UCLouvain

FACULTE DES SCIENCES

Faculté des sciences

Long Short-Term Memory neural network for econometric forecasting

A comparison between a statistical method and a neural network in the case of Value at Risk

Auteur·es : Antoine Geller

Promoteur·rices : Donatien Hainaut

Lecteur·rices : Christian Hafner

Année académique 2020-2021

Université Catholique de Louvain

Abstract

Antoine Geller

Financial forecasting may be difficult but is a very useful task in the field of market risk management. Statistical models have already proven their worth whereas the machine learning models are being researched and developed continuously. Unlike statistical models, machine learning models do not rely on assumptions and are able to detect non-linear patterns in the data. Furthermore, statistical models often suffer from the curse of dimensionality and need recurrent retraining to keep performing while the machine learning models can process more data, more features and for longer before lacking efficiency.

The objective of this master thesis is to provide a comprehensive comparison between a benchmark econometric model and a Long Short-Term Memory Neural Network on the computation of the Value at Risk of the Standard & Poor's Index (S&P500). To do so, we model the conditional mean and volatility of the financial series using these two multivariate models and proceed to Monte Carlo simulations to retrieve the necessary quantile.
For the benchmark econometric model, we use a Vector AutoRegressive Moving Average to estimate the conditional mean. To estimate the conditional volatility, we adopted the Dynamic Conditional Correlation model. Regarding the neural network, the architecture is built with two blocks. The first one is used for feature extraction purpose and the second one is used to forecast the parameters needed based on the latent codings and the exogenous variables. From the one-day ahead and 10-days ahead conditional estimations, we simulate 10.000 scenarios of returns and compute the Value at Risk and Expected Shortfall. We conclude this master thesis by comparing the performances on both risk measure exceedances and suggest ways to improve the computations.

Acknowledgements

First, I would like to thank M. Hainaut who accepted to supervise this master thesis and guided me through this task. Second, I would like to thank M. Biernaux for the wise recommendations on the structure and presentation of my work.
And finally, I would like to thank my family for their support as well as M. Dieu and M. Leroux for their sound advice on the spelling of this master thesis.

Contents

Abbreviations

ACF Autocorrelation Function. 26, 30

AE Autoencoder. 42

AE-Stacked LSTM Autoencoder- Stacked Long Short-Term Memory Neural Network. 45, 47, 51, 54, 55, 58, 60, 61

AIC Aikike Criterion. 27

ANN Artificial Neural Network. 36, 46

AR AutoRegressive. 25, 40

ARMA AutoRegressive Moving Average. 11, 22

BCBS Basel Committee on Banking Supervision. 3

BIC Bayesian Information Criterion. 27

CAPM Capital Asset Pricing Management. 3

CCC-GARCH Constant Conditional Correlation-Generalized Autoregressive Conditional Heteroskedasticity. 10, 31, 32

CDOs Collateralize Debt Options. 3

DCC-GARCH Dynamic Conditional Correlation-Generalized Autoregressive Conditional Heteroskedasticity. 10, 22, 30–32, 34, 54, 55, 59, 60

EGARCH Exponential GARCH. 10

ES Expected Shortfall. 7, 56, 57, 59–61

FIGARCH Fractional Integrated GARCH. 9

GAN Generative Adversarial Neural Networks. 61

GARCH Generalized Autoregressive Conditional Heteroskedasticity. 9, 11, 31

IGARCH Integrated GARCH. 9

LSTM Long Short-Term Memory. 40, 44, 48, 50, 61

MA Moving Average. 25, 40

MLP Multilayer Perceptron. 36, 37, 39

MSE Mean Squared Error. 49, 50, 53

NN Neural Network. 35, 36, 44, 46, 54, 58

PACF Partial Autocorrelation Function. 26, 30

RNN Recurrent Neural Network. 39

S&P500 Standard & Poor's Index. I, 2, 13, 14, 18, 19, 21, 22, 27, 33, 44, 45, 53, 55, 61

SSE Sum of Squared Errors. 28, 34, 51, 52, 54, 55

VaR Value at Risk. 4–9, 11, 12, 44, 56–62

VARMA Vector AutoRegressive Moving Average. 22, 24–27, 30, 32, 34, 59, 60

List of Figures

1 Introduction

"Risk management: one of the most important innovations of the 20th century."

