can define a **discrete random variable**; on the other hand, the image is uncountably infinite and the random variable is continuous. In our analysis, the main focus will be on the first and second moment of the distribution.
The broad availability of data on time series on financial data has led to a widely spread literature on the forecastability of volatility. Before analysing the current state of the literature, we must distinguish what is the variance (and consequently volatility). The population variance is the defined in the following way:

$$\mathbb{E}[r_t - \mathbb{E}(r_t)]^2 = \mathbb{E}[r_t^2] - \mathbb{E}[r_t]^2, \tag{1}$$

where $\mathbb{E}[r_t]$ is the first and $\mathbb{E}[r_t^2]$ is the second moment of the distribution of the random variable r_t. The volatility, financially speaking, is the standard deviation of the returns in a given time horizon and can be used a measure of risk of an asset. Unfortunately, it can be observed but can be measured ex - post. One way to capture it, is with the Equation (2):

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

where r_t is the return and μ its mean over a period T. It measures the dispersion of the returns across the mean.

Computing historical variance (and volatility), we need to estimate the mean return of the sample. Even if the sample estimator is unbiased, the estimate of the mean return could be noisy. Even increasing the sample size, if a security experiences a downtrend in its return, it conflicts with the non-negativity assumption on ex-ante expected return. Assuming the mean return to zero, we can refer to this type of volatility as realized volatility.

To understand the basic rationale behind the realized volatility approach, we need to

define a simplified setting with continuously compounded return driven by a simple time - invariant Browian motion:

$$ds(t) \ = \alpha dt + \sigma dW(t), 0 \leq t \leq T, \tag{3}$$

where α and σ denote the constant drift and diffusion coefficients scaled to the unit time interval. Generalizing the framework, we can define a continuos - time diffusive framework without price jump and frictionless where the asset's logarithmic price process **s** is a semimartingale:

$$ds(t) = \mu(t)dt + \sigma dW(t), 0 \leq t \leq T, \tag{4}$$

where W is a standard Brownian motion process, $\mu(t)$ and $\sigma(t)$ are predictable process, $\mu(t)$ is of finite variation while $\sigma(t)$ is square integrable and strictly positive. The continuosly compounded return over a period of time t-k to t, 0 ¡ k $\leq$ t is:

$$\mathbf{r}(t,k) = \mathbf{s}(t) - \mathbf{s}(t-k) = \int_{t-k}^{t} \mu(\tau)d\tau + \int_{t-k}^{t} \sigma(\tau)dW\tau, \tag{5}$$

and its *quadratic variation* is:

$$\mathbf{QV}(t,k) = \int_{t-k}^{t} \sigma(\tau)d\tau, \tag{6}$$

Equation (6) evidences that the innovations to the mean component $\mu(t)$ do not affect sample path variation of the return. The return quadratic variation can be approximated arbitrarily well by the cumulative squared return process. Given the following partition: $\{t-k+\frac{j}{n}, j = 1, ..., n*k\}$ on the [t - k, t] interval. Then, the realiazed volatility of the logarithmic price process is :

$$\mathbf{RV}(t,k,n) = \sum_{j=1}^{n*k} r\left(t-k+\frac{j}{n}, \frac{1}{n}\right)^2 \tag{7}$$

Semimartingale theory ensures realized volatility measure converges in probability to quadratic variation (Equation (6)) as the sampling frequency n increases:

$$\mathbf{RV}(t,k,n) \xrightarrow{p} \mathbf{QV}(t,k) \quad \text{as} \quad n \to \infty \tag{8}$$

The formal connection between realized volatility measured on high - frequency returns and quadratic variation was developed by Andersen and Bollerslev 1997 as they explored the empirical properties of return volatility. The distributional results can be generalized as $n \to \infty$ in the following equation:

$$\sqrt{n*k}\left(\frac{\mathbf{RV}(t,k,n)-\mathbf{QV}(t,k)}{\sqrt{2IQ(t,k)}}\right)\xrightarrow{p}N(0,1) \tag{9}$$

where IQ (t,k) = $\int_{t-k}^{t}\sigma^4(\tau)d\tau$ is the integrated quarticity and IQ as independent from the Gaussian ditribution as shown in Barndorff-Nielsen and Shephard 2002.

Many of the work based on the prediction of the realized volatility used high - frequency data, by exploiting the asymptotic properties of it, to predict next realized volatility. Our analysis would consider closing price of the S&P500 instead of considering intra day observation. Generally speaking, the standard deviation of returns is defined in the following way:

$$\sigma_n^2=\frac{\sum_{t=1}^{n}(r_t-\hat{r})^2}{n-1}, \tag{10}$$

where $r_{i,t}^2$ is the i- th daily logarithmic return in a day t. Equation (5) is the most common volatility proxy (As in the book of Brooks 2014). Assuming $\hat{r}$ to be zero, the Equation (5) can be written as the following:

$$\sigma_n^2=\frac{\sum_{t=1}^{n}(r_t)^2}{n-1}, \tag{11}$$

In our analysis, the main object of interest would be the realized volatility of one day which can be calculated with the following equation:

$$\sigma_n=\sqrt{r_t^2}, \tag{12}$$

with this approach, we square the log return of the index in that day and then take the square root since this method is a widely spread and used in the literature.

2.2 Data download and features

The data, which will be used with the objective of forecasting realised volatility, were downloaded from Wharton Research Data Server (WRDS) and from Yahoo Finance using the library *yfinance*. The analysis would be conducted about the SP500 Index starting from 31 december 1927 to 31 december 2022. We downloaded 23937 daily observations of prices of the index.

We construct the daily returns in the following way:

$$R_t\ =\frac{P_t-P_{t-1}+Div_t}{P_{t-1}}, \tag{13}$$

The log returns are constructed as the following:

$$r_t\ =log(\frac{P_t+D_t}{P_{t-1}}), \tag{14}$$

where r_t is the daily return, P_t is the price at day t and P_{t-1} is the price at day $t-1$ of the previous day. In figure 1 it's reported the closing price for the SP500. In figure 2, we plot the returns time series for both log return and return with their empirical distribution. To give a better comprehension of the returns, we plot also the return in the last 3 years in figure 3. We can evince from the figure 3 and from the distribution of the returns that the last 3 years have seen lower extreme events since the tails have lower peak.

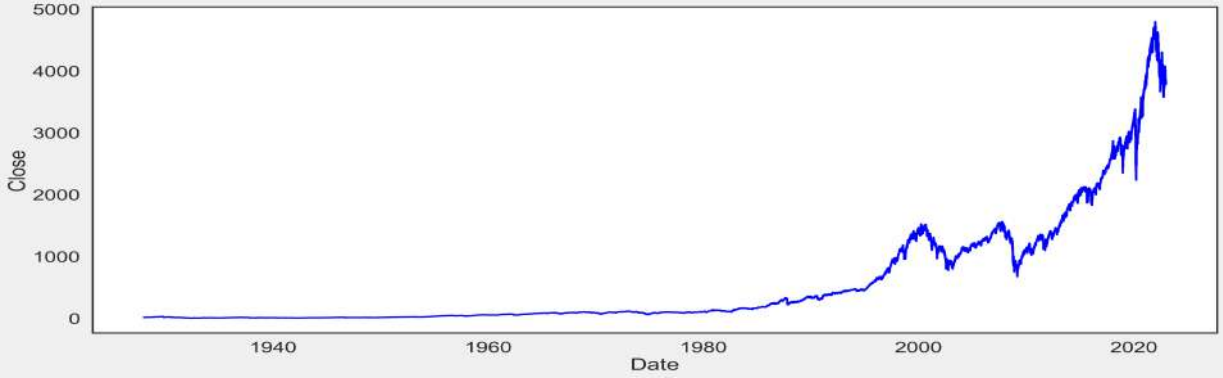
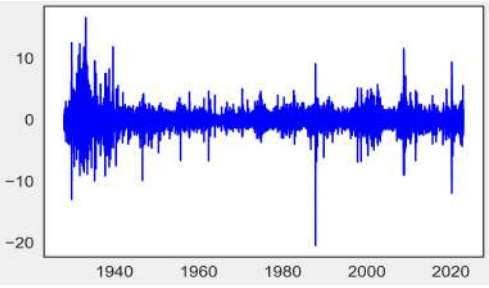
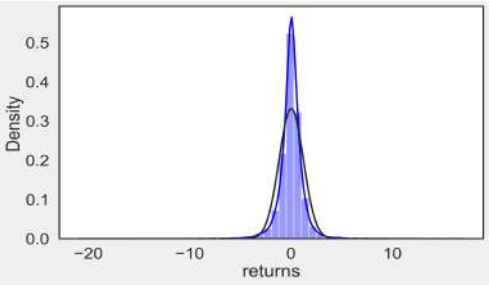


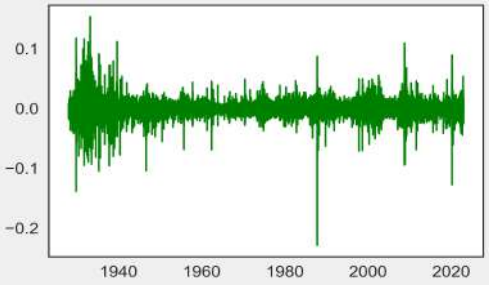
Figure 1: Closing Price of the SP500 on daily base.



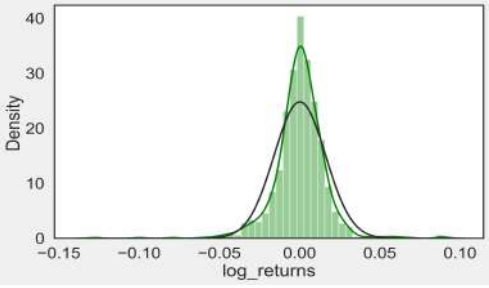
(a) Returns on SP500.



(b) Empirical dist. of ret on SP500.

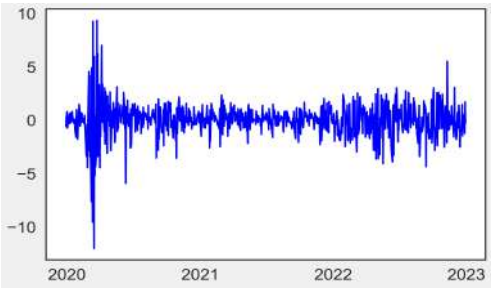


(c) Log Returns.

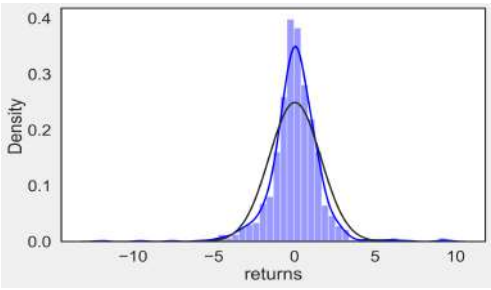


(d) Empirical dist. of log ret on SP500.

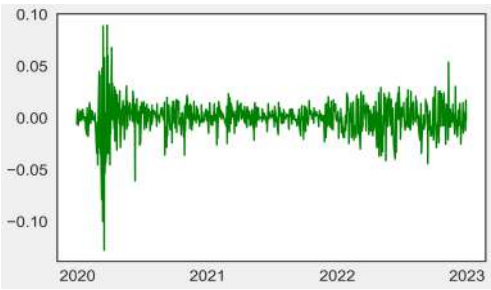
Figure 2: *In Figure 2 are reported the values of returns and log returns and their empirical distribution derived from data. The empirical distributions are compared to the normal distribution (in black)*



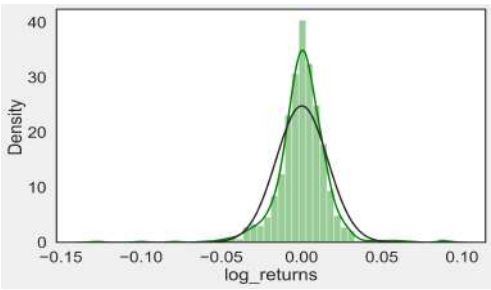
(a) Returns on SP500.



(b) Empirical dist. of returns on SP500.



(c) Log Returns.



(d) Empirical dist. of log ret. on SP500.

Figure 3: *In Figure 3 are reported the values of returns and log returns and their empirical distribution derived from data. The empirical distributions of the last 3 years are compared to the normal distribution (in black).*

Numerous studies have highlighted many evidence regarding the volatility. The baseline assumption to model returns and to try to predict them is to assume the returns are normally distributed. From Figure 2 and 3 and from the descriptive statistics, comparing the empirical distribution with the normal density function, the actual return distribution has fatter tails and negative skewness meaning that large moves tend to occur more frequently than expected. The presence of fatter tails is even more evident from QQ Plot in figure 4.

In the Figure 5 and 6 are reported the correlation plot of 20 lags of daily and monthly returns and their absolute values. The main goal of many model was to predict the next period/one step ahead return for one particular asset or index. The empirical evidence, widely expressed in the literature, shows that returns cannot be predicted by only looking at their past values. Many researchers, as Fama et al. 1969, founded lack of predictability using past returns as independent variable with weak results. Instead, absolute returns are more likely to captured. There's strong evidence that absolute and square returns are serial dependent suggesting that the scale of returns changes in time. Daily returns show little autocorrelation which goes to zero after lag 1. Monthly returns show higher autocorrelation. Monthly returns dependency decays slowly over time as the absolute values ($Corr(|R_{t+1}|, |R_{t+1-\tau}|) > 0$ shows the same effect as $Corr(R^2_{t+1}, R^2_{t+1-\tau}) > 0$).

The possibility to be able to predict next period's return with the previous return (at least in magnitude), is reflected also in volatility and as the name of *volatility clustering*. As can be seen graphically, there's presence of long period of high/low volatility. As mentioned

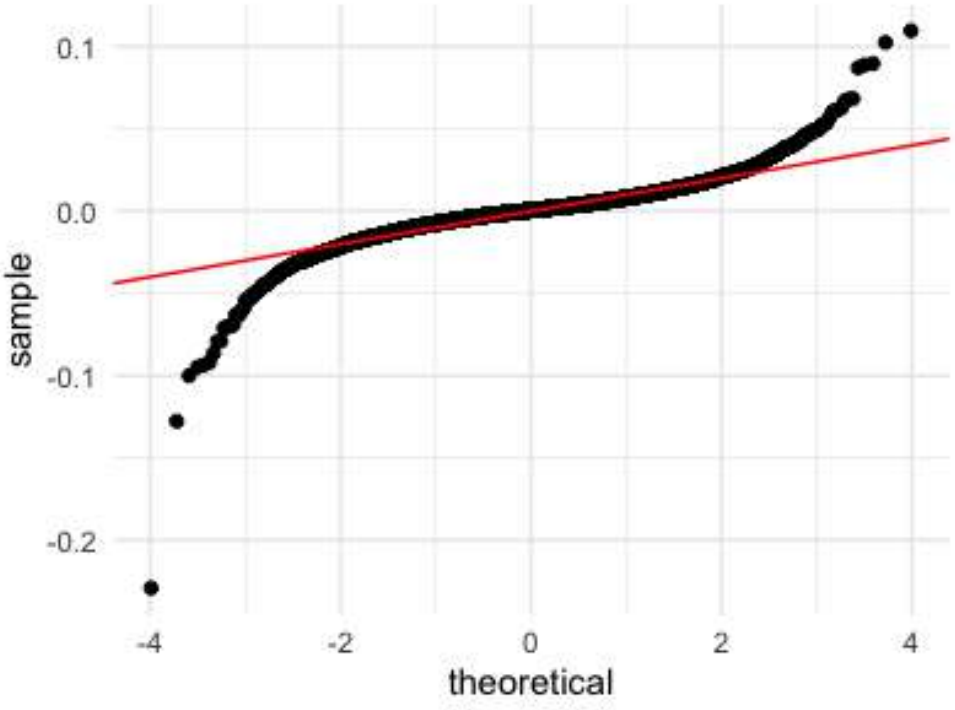
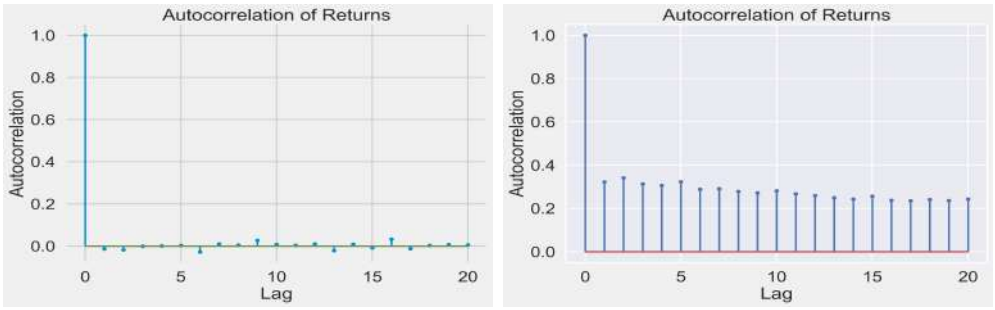


Figure 4: In Figure 4 is reported the QQ - Plot of the distribution of the empirical data of return.

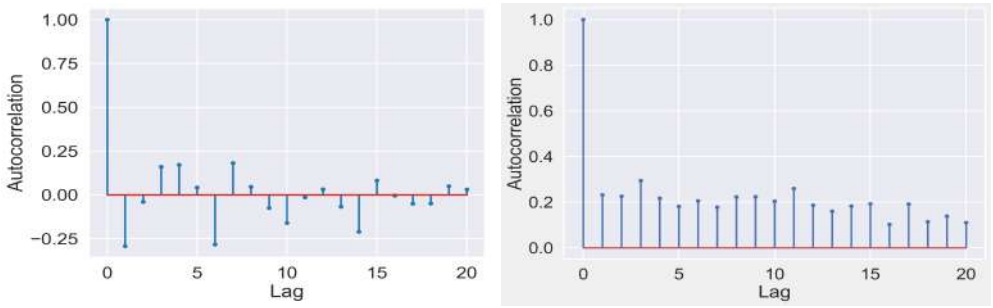
before, the autocorrelation of returns is insignificant at different lags. Mandelbrot 1971 exploits how the activity of arbitrageurs can be linked to the spectral whiteness of the returns as in figure 6.



(a) Autocorrelation of returns on SP500.

(b) Autocorrelation of absolute return on SP500

Figure 5: *In Figure 5 is reported the correlogram of the daily returns and the absolute value of daily returns in order to understand the stationarity and the decay of the autocorrelation.*



(a) Autocorrelation of returns on SP500.

(b) Autocorrelation of absolute return on SP500

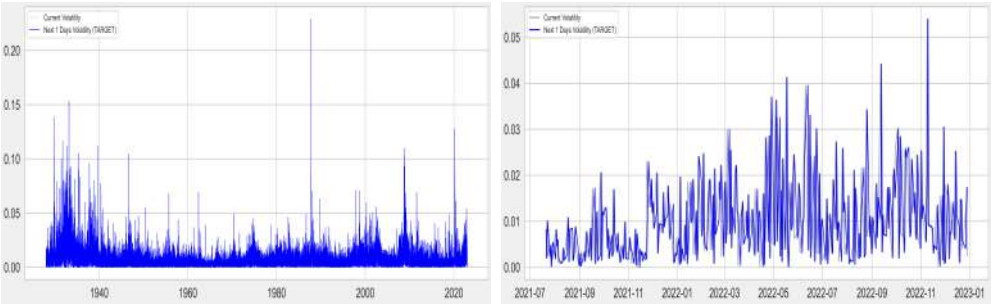
Figure 6: *In Figure 6 is reported the correlogram of the monthly returns and the absolute value of monthly returns in order to understand the stationarity and the decay of the autocorrelation.*

2.3 Evidence from studying volatility: the SP500

Time-series forecasting models are used to predict future values based on past observations. In the context of volatility forecasting, the "future" data refers to the volatility of a specific time period that is obtained by shifting the current volatility backward by a certain number of lag periods.

In our analysis, we consider a shift of 1 day ahead to forecast the realized volatility of the next day. This shift is essential to facilitate the training and usage of machine learning models. By shifting the variable, we enable the model to learn from historical data and make predictions for future volatility based on past patterns.

By incorporating the lagged volatility as the target variable, we can train forecasting models to learn the relationship between past volatility patterns and their corresponding future values. This enables the models to make predictions about future volatility based on historical information.



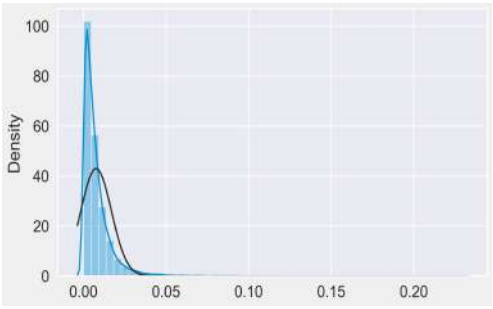
(a) Future and Current daily RV.

(b) Future and Current daily RV.

Figure 7: *In Figure 7 is reported the volatility object of our analysis. The shifting of the volatility would be used into machine learning model to create the target variable as the n - steps ahead of the volatility.*

Volatility is highly skewed to the right as could be seen in figure 10 with more period of

high volatility respect the one expected from the normal distribution.



(a) Distribution of volatility.

Figure 8: *In Figure 8 is reported the volatility object of our analysis.*

2.4 Descriptive statistics

Below there are a series of summary statistics to try to begin to better understand the context in which our models will operate.

	Open	High	Low	Close	returns	log returns	RV
count	23862.00	23862.00	23862.00	23862.00	23862.00	23862.00	23862.00
mean	564.17	587.75	580.52	584.36	0.03	0.001	0.007
std	938.620	932.73	921.51	927.48	1.20	0.012	0.009
min	0.000	4.40	4.40	4.40	-20.47	-0.23	0.000
25%	9.52	24.51	24.51	24.51	-0.46	-0.01	0.002
50%	40.77	102.21	100.62	101.45	0.048	0.00	0.005
75%	960.32	970.68	950.87	960.46	0.55	0.005	0.008
max	4804.51	4818.62	4780.04	4796.56	16.61	0.154	0.009

Table 1: Descriptive statistics of the data.

In Table 1, there are some summary statistics for opening and closing market prices and low and high prices are given. Returns and log returns of the SP500 are also shown. In the Table 1 is reported the Realized Volatility of 1 day.

2.5 Testing normality of return

Due to the non-normal nature of financial returns evident in the previous exhibit, it is crucial to perform statistical tests to examine the validity of the assumption of non-normality. One commonly used test for assessing the normality of data is the Kolmogorov-Smirnov test, originally introduced by Massey 1951. This test quantifies the statistical difference between the empirical distribution of the data and a theoretical distribution. However, due to its limited power, an alternative test called the Jarque-Bera test, introduced by Jarque and

Bera 1987, can be employed to assess the distribution's departure from normality. The Jarque-Bera test considers the third and fourth moments of the distribution and can detect deviations from a normal Gaussian distribution. Additionally, another test that can be utilized is the Shapiro-Wilk test, proposed by Shapiro and Shapiro and Wilk 1965, which examines the match between the sample distribution's skewness, kurtosis, and those of a normal distribution.

The (KS) test is a nonparametric test to quantify the distance between the empirical distribution function of the financial data and the cumulative distribution function of a reference distribution which is the one test. The KS tests the null that empirical and hypothesized distributions have the same cumulative distribution function against the alternative that they do not, using a statistic represented by the maximum difference in absolute value between the empirical cumulative distribution function $\hat{S}(x)$ and the theoretical one $S_n(x)$:

$$D^* = \max_x(|\hat{S}(x) - S_n(x)|).$$

From the above formula, can be seen that, if the above difference between the empirical and theoretical cumulative distribution functions tends to zero, there is no significant evidence against the null hypothesis of normality of the data. Another approach to evaluate whether a sample follows a normal distribution is the Shapiro-Wilk test.

This test, as the previous one, exploits the normality of the data. The test is obtained as the ratio between a linear combination of the sample order statistics and the usual symmetric estimate of the variance. The test statistic used in the Shapiro-Wilk test is defined as:

$$W = \frac{R^4\hat{\sigma}^2}{C^2S^2} = \frac{\sum_{i=1}^n(a_iy_i)^2}{\sum_{i=1}^n(y_i - \hat{y})^2}.$$

where $R^2 = m'V^{-1}m$, $C^2 = m'V^{-1}V^{-1}m$, $S^2 = \sum_1^n(y_i - \hat{y})^2$, $\hat{\sigma} = \frac{m'V^{-1}y}{m'V^{-1}m}$ and $a' = (a_1, a_2, ..., a_n) = \frac{m'V^{-1}}{(m'V^{-1}V^{-1}m)^{\frac{1}{2}}}$. Moreover, we can define m' as a vector of expected values of standard normal order statistics, $V = v_{i,j}$ is the corresponding **n**x**n** covariance matrix.