Stated by A. Steinherr in his book *"Derivatives: The Wild Beast Of Finance"* (2000), the quote praises risk management in the financial institutions as the increase in volumes and values of derivatives threatens to crush the entire market structure without effective supervision [61].
Over the years, the financial institutions witnessed the increase of regulations and the unmissable impacts of the recurrent crisis. The last twenty years were home to multiple terrorist attacks, a global financial crisis, several crashes and a pandemic, to name but a few. *Extraordinary events are becoming the norm.* As a result, risk management gained a lot of popularity in the last decades and became a fundamental discipline in the financial institutions. With the aim of identifying and controlling the risks, the main challenge that comes with it is the ability to measure the exposure to the danger or the "hazard". Therefore, risk management is related to the monitoring of the uncertainty via probabilistic and stochastic models.

The risks present in any financial institutions can be divided into three main categories:

- **Market Risk**: The losses in financial investments due to changes in the underlying components.
- **Credit Risk**: The risk of a counterparty failing to meet its obligations.
- **Operational Risk**: The loss occurring from a poor internal process.

On a further note, we can add the liquidity and model risk which are, respectively, the difficulty to buy or sell an asset and the risk of using a unspecified or inappropriate model. To measure the exposure, risk managers take advantages of multiple statistical models, also known as financial econometrics. These models are commonly based on assumptions and constructed over linear dynamics, which are not always met in the real world.

Simultaneously, the twentieth century saw the exponential evolution of computational resources, data availability and affordability. This led to the expansion of data-based models. Also known as Machine Learning models, it is *"the science of building statistical models from data with few or no assumptions about the underlying process"*[1]. The data-based models have gained momentum as they opened new ways of solving problems in a wide range of applications. With the exciting developments in the field of computer science and statistics, the neural networks have resurfaced around 2010 after being sidelined for over a decade [26].

[1]Michel Denuit, Donatien Hainaut and Julien Trufin, *Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions*, Springer Actuarial, 2019

The application of Machine Learning in the field of risk management is only a logical continuation of the development of models to measure the risks more precisely. In point of fact, it is in the best interest of financial institutions to know how to determine at best risk factors it faces every day to be able to react. Point being, the objective of this master thesis is to compute with two different models a well-known market risk measure, the Value at Risk.

In the first half, we define the contribution to the literature. Based on multiple articles, we state the limits and the objectives of this master thesis. Thereafter, we define the empirical properties of the financial series of returns where we focus on the *Standard & Poor's Index* (S&P500). We move on to the feature selection for our models where we test the relevance of different variables to introduce in the later models. We finish the first half of our master's thesis with one of the most important parts being the benchmark econometric model. Starting with a theoretical background and stating the hypothesis, we build a multivariate Autoregressive Moving Average combined with a Dynamic Conditional Correlation model to estimate the mean and volatility of the S&P500 returns.
In the second half, we illustrate the theoretical concepts about the neural networks that we use in the following. Then, we compile a two-block Long Short-Term Memory neural network model to define the mean and the volatility of the S&P500 returns. The two specific blocks are defined for feature extraction and for forecasting purpose. Next, we simulate multiple scenarios of the S&P500 returns and retrieve the necessary quantile. We conclude this master thesis by comparing the performances on the Value at Risk exceedances and suggest ways to improve the computations.

2 Background of Risk Management

Prior to the 1970s, the field of risk management in the financial sector was lacking efficiency. On one hand, the institutions were flawed on tools to model the credit risks applicable to them and focused only on the operational risk. And on the other hand, the bank regulators failed to make any defensible interventions in the sector [38]. Nevertheless, the increase in influence of large banks in the economy as well as the rise of interdependence between financial markets [35] pressured the importance of regulations in the field.