Last but not least is the Jarque-Bera test. The idea behind the test is to infer if the skewness and kurtosis of the empirical sample can be brought back from a normal distribution. The test is defined in the following way:

$$JB = \frac{n}{6}(s^2 + \frac{1}{4}(k-3)^2)$$

where n is the sample size, s is the skewness and k the kurtosis. Intuitively, higher value of skewness and kurtosis increases the value of the statistics. This ensure the possibility to capture any distortion of the empirical distribution within the theoretical one. Higher is the value of the statistics, higher the critical value and H_0 is rejected.

Test	Statistic	P-value
Jarque-Bera	2481.44	0.00
Kolmogorov-Smirnov	0.45	0.00
Shapiro-Wilk	0.98	0.00

Table 2: Summary of statistical tests

The p - value of all the tests is 0 we can reject the null hypothesis of normality of the data as evident even from the graph above.

3 Methodology

In the subsequent section, we will present the theoretical framework that will serve as the foundation for the subsequent code implementation. The analysis will employ two distinct frameworks: econometric models and machine learning models. For the econometric models would be taking into consideration the GARCH classes with variant (GJR - GARCH, E - GARCH). Instead, for the machine learning model would be taking into consideration Recurrent Neural Network (RNN), Long Short Term Memory (LSTM) and other typology of Neural Network as Convolutional Neural Network

In the results' section would be implemented the models and also particular variation of them aiming to check which model has a better perform compared to the other.

3.1 GARCH Model

The ARCH model differs from conditional time series models by accounting for the changing variance of the error terms over time. Unlike models with a constant unconditional variance, the ARCH model explicitly acknowledges the distinction between unconditional and conditional variance. This allows the conditional variance to vary based on past errors. The main issue with ARCH is that setting long lags might lead to negative coefficients which collid with the assumption of positivity of the variance. Bollerslev 1987 introduced a new type of model called *Generalized Autoregressive Conditional Heteroskedasticity* (GARCH) which allows for both longer memory and flexible lag structure since it is a generalization of the ARCH introducing an autoregressive component in the variance term allowing the *ith* conditional variance to be function of the past conditional variances.

In our analysis, we can define a stationary time series $\{r_t\}_{t=1}^{T}$ with $\mathbb{E}[r_t] = \mu$ and $\text{Var}[r_t] = \sigma^2$ not time varying. The empirical evidence suggests that the scale of return to log return implies that the variance is time-varying. Therefore, we can define the conditional mean as $\mu_t = \mathbb{E}[r_t|\psi_t]$ and the conditional variance as $\sigma_t^2 = \text{Var}(\epsilon_t|\psi_{t-1}) = \mathbb{E}[(r_t - \mu_t)^2|\psi_{t-1}]$ where ψ_t is the information set available at time t. A GARCH(p,q) with $p \geq 0$ and $q > 0$ can be defined starting assuming that:

$$r_t = \sqrt{\sigma_t^2} z_t \quad z_t \sim D(0,1),$$

then,

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i r_{t-1}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2, \tag{15}$$

$$\sigma_t^2 = \omega + A(L) r_t^2 + B(L) \sigma_t^2, \tag{16}$$

where $\omega > 0$, $\sum_{i=1}^{p} \alpha_i$ and $\sum_{j=1}^{q} \beta_j$ are the persistence and $\sum_{i=1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j < 1$.

In this analysis would be considered as a baseline model for the GARCH the GARCH(1,1) model. The realized volatility of financial returns would be captured byt the following process:

$$\sigma_t^2 = \omega + \alpha r_{t-1}^2 + \beta \sigma_{t-1}^2, \tag{17}$$

where ω is the constant term of the relation which allows to understand and catch the long term behavior of the process, α is the parameter which exploits the impact of the innovation term and β is the autoregressive parameter of the relation.

3.2 Asymmetric GARCH Models

Volatility clustering is a phenomenon observed in financial markets where the magnitude of absolute (or squared) returns today can provide information about the magnitude of absolute (or squared) returns in the next period. It suggests that periods of high volatility tend to be followed by periods of high volatility, and vice versa. One important characteristic of volatility clustering is its symmetry. It implies that the impact of a positive return is equivalent to that of a negative return of the same magnitude. In other words, the direction of the return does not affect the volatility, only its magnitude does. However, GARCH models offer the flexibility to capture asymmetry in volatility. They allow for the possibility that positive returns may have a different impact on volatility compared to negative returns. This feature enables GARCH models to better capture the complex dynamics of financial markets, where the relationship between returns and volatility is not purely symmetric.

3.2.1 GJR - GARCH

On the economic side it's arguable that the sign of return does matter: a stock which register a negative return has the firm's equity decreased. Assuming a constant value of the *Debt*, the leverage of the firm increases making the firm more risky. The leverage effect implies that a negative return increases volatility more than a positive one of the same size. Leverage implies that volatility dynamics are asymmetric since it reacts differently to positive and negative shocks.

Glosten et al. 1993 introduced a new typology of GARCH taking into account the this leverage effect. We can define the GJR-GARCH(p,q) model as in eq. 17. Assuming:

$$r_t = \sqrt{\sigma_t^2} \quad z_t \sim D(0,1),$$

then,

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} (\alpha_i + \gamma_i \mathbb{I}_{t-i}) r_{t-1}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2, \tag{18}$$

$$I_{t-k} = \begin{cases} 1 & if r_t^2 < 0 \\ 0 & otherwise \end{cases}$$

$$\omega > 0; \alpha_i \geq 0; \beta_j \geq 0; \alpha_i + \gamma_i \geq 0,$$

Moreover,

$$\sum_{i=1}^{p} \alpha_i + \frac{1}{2} \sum_{i=1}^{p} \gamma_i + \sum_{j=1}^{q} \beta_j < 1,$$

As in the previous model, we are going to examine a baseline GJR - GARCH (1,1) define in the following way:

$$\sigma_t^2 = \omega + \alpha r_{t-1}^2 + \gamma \mathbb{I}_{t-1} r_{t-1}^2 \beta_{\sigma t-1}^2, \tag{19}$$

with $\omega, \alpha, \beta > 0$. The idea behind the following model is that for positive value of returns, we have exactly the GARCH model, for negative value, instead, we obtain a modified version of GARCH with a slope equal to $\alpha + \gamma$. The model would be the following:

$$\sigma_t^2 = \begin{cases} \omega + \alpha r_{t-1}^2 + \beta_{t-1}^2 & if r_{t-1} > 0 \\ \omega + (\alpha + \gamma) r_{t-1}^2 + \beta_{t-1}^2 & if r_{t-1} \leq 0 \end{cases}$$

A slight different version of the GJR - GARCH model is the TARCH model that would be implemented into the empirical part.

3.2.2 E - GARCH

In this case you don't have to specify the distribution of return since the E-GARCH, taking the log of σ, ensure the positivity of it leaving the possibility to choose the distribution of the innovation. A generic E-GARCH(p,q) model can be define in the following way:

$$r_t = \sqrt{\sigma_t^2} \quad z_t \sim D(0,1),$$

$$ln(\sigma_t^2) = \omega + \sum_{i=1}^{p} \alpha_i (|r_{t-1}| + \gamma_i r_{t-1}) + \sum_{j=1}^{q} \beta_j ln(\sigma_{t-j}^2), \tag{20}$$

In our analysis would be implemented a E-GARCH(1,1) as in eq. 20:

$$\ln(\sigma_t^2) = \omega + \alpha \left(\frac{|r_{t-1}|}{\sigma_t} - m \right) + \gamma \frac{r_{t-1}}{\sigma_{t-1}} + \beta \ln(\sigma_{t-1}^2), \tag{21}$$

where $\beta \in (0,1)$ and $m = \mathbb{E}[z_t]$.

3.3 Machine Learning Model

Neural networks (NN) are mathematical models inspired by the structure of biological neural networks, aiming to replicate the behavior of human neurons. The term "neural network" often refers to the widely used "single hidden layer back-propagation network" or "single-layer perceptron." In contrast to traditional econometric models, NN offer several compelling features.

One key advantage is their ability to identify nonlinear patterns in the data without making any assumptions between input and output variables. This makes them highly versatile for various applications. NN can also serve as effective approximators of complex econometric models, providing valuable insights into the underlying relationships in the data.

Furthermore, NN possess the capability to generalize information learned from training data, allowing for accurate predictions on unseen data during out-of-sample analysis. This adaptability and capacity to handle complex, real-world problems make neural networks a powerful tool in modern data analysis and predictive modeling. In our analysis, the model would be applied to the realized volatility time series constructed as in Equation (15) using as feature the time series shifted by one day to the past and as predicted value the time series not shifted.

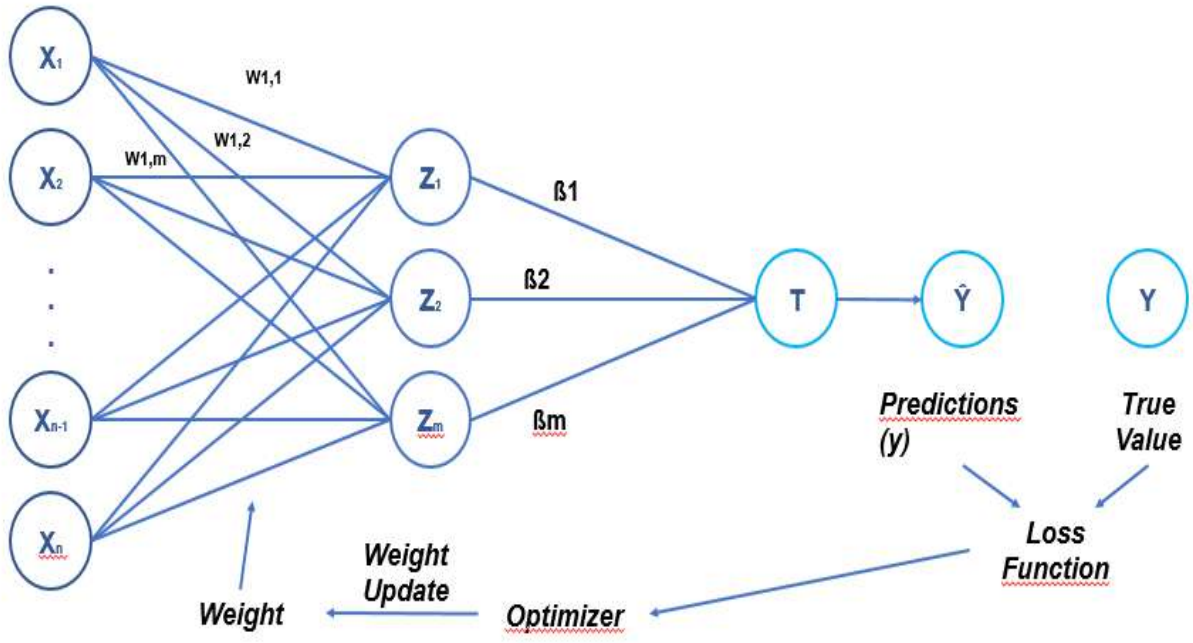


Figure 9: *Example of simple Neural Network fully connected which is an example of a model that would be implemented later.*

A neural network is a two - stage regression or classification model typically represented by the Figure 12.

In our analysis, we can define the following general equation to above diagram:

$$Z_m = \sigma(\alpha_{0m} + \alpha'_m X), \quad m = 1, \ldots, M, \tag{22}$$

$$T_k = \beta_{0k} + \beta'_k Z, \quad k = 1, \ldots, M, \tag{23}$$

$$f_k(X) = g_k(T), \quad k = 1, \ldots, M, \tag{24}$$

where $Z = (Z_1, Z_2, ..., Z_M)$ is a linear combination of the independent variable X computed with the weight alpha and $T = (T_1, T_2, ..., T_M)$ is a linear combination of the Z.