In 1974, a small German bank, called Herstatt was forced by the German authorities to be liquidated. The incident, although small, had major repercussions in the United States as the German bank found itself unable to repay its debts. This episode is followed by a crisis that revealed the importance of systemic risk in the markets [50]. The incident led to the creation of the Basel Committee on Banking Supervision (BCBS). The BCBS was founded with the aim to improve the banking supervision by imposing regulations.
In parallel to the BCBS, the rise of derivatives and financial innovations saw the advent of two major innovations: Fischer Black and Myron Scholes who introduces the option pricing model [12] and William Sharpe who presented the Capital Asset Pricing Management (CAPM) [60]. These two novelties lead to new ways to model the risk in the financial markets.

In 1988 is issued the *First Basel Accord* which is the first outcome of the BCBS. The main objective is to improve the banking stability by imposing a minimum capital requirement to minimise the credit risks [8].
But from 1988 to the years 2000, the notion of credit risk was essentially reinvented with the apparition of Equity Index Swaps, catastrophe options, Collateralize Debt Options (CDOs), macroeconomic derivatives, etc. pointing to more and more complexified markets [38]. Therefore, a remodelling of the regulations becomes necessary [38]. The *Second Basel Accord* initiated in 2001 and published in 2004, is based on three main pillar-concepts. The first and the most important is a change in the computation of the minimum capital charge to be more coherent with the banks' activities [8]. *Basel II* takes into account the capital charge for credit risk, market risk, and operational risk. The second pillar introduces a monitoring review process. It increases the importance of supervision and overview of the various risks. The third pillar seeks to establish market discipline through various public-disclosures requirements with the customer, rating agencies, analysts and others [36]. A second version of *Basel II* addresses the build-up of risk in the trading book [8].
Despite the efforts of the Basel Committee to provide regulations, the fast-growing and multi-trillion dollars market of CDOs inevitably paved the way to the 2008 crisis. The United States saw the bankruptcy of the bank Lehman Brothers before bailing out the other institutions that could potentially crash the economy of the world [17]. In Belgium, this crisis led to severe problems for the two major banks, Dexia and Fortis [40].

In response to the 2008 financial crisis, the *Third Basel Accord* was issued in 2011. It is built on three principles extensions to *Basel II* [8]:

1. Increase of the minimum regulatory capital from 2% (*Basel II*) to 4.5% and allowing countercyclical adjustments to these ratios in periods of crisis.
2. Use of leverage ratio to prevent excessive debt.
3. Introduction of two main liquidity ratios to force the banks to hold highly liquid assets and maintain stable funding.

This time being, the *Basel IV* has been discussed and will mainly complete the *Basel III* accords. *Basel IV* introduces modifications in the computation of capital requirement based on internal models. This involves a standardised floor, a reduction in standardised risk weights for low risk mortgage loans, a rise in the leverage ratio for Global Systemically Important Banks, and more detailed public disclosure including financial statistics [52]. These reforms should take place in January 2023.

3 Market Risk Measures

With the increase of risks and regulations in place, especially with the Basel Accords, the challenges behind market risk management are gaining a lot of reputations. One of those challenges is the ability to evaluate the unexpected or extreme outcomes rather than the expected or average outcomes. This particular challenge has been studied with the introduction of the so-called Value-at-Risk popularised by JP Morgan in the nineties in the framework of capital-adequacy in *Basel I*.

3.1 Value at Risk

The *Value at Risk (VaR)* is a measure of risk of loss for a financial position. The distribution function of the loss of the asset, also called loss distribution function, is denoted by $F_L(l) = P(L \leq l)$ where $F_L(l)$ is the probability of bearing a loss l being higher or equal to a loss value of L. Given a confidence level of $\alpha \in (0, 1)$, the aim of the VaR is to define the smallest amount of loss, l, such that the probability that the actual loss L exceeds l is not higher than 1-α. The mathematical notations in this section come from the book *"A. McNeil, R. Frey and P. Embrechts,Quantitative Risk Management,Princeton University Press, 2005, pg 38-45"*.

$$VaR_\alpha = \inf\{l \in \mathbb{R} : P(L > l) \leq 1 - \alpha\} = \inf\{l \in \mathbb{R} : F_L(l) \geq \alpha\}$$

In other words, the probability of having a loss higher than the Value at Risk is equal to $1 - \alpha$.

$$P(L \leq VaR_\alpha) = P\left(\frac{L - \mu}{\sigma} \leq \Phi^{-1}(\alpha)\right) = \Phi(\Phi^{-1}(\alpha)) = \alpha$$

Intuitively, the VaR is a quantile of the returns' distribution. It is worth noting that the returns' distribution is another term for the loss distribution, as both represent the loss, either positive or negative. The standard confidence level in quantitative risk management is usually 95% and 99% with a time horizon of 1 and 10 days. For the purpose of our analysis, we distinguish between conditional and unconditional VaR. We focus our work on the conditional VaR which implies that the value of l is time dependent. We note the conditional VaR as $\text{VaR}_{t,\alpha}$ for time t and confidence level α.

There are three major ways to compute the VaR: *The Historical Method*, *The Parametric Method* and *The Monte Carlo Simulations*.

3.1.1 Historical Method

The historical method, as the name suggests, uses the historical returns of the financial time series. This method can be thought as estimating the VaR using the empirical quantile estimation. For this matter, the theoretical quantiles of the loss distribution are estimated by sample quantiles of the data.

We consider a time series of ordered returns $\mathbf{X} = x_1, ..., x_t$. We can estimate the VaR by retrieving the sample of returns over $\mathbf{X}_\Delta$, where Δ is a time horizon defining the sample, usually one year of trading days. From the sample of returns, we can easily capture the α-th quantile of the sample needed to compute the VaR_α at time $t + 1$.

The historical method has multiple advantages, such as the ease of implementation and the lack of assumption on the distribution of returns. However, the main drawback is that the dynamics between assets in the markets may have changed and hence, the historical returns do not represent adequately the present dynamics between assets. Another disadvantage would be the possible absence of relevant data in the history to represent the risks faced today.

3.1.2 Parametric Method

The second method is a parametric approach with the required assumption being the Normal distribution of the returns, $X_t \sim \mathcal{N}(\mu, \sigma^2)$. In such manner, we need to estimate the mean and the standard deviation of the asset to be able to retrieve the desired quantile of the distribution. In a conditional point of view, the mean and standard deviation of the asset have to be re-estimated each day.

$$VaR_{t,\alpha} = \mu_t + \sigma_t \Phi^{-1}(\alpha), \quad t \in \mathbb{Z}$$

where $\Phi^{-1}(\alpha)$ denotes the α-quantile of the standard normal distribution function and, μ_t and σ_t, the estimation of the mean and standard deviation at time t.

Elseways if we need the VaR of a portfolio composed of multiple assets, with the assumptions $\mathbf{X}_t \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, we must compute the mean of each asset composing the portfolio as well as their matrix of variance-covariance. Thereafter, we can weigh the matrices by each asset

$$\sigma^2_{t,ptf} = \boldsymbol{\omega}'\Sigma_t\boldsymbol{\omega}, \quad t \in \mathbb{Z}$$
$$VaR_{t,\alpha} = \omega'\boldsymbol{\mu_t} + \sigma_{t,ptf}\Phi^{-1}(\alpha), \quad t \in \mathbb{Z}$$

where Σ_t is the matrix of variance-covariance of all the assets, at time t, and ω is the vector of weights of all the assets.