In our forecasting context, we are dealing with a regression problem, with a single output unit denoted as Y_1 which is the vector within the value of daily realized volatility. Additionally, we have $\mathbb{Z}$, which is the result of a linear combination of the input variables, and g serves as the activation function for the hidden layer. As mentioned previously, our prediction goal is a regression problem. Our focus would be to predict at n different time steps (1 days ahead) using m different features (in our context would be used the previous volatility. Let $\mathbb{X}_n$[1] $= [1, x_{1,n}, x_{2,n}, ..., x_{m,n}]$ be the (m+1) x 1 vector with 1 as constant term of the regression as the input variable for the n-th time step, $\mathbb{Z}_n = [z_{1,n}, z_{2,n}, ..., z_{q,n}]$ the q x 1 vector of hidden note and β the q x 1 vector of weights for the connection between the hidden layer and the output layer with $\mathbb{Z}$ as the matrix of the linear combination of the input variable. $\hat{y}_n$ would be the predicted target variable and can be expressed as the following:

$$\mathbb{Z}_n = g(\alpha_0 + \alpha' \mathbb{X}_n), \tag{25}$$

$$\mathbb{Y}_n = f(\beta' \mathbb{Z}_n + \beta_0), \tag{26}$$

with f and g, respectively, the activation function of output and hidden layer.

The most common activation function are following:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \quad \text{ReLU}(z) = \max(0, z), \quad \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$

The aforementioned functions are employed in both the hidden layers and output layers of the neural network (NN). Understanding the characteristics of these functions is essential for comprehending their role in the NN architecture.

The sigmoid function is confined within the range of 0 to 1, ensuring that the output is always within this interval. The ReLU (Rectified Linear Unit) function has a lower bound of 0, meaning that any negative input values are replaced with 0. On the other hand, the tanh function is bounded between -1 and 1, guaranteeing that its outputs fall within this range.

The rationale behind using these activation functions lies in the primary objective of

[1]The notation would always refers to vector

NN, which is to capture the non-linearity present in the data. By employing linear functions as the sole activation functions, the NN would be reduced to a simple regression model. Introducing non-linear activation functions, such as sigmoid, ReLU, and tanh, enables the NN to explore and represent the intricate relationships within the independent and dependent variables, allowing it to tackle more complex patterns in the data.

3.3.1 Gradient descent and backpropagation: fitting a neural network

The NN model has unknown parameters called weights and the idea is to seek values in order to fit the model on the training data in a proper way. We can define θ the complete set of weights, which consists of:

$$\{\alpha_{0m}, \alpha_m; m = 1, 2, ..., M\} \quad M(p+1) \quad weights, \tag{27}$$

$$\{\beta_{0k}, \beta_k; k = 1, 2, ..., K\} \quad K(M+1) \quad weights. \tag{28}$$

Another building block of the NN is the existence of a error function which is used to measure the fit of the model. In this case we can consider the sum - of - square errors as measure of the fit:

$$\mathcal{L}(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{i,k} - f_k(x_i))^2,$$

The general method for minimizing $\mathcal{L}(\theta)$ is known as *back propagation*, which involves gradient descent. Due to the structure of the model, the gradient can be efficiently computed using the chain rule for differentiation, performing both forward and backward sweeps over the network and keeping track only of quantities local to each unit. In appendix would be explain the derivation of equation 28:

$$s_{m,i} = f'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{k,m} \delta_{k,i}, \tag{29}$$

is the back - propagation equation. The idea is that in the forward pass, the current weights are fixed and the predicted values are computer from the eq. 28. In the backward pass, the errors $\delta_{k,i}$ are computed and then back propagated via eq. 28 to obtain errors $s_{m,i}$.

The main advantages of using back propagation is the simple local nature of the algorithm. In the algorithm, each hidden unit passes and receives information only to and from units that share a connection within them. The learning rate γ_r for batch learning is usually taken as constant or optimized bu a line search aiming to minimize the error function within each iteration. This learning rate is a stochastic optimization (Robbins and Monro 1951) which ensure convergence if $\gamma_r \to 0, \sum_r \gamma_r$ and $\sum_r \gamma_r^2 < \infty$. Gradient descent with backpropagation does not guarantee finding the global minimum of the error function but rather a local minimum; it may encounter challenges crossing plateaus in the error

function landscape due to the non-convexity of error functions in neural networks. While this non-convexity was once considered a significant drawback, some researchers argue that it is not always problematic in practical problems.

Normalization of input vectors is not mandatory for backpropagation learning, but it could potentially enhance performance. However, backpropagation does require knowing the derivatives of activation functions at the time of network design.

3.3.2 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a specialized type of neural network designed to handle time series or sequences of data. Conventional feedforward neural networks are suitable for independent data points. However, when dealing with data points that exhibit dependencies or temporal order, modifications are required in the neural network to accommodate these dependencies. This is where RNNs come into play, as they are well-suited to capture and process sequential information in the data. RNNs allow data to propagate not only from inputs to output but also from hidden layers.

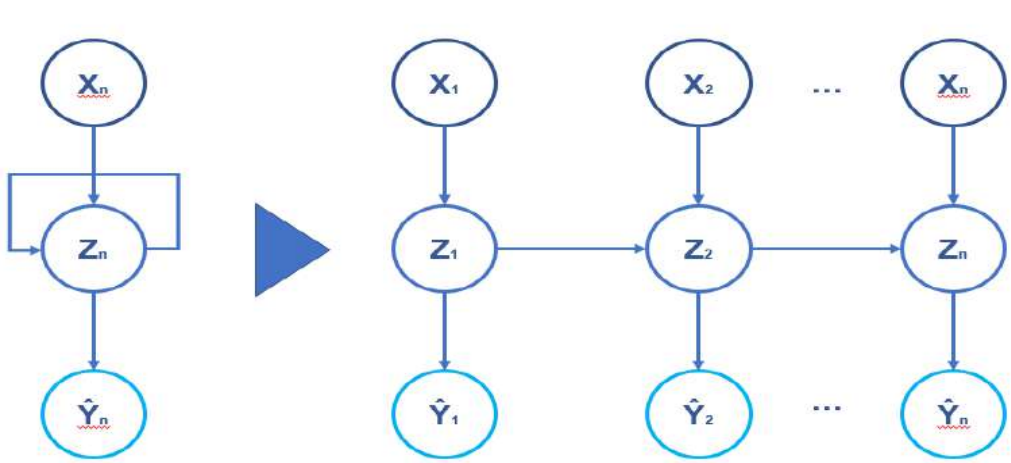


Figure 10: *Example of simple Recurrent Neural Network.*

In figure 13 it's shown how a RNN model works: $\mathbb{Z}$ depends on the arrows. It not only depends on the past time - step, but effectively all past time steps through the previous hidden representation. It's possible to define the following equation to build a RNN:

$$\mathbb{Z}_n = g(\alpha_0 + \alpha' \mathbb{X}_n + \gamma_0 + \gamma' \mathbb{Z}_{n-1}), \tag{30}$$

$$\mathbb{Y}_n = f(\beta' \mathbb{Z}_n + \beta_0), \tag{31}$$

Where g and f are, as before, the activation function of the hidden and output layer.

Compared to the NNs of the previous section, the vector $\mathbb{Z}_n$ is not only a combination of the input but also contains value of the hidden node where γ acts as a connector between

the vector $\mathbb{Z}_n$ and $\mathbb{Z}_{n-1}$. Since RNN models are very similiar to the NNs, they can be trained with backpropagation too.

In the context of neural networks, when computing the gradient with respect to the previous hidden state (the downstream gradient), the upstream gradient flows through the tanh non-linearity and is multiplied by the weight matrix. This process is repeated at every time step as the downstream gradient flows back across time steps. However, this approach has some drawbacks: firstly, the repeated multiplication by the weight matrix can cause the gradient to be scaled up or down, depending on the largest singular value of the matrix. This can lead to issues like exploding gradients if the singular value is greater than 1 or vanishing gradients if it is less than 1. Secondly, the gradient passes through the tanh non-linearity, which saturates at the extremes. As a result, the gradient can become close to zero after passing through the non-linearity, making it difficult for the gradient to effectively propagate across long sequences. This leads to inefficient optimization.

Although there is a technique to address the exploding gradient problem by clipping the gradient if it exceeds a certain threshold, it does not fully resolve the limitations of RNNs for handling long sequences effectively.

As a consequence, the shortcomings of RNNs in handling long sequences have motivated the development of more advanced architectures like Long Short-Term Memory (LSTM), which mitigate the issues of vanishing and exploding gradients and have proven to be more effective in handling long sequences in practice.

3.3.3 Long-Short-Term-Memory

A specific variant of Recurrent Neural Network is the Long-Short Term Memory (LSTM) model, introduced by researchers such as Hochreiter and Schmidhuber 1997 and Gers et al. 1999. The LSTM model possesses properties akin to a Universal Turing Machine Learner, making it an intriguing and powerful tool in machine learning. By providing a sufficient number of units to capture the system's state and employing an appropriate weighting matrix to regulate its evolution, the LSTM model has the remarkable capability to mimic the output of any computable function. This universality underscores the potential of LSTM networks to handle complex tasks and learn intricate patterns in sequential data.

Regarding RNNs, LSTM models possess the capability to make informed decisions at each time step during the training process. They can selectively choose which past information to forget, which information to incorporate, and what to output as results. Hochreiter and Schmidhuber 1997 developed an algorithm based on three different gate which, instead of the normal model, consider a forget gate in addiction to the input gate and the output gate.

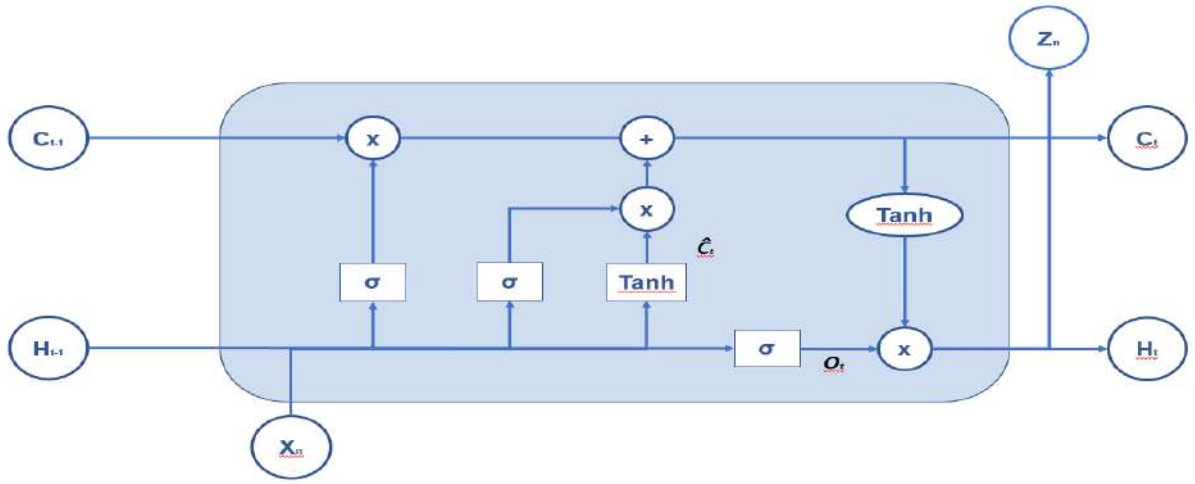


Figure 11: *Example of simple Long Short Term Memory Model.*

Because of its characteristic, LSTM is largely employed in tasks of sequence processing like analysing and forecasting time series: they are able to model in accurate manner the long and short range dependencies in the data. There are several variation of the model but the one will use in our analysis is the one presented in the Hochreiter and Schmidhuber 1997 work. The framework is the following:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{32}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{33}$$

In the LSTMs model, information flows both vertically from the input to the hidden layer, and horizontally evolving within the third equation below. The central component of LSTM is C_t, which acts as a memory accumulator, storing state information at time t. The gates f_i and i_i can be viewed as the forget and input gates, respectively. The forget gate f_i erases the memory of c_{t-1} based on the current input x_t, the previous state l_{t-1}, and the memory c_{t-1}. Since the sigmoid function is bounded between 0 and 1, the result of the linear combination within this layer act as a "classifier" of the information telling the model to forget the previous information (when the results is near 0) or to keep it (when the value is near 1). In contrast, the input gate i_i, utilizing the same information, reinforces or replaces the memory by activating a combination of x_t and l_{t-1}.

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{34}$$

The first layer (eq. 31) takes as input both current inputs and the past hidden state with a matrix of weights W to have a linear combination of them and then applying a sigmoid function to the weighted sum.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t, \tag{35}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{36}$$

The component c_t uses the hyperbolic tangent (tanh) activation function, and it includes a constant term as in the previous framework having the same mechanism as the previous layer. At each time step t, the memory is recomputed, and the LSTM produces the output o_t based on x_t, l_{t-1}, and the memory c_t.

The last layer in eq. 35 applyes a tahn function to the C_t layer and then pass it to the output one with output function f:

$$h_t = o_t \cdot \tanh(C_t) \tag{37}$$

$$\mathbb{Y}_t = f(\beta h_t + \beta_0), \tag{38}$$

The primary advantage of this architecture is that the memory c_t is refreshed under the control of gates, allowing the gradient to be limited to the last stage and preventing it from vanishing too quickly. This is a well-known limitation in RNN architectures, which can be effectively addressed by using the LSTM model.

LSTM learning function can be decomposed into multiple steps and each output of each step can be used as input for the next LSTM step. Sutskever et al. 2014 explore this possibility applying this "new" structure to sequence modeling problems.

3.3.4 Convolutional Neural Network

Another type of neural network that can be employed for time series forecasting is the Convolutional Neural Network (CNN). In structure, CNNs closely resemble regular neural networks, consisting of interconnected neurons with weights that can be learned from data. They receive input and produce output, which in our case can be utilized for predicting volatility, akin to the models described earlier.

Originally, CNNs were developed to enhance image recognition. In 1963, Larry Roberts first proposed the idea of extracting 3D geometric information from 2D perspective views. The underlying concept involves representing every image as a matrix of pixels. Formally, this can be represented by a function f that maps from R^2 to R, assigning values to each combination (x, y) that defines the matrix. By taking images as input, the model can encode several properties into the network, reducing the number of parameters required to identify the image. CNNs arrange their neurons in three dimensions: width, height, and depth. Each layer transforms its 3D input volume into a 3D output volume of neurons using activation functions within the layers.

The main distinction between RNNs and CNNs lies in the implementation of three new types of layers that the model can use to improve the forecasting of the target variable. Specifically, CNNs utilize convolutional layers, pooling layers, and fully connected layers. These specialized layers enable the CNN to better extract meaningful features from the

data, making them particularly effective for image recognition and other tasks where local patterns play a significant role.

The convolutional layer uses filters that perform convolution operation as it scan the input I (the vector/matrix of the time series) defining the hyperparameters including the filter size F and the stride S. The resulting output is a activation map. The pooling layer, instead, is a downsampling operation which is applied after a convolutional layer which does some spatial invariance. The fully connected layer operates on a flattened input where each input is connected to all neurons.

For the convolutional layer, we have a NxN square neuron layer which is followed by the aforementioned layer. We have a mxm filter ω which allow us to define a output layer of dimension (N-m+1) x (N-m+1). In order to compute the pre - nonlinearity input to some unit $x_{i,j}^l$ in our layer, we need to sum up the contributions from the previous layer cells:

$$x_{i,j}^l = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{a,b} y_{(i+a)(j+b)}^{l-1}, \tag{39}$$

and then applying the activation function to the output:

$$y_{i,j}^l = f(x_{i,j}^l),$$

where $x_{i,j}^l$ is the vector of input and $y_{i,j}^l$ is the output and f is the activation function.

The pooling layer is not a part of the model used to learn from the data: they only operate a selection of k x k region of the input returning a single value which is the maximum of that particular region. Having a NxN input, they return a $\frac{N}{k}$ x $\frac{N}{k}$ layer. Also for the CNNs, the model works with back propagation.

3.4 Scaling variables and Data Splitting

In order to understand which model would be the optimal one in predicting volatility, we need to split the dataset into three different subset. The three sample are named training, validation and test set and they are used by the model to train and then to choose which model perform the best into data that the model hasn't seen before.

We split the dataset into training, validation and test set, with a rule of 70%, 15% and 15% of the dataset which correspond to the following date:

- from 1928-01-04 to: 1994-07-26 (16703 days)
- from: 1994-07-27 to: 2008-10-09 (3579 days)
- from: 2008-10-10 to: 2022-12-29 (3580 days)

Here the visualization of the data split:

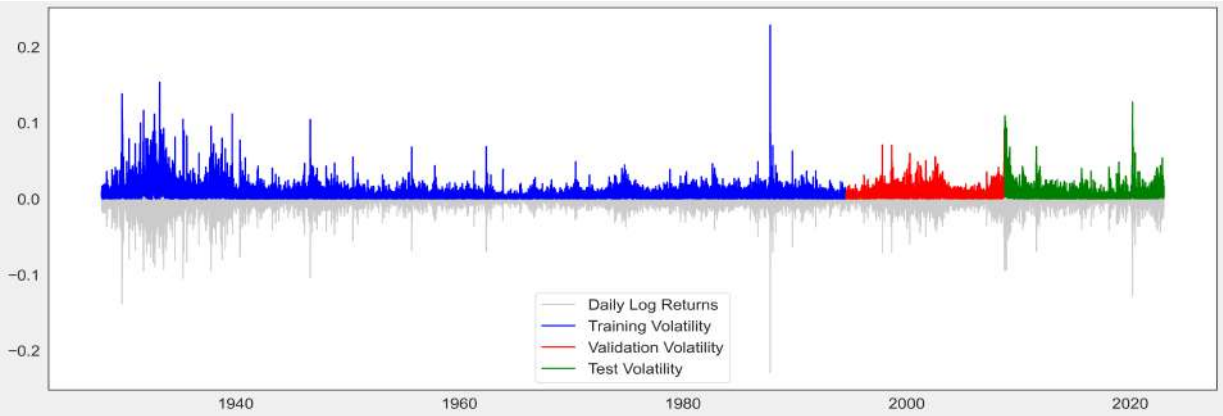


Figure 12: *In Figure 12 is reported the Realized volatility before scaling with the Equation 40.*

This approach ensures that our model undergoes validation on a dataset as extensive as the test set, capturing at least one major disruptive event in each set, such as the Great Depression, the 2008 financial crisis, and the recent pandemic-induced downturn. Moreover, an important features of machine model is to scale the independent variable. Scaling variables is a critical preprocessing step that has numerous important advantages in the context of machine learning models. This procedure, which involves converting a dataset's numerical properties to a standard scale, is crucial for improving the efficiency and performance of different algorithms. In the optimization phase, when machine learning algorithms look for the model parameters that reduce error, scaling has a number of important advantages. We ensure that no characteristic dominates others due to disparities in their scales by scaling the variables. The optimization procedure converges more quickly as a result, hastening the training of the model and enhancing its prediction skills.

This procedure is a crucial step in getting data ready for machine learning models, to sum up. By harmonizing the features, we promote more efficient optimization, reduce inaccurate distance estimations, apply regularization consistently, and improve model performance as a whole. The whole process would be implemented only for the Machine Learning part since scaling returns could affect time series properties and affect GARCH model's predictabily power.

In our analysis and only for the machine learning analysis, we are going to scale the features in the following way:

$$\tilde{x}_{i,t} = \frac{2 * (x_{i,t} - min_t(x_{i,t}))}{max_t(x_{i,t}) - min_t(x_{i,t})} - 1, \tag{40}$$

Here is reported the realized volatility before and after the scaling and the full sample scaled and splitted into training, validation and test set.

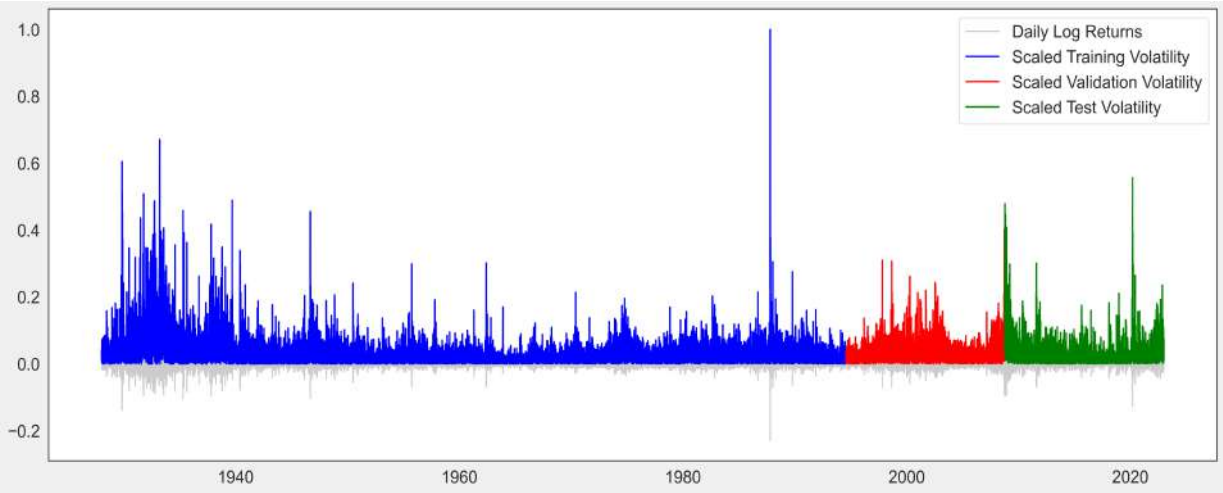


Figure 13: *In Figure 13 is reported the Realized volatility after scaling with the Equation 40.*

4 Results

In the upcoming section, we'll showcase the findings from our empirical study. Our primary objective is to evaluate the predictive capabilities of various models and leverage their forecasts to devise a portfolio strategy that outperforms the S&P 500 Index.

We initiated our investigation with the estimation of econometric models. Our study encompasses six distinct models: GARCH(1,1), GJR-GARCH(1,1), a TARCH(1,1), and a E-GARCH(1,1). The subsequent phase of our analysis delves into neural network modeling. Here, we'll implement a basic fully-connected ANN, a single-layer LSTM with 20 units, a bi-directional LSTM with two layers (32/16 units), and a hybrid CNN model featuring one convolutional layer followed by two LSTM layers. For the LSTM models, we'll test with two different window lengths for past observations: 14 days and 30 days.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t}^{n}(\hat{\sigma}_t - \sigma_t)^2},$$
$$\text{MSE} = \frac{1}{n}\sum_{t}^{n}(\hat{\sigma}_t - \sigma_t)^2,$$
$$\text{MAE} = \frac{1}{n}\sum_{t}^{n}|\hat{\sigma}_t - \sigma_t|,$$

Given the nature of the performance measure, the MSE, which squares the error, is sensitive to outliers. This is because squaring the error can amplify large deviations, especially outliers. As such, larger discrepancies will lead to a higher value in the measure, capturing the impact of these outliers. RMSE too but the outliers penalize less the value of the measure. MAE, instead, is the mean of the absolute value which implies that each error influences MAE in direct proportion making it less sensitive to the outliers.

4.1 GARCH Model

In-sample performance

We estimated each of the following GARCH model in the training sample from the 4th of January 1928 to the 26th of July 1994. We parameters we got are the following:

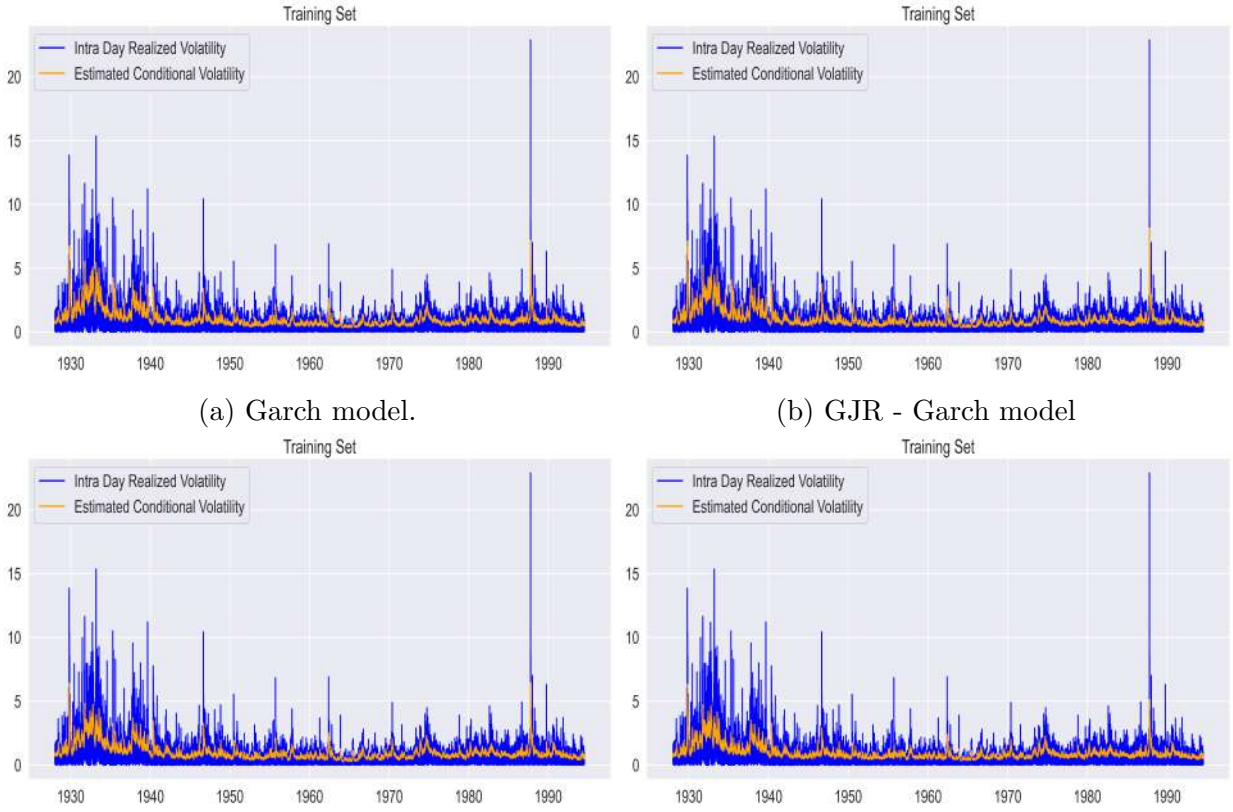
According to all the valuation criteria, the best performer of the Garch class of models, i.e. the model with minimum value of RMSE/MSE/MAE is the Tarch (1,1).