3.1.3 Monte Carlo Simulations

The third method is based on Monte Carlo simulations which is a commonly used term to denote simulations of scenarios. They are used to estimate results that are not easily determined due to the intervention of random variables. For this method, we need a stochastic model to estimate the parameters needed. The choice of the model is up to the risk manager. The main ideas of this method are crossing the historical and the parametric method principles.

Suppose a series of ordered returns, $\mathbf{X} = x_1, ..., x_t$, with a conditional mean μ_t and conditional variance σ_t. We model these first and second moments of the series until t with an arbitrary stochastic model and estimate the value at time $t + 1$. Based on these parameters, we can simulate n independent scenarios of returns with the intervention of a random variable changing for each simulation. Thereafter, such as in the historical method, we can retrieve the α-quantile in our simulations, at each time t, needed to compute the $VaR_{t,\alpha}$. This method is very dependant upon the underlying model chosen to estimate the mean and the volatility. It is worth noting that the distribution chosen for the random variable can be adapted to a Normal distribution, a Student distribution or other to be more consistent with reality.

Although very intuitive and popular, the Value at Risk is not the best measure of loss. In fact, it was very criticised by Artzner et al. with the introduction of the term *coherent risk measure* in their paper *Coherent Measures of Risk* [2].
With $\mathcal{L}$ being a linear space, a risk measure represented by a function $\rho : \mathcal{L} \to \mathbb{R}$ is coherent if it satisfies these four main properties:

- **Translation Equivariance**: The risk of $\rho(L + l)$ has to be less than the risk of $\rho(L)$.
$$\forall L \in \mathcal{L}, l \in \mathbb{R}, \rho(L + l) = \rho(L) + l$$
- **Subadditivity**: This is comparable to the diversification principle where the risk of a single stock is higher than a combination of multiple stocks in a portfolio.
$$\forall L_1, L_2 \in \mathcal{L}, \rho(L_1 + L_2) \leq \rho(L_1) + \rho(L_2)$$
- **Positive Homogeneity**: The risk of a financial position is proportional to its size.
$$\forall L \in \mathcal{L}, \lambda \geq 0, \rho(\lambda L) = \lambda\rho(L)$$

- **Monotonicity**: Positions that leads to a higher loss require higher capital.

$$\forall L_1, L_2 \in \mathcal{L}, L_1 \leq L_2, \rho(L_1) \leq \rho(L_2)$$

Artzner et al. denounced the Value at Risk for its non-subadditivity characteristic. Therefore, the VaR is not a coherent risk measure. The subadditivity encourages the risk manager to diversify his positions while the VaR tends to discourage it. An intuitive example of this affirmation about the VaR is presented in the Appendix, section A. This is the reason why we introduce another measure of risk, known as the Expected Shortfall.

3.2 Expected Shortfall

The *Expected Shortfall (ES)* is a second measure of the market risk with the particularity that it is coherent. Recall the loss distribution function denoted by $F_L(l) = P(L \leq l)$.

For a loss L with $E(L) < \infty$, the expected shortfall at a confidence level $\alpha \in (0, 1)$ is represented as,

$$ES_\alpha = \frac{1}{1-\alpha} \int_\alpha^1 q_u(F_L)\, du$$

where $q_u(F_L)$ is the quantile function of F_L. Hence, as $q_u(F_L)$ is equal to the $VaR_u(L)$, the ES is related to the VaR by,

$$ES_\alpha = \frac{1}{1-\alpha} \int_\alpha^1 VaR_u(L)\, du$$

Intuitively, the ES is an average of the VaR over $u \geq \alpha$, hence we are taking a measure further into the tail of the loss distribution. We can rewrite the Expected Shortfall as a more intuitive equation,

$$ES_\alpha = \frac{E(L; L \geq q_\alpha(L))}{1-\alpha} = E(L|L \geq VaR_\alpha)$$

As the book *Quantitative Risk Management*, from A. McNeil, R. Frey and P. Embrechts, p. 44, explains it well, "an even more intuitive expression can be derived which shows that the expected shortfall can be interpreted as the expected loss that is incurred in the event that VaR is exceeded." [2]. This implies an obvious characteristic of the ES which is $ES_\alpha \geq VaR_\alpha$.