	GARCH		GJR-GARCH		TARCH		EGARCH	
Par.	Coef.	t-value	Coef.	t-value	Coef.	t-value	Coef.	t-value
ω	7.2946e-03	4.3	8.2670e-03	4.1	0.02	2.5	8.4038e-03	4.4
α	0.0894	7.0	0.0450	6.0	0.0501	5.2	0.165	9.1
β	0.91	78.9	0.917	82.6	0.914	47.6	0.989	448.7
γ	-	-	0.0714	4.9	0.0715	4.9	-0.0563	-5.8

Table 3: *This table shows the estimated parameters of the given GARCH models estimated on the Training set.*

Model	**Training RMSE**	**Training MSE**	**Training MAE**
GARCH(1,1)	0.802	0.642	0.556
GJR - GARCH(1,1)	0.802	0.643	0.556
E - GARCH(1,1)	0.783	0.613	0.545
TARCH(1,1)	**0.771**	**0.593**	**0.531**

Table 4: *Training performance metrics for the specified GARCH models with constant mean and normal distribution.*



(a) Garch model.

(b) GJR - Garch model

(c) E - Garch model

(d) Tarch model

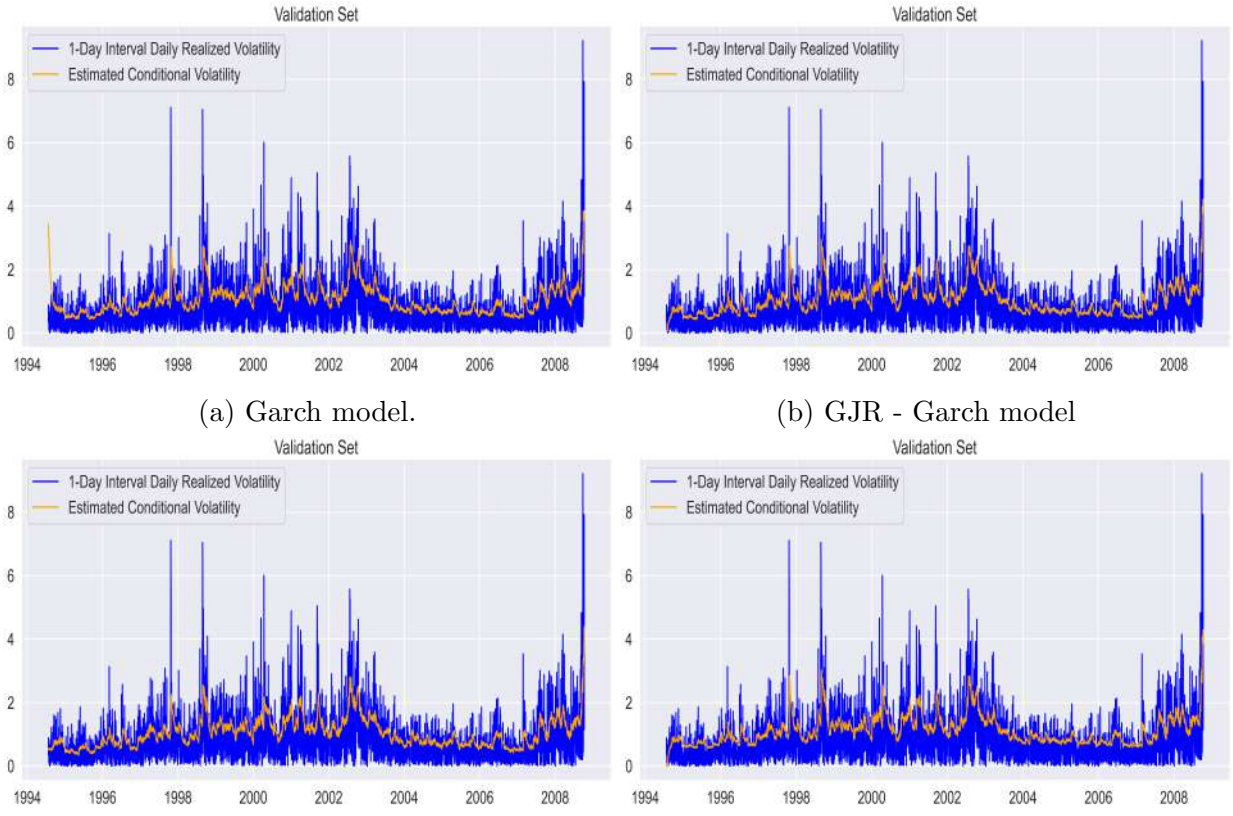
Figure 14: *In this figure is reported the estimation of the conditional volatility of the log return of the SP500*

Out-of-sample performance

The primary objective of evaluating in-sample performance (using the training set) is to discern which model most accurately aligns with the data. However, a significant limitation of this method is its focus solely on past performance. The essence of forecasting is to gauge a model's capability to predict forthcoming results. The GARCH family of models we assessed displayed the following performance:

Model	Validation RMSE	Validation MSE	Validation MAE
GARCH(1,1)	0.786	0.618	0.603
GJR-GARCH(1,1)	**0.759**	**0.576**	**0.581**
E-GARCH(1,1)	0.789	0.624	0.622
TARCH(1,1)	0.789	0.622	0.621

Table 5: Validation metrics for different GARCH models. All the models have mean zero and normal distribution



(a) Garch model.

(b) GJR - Garch model

(c) E - Garch model

(d) Tarch model

Figure 15: *In this figure is reported the conditional volatility estimated on the validation set.*

After the estimation and the analysis, we also perform a residual diagnostic. We perform the test on the best model we had considered which is the GARCH (1,1). The test we conduct is a ARCH - LM test.

Metric	Value
LM Statistics	89.26
LM Test p - value	0.000

Table 6: Validation metrics for different GARCH models. All the models have mean zero and normal distribution

At a significance level of 1%, the test suggests that there's no evidence against the null of no conditional heteroskedasticity in the standardized residuals.

4.2 Machine Learning Model

The methodology applied in the Econometric section is also adopted for Machine Learning. In this results segment, we will examine the predictive capacity of the machine learning model, maintaining the same training and validation set division. Usually, when working with Neural Networks, they are feed inputting all datapoints simultaneously. In our case is implemented a rolling lookback window to standardize input arrays and corresponding outputs. This entails that to procure future predictions for one time step, it's essential to collect the previous n datapoints, ranging from time step t $n + 1$ to t. For all the model is considered a past window of 14 and 30 days.
The LSTMs are constructed with the aid of Keras, an open-source Python library. In every model, we employed the "ReLu" activation function, maintaining it unchanged for the output layer. We utilized the "Adam" optimizer, a nuanced variant of the mini-batch gradient descent that modifies the learning rate with each iteration.

In-of-sample performance

As shown in table (7), based on our analysis, the Simple Linear Regression Fully Connected Neural Network emerges as the most suitable model, boasting the best scores across all three performance metrics. Interestingly, the use of both forward and backward windows doesn't seem to enhance the model's predictive performance.

One significant takeaway from these results is the superior predictive prowess of the ML model for such variables. The flexibility in volatility estimation provided by ML models contrasts with econometric models that often smooth out daily realized volatility. ML models aptly capture the mean-reverting nature of volatility, signifying a swift return to the mean following disturbances. However, a potential pitfall of these models might be overfitting, where they excel in explaining the training data but falter with the validation set.

Model	Training RMSE	Training MSE	Training MAE
Model 1, n_past=14	**0.0432**	**0.00187**	**0.0296**
Model 1, n_past=30	0.0435	0.00188	0.0301
Model 2, n_past=14	0.0434	0.00188	0.0300
Model 2, n_past=30	0.0434	0.00189	0.0301
Model 3, n_past=14	0.0432	0.00189	0.0301
Model 3, n_past=30	0.0434	0.00190	0.0300
Model 4, n_past=14, batch=64	0.0434	0.00188	0.0299
Model 4, n_past=30, batch=64	0.0433	0.00187	0.0300

Table 7: Training metrics for various models. Model 1 = Simple LR Fully Connected NN; Model 2 = LSTM 1 layer 20 units; Model 3 = 2 layers Bidirect LSTM (32/16 units); Model 4 = 1 Conv1D 2 Bidirect LSTM layers.

Out-of-sample performance

The out-of-sample metrics in the table provided in the Appendix reinforce our prior insights: ML metrics are lower, demonstrating a superior capability to grasp the smoothness of daily realized volatility.

Model	Validation RMSE	Validation MSE	Validation MAE
Model 1, n_past=14	**0.0508**	**0.00258**	**0.0302**
Model 1, n_past=30	0.0512	0.00262	0.0304
Model 2, n_past=14	0.0514	0.00264	0.0304
Model 2, n_past=30	0.0514	0.00263	0.0304
Model 3, n_past=14	0.0513	0.00264	0.0304
Model 3, n_past=30	0.0513	0.00263	0.0303
Model 4, n_past=14	0.0513	0.00263	0.0304
Model 4, n_past=30	0.0511	0.00261	0.0302

Table 8: Validation metrics for various models. Model 1 = Simple LR Fully Connected NN; Model 2 = LSTM 1 layer 20 units; Model 3 = 2 layers Bidirect LSTM (32/16 units); Model 4 = 1 Conv1D 2 Bidirect LSTM layers.

It is possible to find the value of the MSE and RMSE in both training and validation for different epochs. The forecast value for all models in the validation set is also given.

In figure (...) is reported the value for each epochs of MSE and RMSE:

In the Figure (...) is reported the predicted value of the best model within both classes of model with a focus for the last year in order to get a better comprehension of the prediction of the model.

4.3 Strategy Construction

After identifying the optimsal model for forecasting daily realized volatility, we devised a strategy to leverage the predictions from the test set. We crafted a strategy grounded on the mean-reverting nature of volatility. Initially, we calculated a 50-day rolling mean to

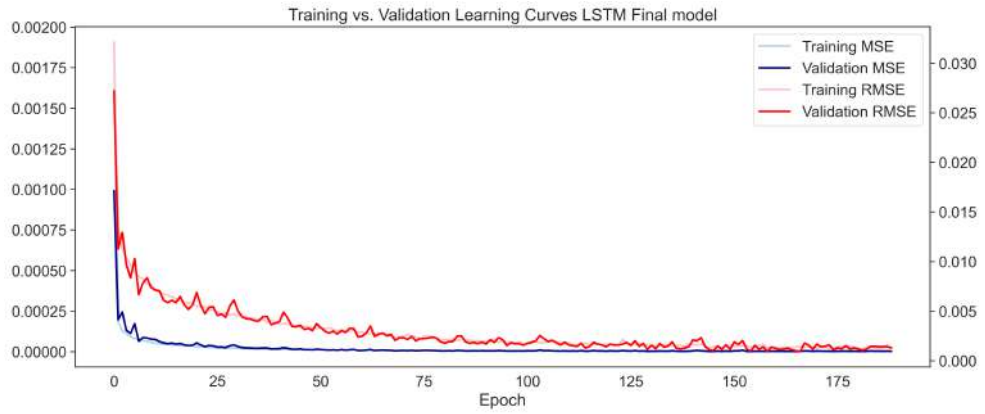


Figure 16: Simple Linear Regression Fully Connected Neural Network metrics.

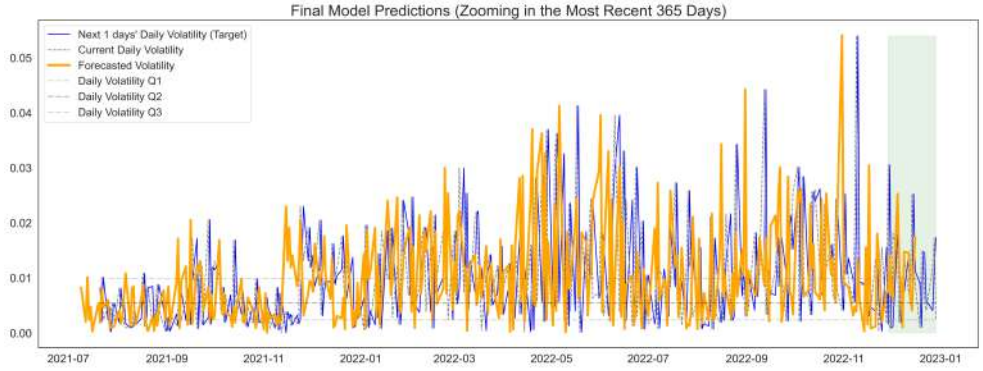


Figure 17: Simple Linear Regression Fully Connected Neural Network predicted value on test set.

trace the trend of the realized volatility. Upon establishing this, we compared the forecasted volatility value to its corresponding prior rolling mean to discern the likely behavior for that day. If the predicted volatility was below the rolling mean, we opted to buy the S&P500. Conversely, if it was higher, we decided to sell the index; otherwise, we maintained our position. Beginning with an initial investment of $10,000 at the start of the test set, the investment's value would now be $20,130.55. Investing the same amount in the S&P500 over the same period would have yielded $17,599.77. Thus, our strategy outperformed the S&P500 by 14.38%.

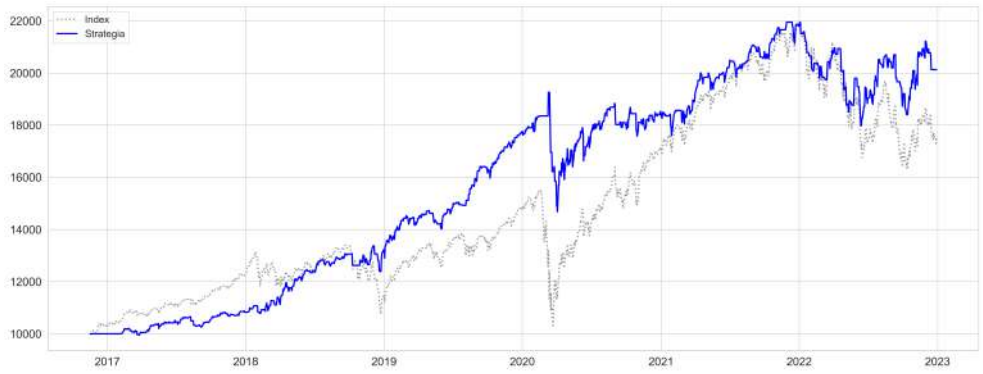


Figure 18: Investment in S&P500 and Strategy.

5 Conclusion

This study offered a side-by-side examination of various model types and their predictive capacities. Specifically, we delved into the forecasting prowess for one-day-ahead realized volatility using the GARCH model and its variations, as well as the LSTM model and its adaptations. Our data consisted of daily closing prices for the SP500, spanning from 01/04/1928 to 29/12/2022. Our primary goal was to concentrate on this singular index and assess the diverse predictive capabilities of the models for it.

Our empirical study divided the entire dataset into two segments: an in-sample section for estimating model parameters and an out-of-sample section for forecasting with the model. The in-sample segment comprises 20,282 daily observations, ranging from 04/01/1928 to 09/10/2008, whereas the out-of-sample portion contains 3,580 daily observations from 10/10/2008 to 29/12/2022. Broadly speaking, our findings indicate that Machine Learning models are more proficient than econometric models in capturing the dynamics of the index's realized volatility across both subsets. Notably, the most effective econometric model, GJR-GARCH (1,1), was surpassed by every Machine Learning model. The GJR-GARCH model had a validation MSE of 0.576, while the average validation MSE for the Machine Learning models was a mere 0.0026.

Such on outperformance may be explained by the greated ability of Artificial Neural Network in LSTM model to detect nonlinear patterns and relationships between input and output variables. In our work we also demonstrated the possibility to use prediction to construct a strategy capable to beat the S&P500 with higher performance (+14.38%.).

Future research may be conducted in order to improve these particular models by introducing new particular types of algorithms or to increase the frequency of the daily data available in order to obtain a better estimate of daily realised volatility.

Appendices

A Methodology

A.1 GARCH(p,q) as ARCH (∞)

The GARCH (p,q) regression is derived considering the following:

$$\epsilon_t = y_t - x_t' b$$

Moreover, you can rewrite the equation (13) as:

$$\sigma_t^2 - \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 = \omega + \sum_{i=1}^{p} \alpha_i r_{t-1}^2$$

$$(1 - B(L))\sigma_t^2 = \omega + A(L) r_t^2$$

so, if the roots of $(1 - B(L))$ exits, it's possible to write:

$$\begin{aligned}
\sigma_t^2 &= \omega(1 - B(L))^{-1} + (1 - B(L))^{-1} A(L) r_t^2 \\
\sigma_t^2 &= \omega(1 - B(L))^{-1} + D(L) r_t^2 \\
\sigma_t^2 &= \omega \left(1 - \sum_{j=1}^{q} \beta_j\right)^{-1} + \sum_{i=1}^{\infty} \delta_i r_{t-1}^2
\end{aligned}$$

since the term $D(\mathrm{L}) = (1 - B(L))^{-}1A(L)$, it's possible to write the relation in the following way:

$$\delta_i = \omega_i + \sum_{j=1}^{n} \beta_j \delta_{i-j}$$

The GARCH (p,q) process is wide - sense stationary with $\mathbb{E}[\epsilon_t] = 0$, $Var[\epsilon_t] = \omega(1 - A(1) - B(1))^{-}1$ and $cov(\epsilon_t, \epsilon_s)$ for $t \neq s \Leftrightarrow \mathrm{A(1)} + \mathrm{B(1)} < 1$

A.2 Back - propagation gradient derivation

Let $z_{m,i} = f(\alpha_{0m} + \alpha_m^T x_i)$ and let $z_i = (z_{1,i}, z_{2,i}, ..., z_{M,i})$. Then,

$$\mathbb{R}(\theta) = \sum_{i=1}^{N} R_i$$

and so,

$$\mathbb{R}(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{i,k} - f_k(x_i))^2,$$

with derivatives,

$$\frac{\partial R_i}{\partial \beta_{k,m}} = -2(y_{i,k} - f_k(x_i))g_k'(\beta_k^T z_i)z_{m,i} \ldots, M \tag{41}$$

$$\frac{\partial R_i}{\partial \alpha_{m,l}} = -\sum_{k=1}^{K} -2(y_{i,k} - f_k(x_i)) - g_k'(\beta_k^T z_i)\beta_{k,m} f'(\alpha_m^T x_i)x_{i,l} \tag{42}$$

Given the above derivatives, a gradient descent update at the (r+1)st iteration has form:

$$\beta_{k,m}^{(r+1)} = \beta_{k,m}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\delta R_i}{\delta \beta_{k,m}^{(r)}} \tag{43}$$

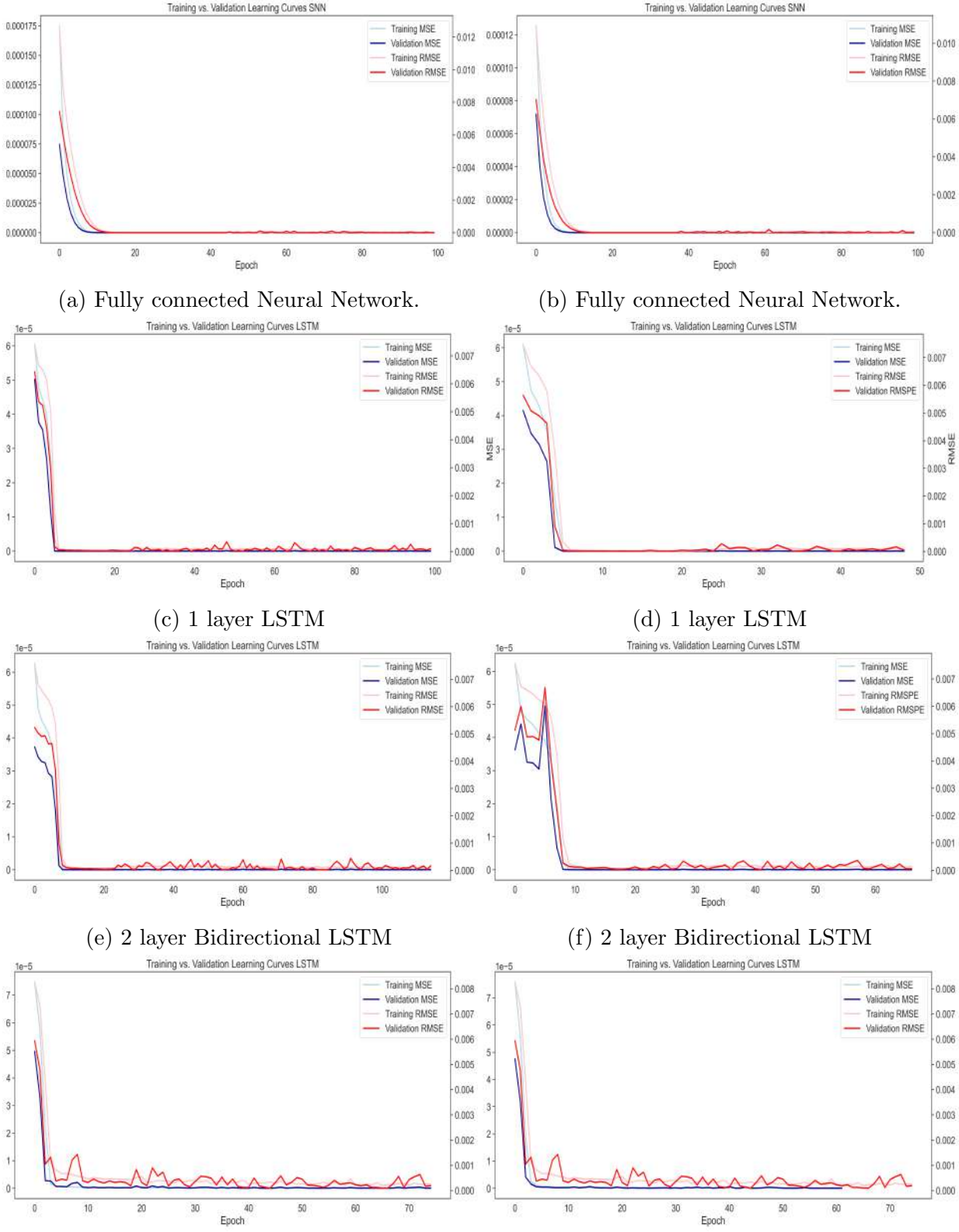
$$\alpha_{m,l}^{(r+1)} = \alpha_{m,l}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\delta R_i}{\delta \alpha_{m,l}^{(r)}}; \tag{44}$$

where γ is the *learning rate*. We can rewrite the relation ... as:

$$\frac{\partial R_i}{\partial \beta_{k,m}} = \delta_{k,i} z_{m,i} \ldots, \frac{\partial R_i}{\partial \alpha_{m,l}} = s_{m,i} x_{i,l}$$

The quantities $\delta_{k,i}$ and $s_{m,i}$ are the errors from the current model output and hidden layer units.

B Results Appendix: History and predicted value



(a) Fully connected Neural Network.

(b) Fully connected Neural Network.

(c) 1 layer LSTM

(d) 1 layer LSTM

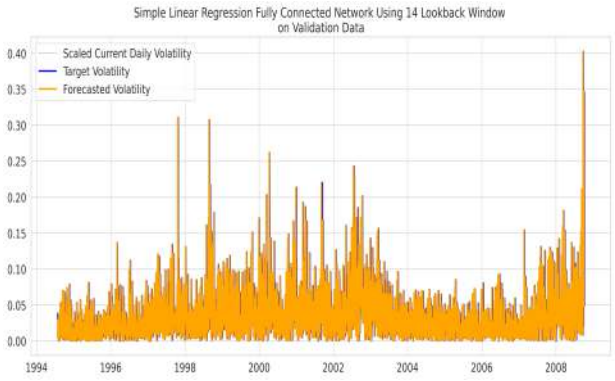
(e) 2 layer Bidirectional LSTM

(f) 2 layer Bidirectional LSTM

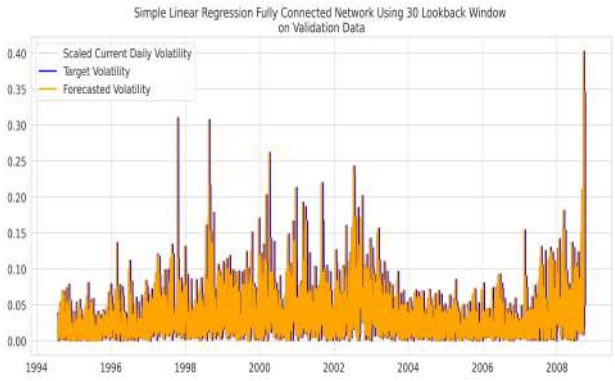
(g) 1 Convolutional 2 Bidirectional LSTM

(h) 1 Convolutional 2 Bidirectional LSTM

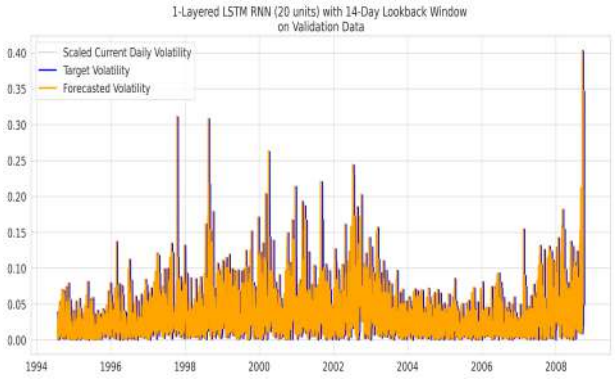
Figure 19: *In this figure is reported the value for the MSE and the RMSE for every Deep Learning model on training and validation set.*



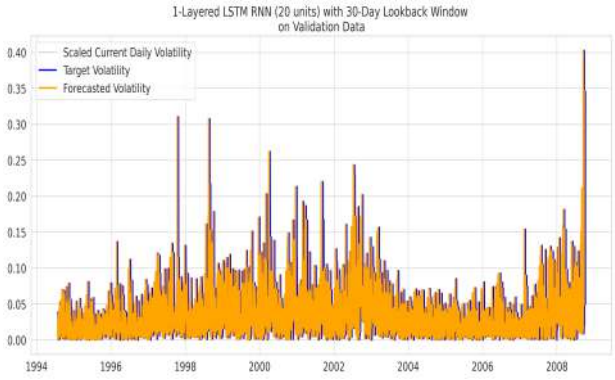
(a) Fully connected Neural Network, 14 window.



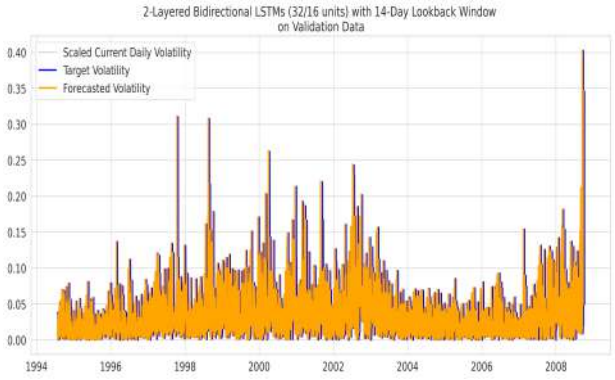
(b) Fully connected Neural Network, 30 window



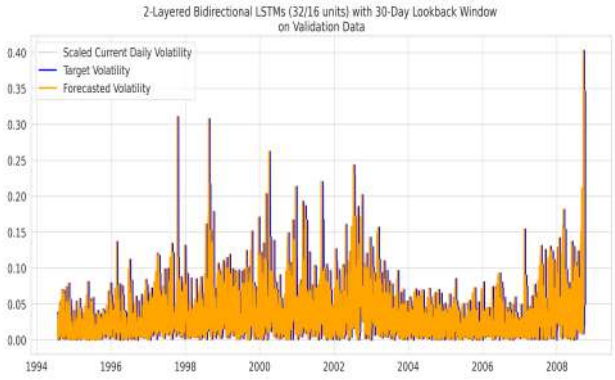
(c) 1 layer LSTM, 14 window



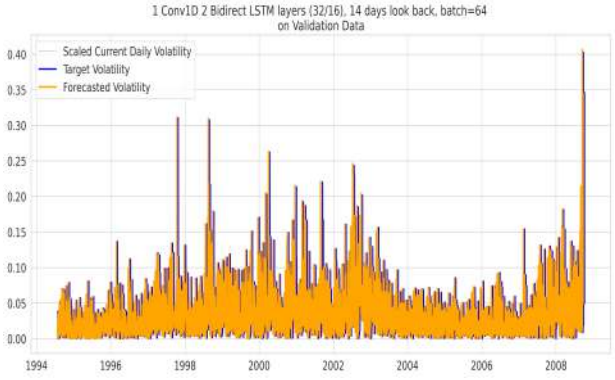
(d) 1 layer LSTM, 30 window



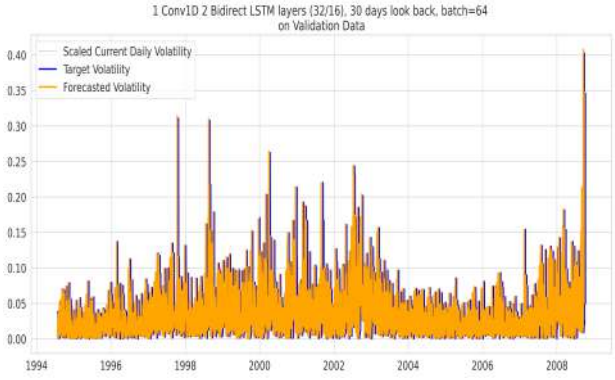
(e) 2 layers Bidirectional LSTM



(f) 2 layers Bidirectional LSTM



(g) 1 Convolutional 2 Bidirectional LSTM



(h) 1 Convolutional 2 Bidirectional LSTM

Figure 20: *In this figure is reported the fitted value for the for every Deep Learning model on the validation set.*

References

T. G. Andersen and T. Bollerslev. Heterogeneous information arrivals and return volatility dynamics: Uncovering the long-run in high frequency returns. *The Journal of Finance*, 52(3):975–1005, 1997. doi: https://doi.org/10.1111/j.1540-6261.1997.tb02722.x. pages 9, 13

T. G. Andersen and T. Bollerslev. Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International Economic Review*, 39(4):885–905, 1998. ISSN 00206598, 14682354. pages 9

T. G. Andersen, T. Bollerslev, F. X. Diebold, and P. Labys. Modeling and forecasting realized volatility. *Econometrica*, 71(2):579–625, 2003. doi: https://doi.org/10.1111/1468-0262.00418. pages 9

T. G. Andersen, T. Bollerslev, and N. Meddahi. Analytical evaluation of volatility forecasts. *International Economic Review*, 45(4):1079–1110, 2004. ISSN 00206598, 14682354. pages 9

T. G. Andersen, T. Bollerslev, and F. X. Diebold. Roughing It Up: Including Jump Components in the Measurement, Modeling, and Forecasting of Return Volatility. *The Review of Economics and Statistics*, 89(4):701–720, 11 2007. ISSN 0034-6535. doi: 10.1162/rest.89.4.701. pages 9

B. Aradi, G. Petneházi, and J. Gáll. Volatility forecasting with 1-dimensional cnns via transfer learning, 2020. pages 10

O. Assaf, G. Di Fatta, and G. Nicosia. Multivariate lstm for stock market volatility prediction. In G. Nicosia, V. Ojha, E. La Malfa, G. La Malfa, G. Jansen, P. M. Pardalos, G. Giuffrida, and R. Umeton, editors, *Machine Learning, Optimization, and Data Science*, pages 531–544, Cham, 2022. Springer International Publishing. ISBN 978-3-030-95470-3. pages 10

O. E. Barndorff-Nielsen and N. Shephard. Non-gaussian ornstein-uhlenbeck-based models and some of their uses in financial economics. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 63(2):167–241, 2001. ISSN 13697412, 14679868. pages 9

O. E. Barndorff-Nielsen and N. Shephard. 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, 2002. ISSN 13697412, 14679868. pages 14

T. Bollerslev. A conditionally heteroskedastic time series model for speculative prices and rates of return. *The review of economics and statistics*, pages 542–547, 1987. pages 22

F. Corsi. A Simple Approximate Long-Memory Model of Realized Volatility. *Journal of Financial Econometrics*, 7(2):174–196, 02 2009. ISSN 1479-8409. doi: 10.1093/jjfinec/nbp001. pages 9

R. Deo, C. Hurvich, and Y. Lu. Forecasting realized volatility using a long-memory stochastic volatility model: estimation, prediction and seasonal adjustment. *Journal of Econometrics*, 131(1):29–58, 2006. ISSN 0304-4076. pages 9

R. Engle and G. Gallo. A multiple indicators model for volatility using intra-daily data. *Journal of Econometrics*, 131(1-2):3–27, 2006. pages 9

E. F. Fama, L. Fisher, M. C. Jensen, and R. Roll. The adjustment of stock prices to new information. *International economic review*, 10(1):1–21, 1969. pages 16

C. Fjellström. Long short-term memory neural network for financial time series, 2022. pages 10

J. Fleming, C. Kirby, and B. Ostdiek. The economic value of volatility timing using "realized" volatility. *Journal of Financial Economics*, 67(3):473–509, 2003. pages 11

K. R. French, G. Schwert, and R. F. Stambaugh. Expected stock returns and volatility. *Journal of Financial Economics*, 19(1):3–29, 1987. ISSN 0304-405X. pages 9

F. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. 2:850–855 vol.2, 1999. doi: 10.1049/cp:19991218. pages 29

E. Ghysels, P. Santa-Clara, and R. Valkanov. Predicting volatility: getting the most out of return data sampled at different frequencies. *Journal of Econometrics*, 131(1):59–95, 2006. ISSN 0304-4076. doi: https://doi.org/10.1016/j.jeconom.2005.01.004. pages 9

L. R. Glosten, R. Jagannathan, and D. E. Runkle. On the relation between the expected value and the volatility of the nominal excess return on stocks. *The journal of finance*, 48(5):1779–1801, 1993. pages 23

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. pages 10, 29, 30

D. A. Hsieh. Chaos and nonlinear dynamics: Application to financial markets. *The Journal of Finance*, 46(5):1839–1877, 1991. ISSN 00221082, 15406261. pages 9

C. M. Jarque and A. K. Bera. A test for normality of observations and regression residuals. *International Statistical Review/Revue Internationale de Statistique*, pages 163–172, 1987. pages 19

W. Lu, J. Li, J. Wang, and L. Qin. A cnn-bilstm-am method for stock price prediction. *Neural Comput. Appl.*, 33(10):4741–4753, may 2021. ISSN 0941-0643. doi: 10.1007/s00521-020-05532-z. pages 10

B. Mandelbrot. When can price be arbitraged efficiently? a limit to the validity of the random walk and martingale models. *The Review of Economics and Statistics*, 53(3): 225–36, 1971. pages 17

F. J. Massey. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951. ISSN 01621459. pages 19

H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. pages 27

G. W. Schwert and P. J. Seguin. Heteroskedasticity in stock returns. *The Journal of Finance*, 45(4):1129–1155, 1990. ISSN 00221082, 15406261. pages 9

S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965. ISSN 00063444. pages 20

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks, 2014. pages 31

A. Vidal and W. Kristjanpoller. Gold volatility prediction using a cnn-lstm approach. *Expert Systems with Applications*, 157:113481, 2020. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113481. pages 10

H. Xiong, G. Yang, and Z. Wang. Factor portfolio and target volatility management: An analysis of portfolio performance in the u.s. and china. *International Review of Economics Finance*, 79:493–517, 2022. ISSN 1059-0560. pages 11

B. Zhou. High-frequency data and volatility in foreign-exchange rates. *Journal of Business Economic Statistics*, 14(1):45–52, 1996. ISSN 07350015. pages 9