Out of the three main methodologies presented for the VaR, we focus on the Monte Carlo Simulations. We use two different underlying multivariate models to model the conditional mean and the conditional volatility of the S&P500 at time t. Then, we forecast the two values over two time horizons, being one-day ahead and 10-days ahead. Next, we run a consequent number of simulations using our forecasts, adding a random variable drawn from a Normal distribution with mean 0 and variance 1. Finally, we capture the 95-quantile needed at each time t to compute the VaR, and the ES from the VaR value.

[2] A. McNeil, R. Frey and P. Embrechts, Quantitative Risk Management, Princeton University Press

4 State of the Art Algorithms for Value at Risk

Research on methods to compute the Value at Risk goes back to the 1920s [32] with the origins of portfolio risk management. Starting with non-mathematical constructions of portfolios with authors such as Hardy (1923) and Hicks (1935) where they discussed the benefits of diversification. The first quantitative explanation traces back to Leavens in his paper *"Diversification of Investments."* published in 1945 where he also discusses how diversification can reduce the risk of a portfolio [42]. Leavens did not explicitly introduce the concept of VaR metric, but he mentioned on many occasions the "spread between probable losses and gains." This can be considered as the first VaR measure published [32].

The year 1952 saw the separated publications of Markowitz (1952) and Roy (1952). They both published papers about the VaR measures that were astonishingly close. Each was studying methods to select portfolios that would optimise the benefit for a specified level of risk [46] [59]. For this purpose, they integrated covariances between the assets so to reflect hedging and diversification. While the two measures were similar, they differ in the way they compute the VaR.

With the years 1970s came the evolution of the computational resources and therefore the possibility to process simulations of time series to create the VaR. In 1971, Bernard Lietaer developed a model that focuses on the risk of exchange rate fluctuations [43]. The topic has been high on the agenda because, after the Second World War, most currencies began to devalue [32]. His model was an evolution to the computation of the VaR with the variance of market value. He assumed that the magnitude of devaluations was normally distributed. Lietaer views this problem as one of generating mean-variance efficient portfolios of foreign currency transactions, from which management may select a portfolio with desirable risk-return characteristic. Lietaer's work was probably the first time, the Monte Carlo method was used in the aim of reducing risks on financial times series and computing the VaR [32].

As explained in the previous section, there are different methods to compute the Value at Risk: the historical method, the parametric method and the Monte Carlo simulations. Since our method of choice is the Monte Carlo simulations with stochastic models, we present a non-exhaustive list of the most popular stochastic models used in the literature to compute the Value-at-risk. Part of this list is based on the paper *Abad, P., Benito, S. López, C., "A comprehensive review of Value at Risk methodologies", The Spanish Review of Financial Economics, 2014.*

4.1 GARCH family

The most popular models are part of the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) family. The GARCH process was introduced in 1986 by T. Bollerslev [13]. Starting with the definition of returns of a financial time series,

$$X_t = \mu_t + \epsilon_t, \quad \epsilon_t = \sigma_t Z_t, \quad t \in \mathbb{Z} \qquad [3]$$

$$with \quad \epsilon_t | \mathcal{F}_{t-1} \sim \mathcal{N}(0, \sigma_t^2)$$
$$alternatively \quad Z_t | \mathcal{F}_{t-1} \sim \mathcal{N}(0, 1)$$

Note that for information purposes only, I state the definition of returns above. An explanation of the details comes later in the work.

A GARCH model is be represented as,

$$\sigma_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 + \beta_1 \sigma_{t-1}^2 + \cdots + \beta_p \sigma_{t-p}^2 =$$

$$\omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 \qquad [4]$$

where σ_t^2 is the variance of the risk factor. The statistical model is very useful to model the clusters of volatility in a financial position. C. Alexander and C. Leigh published a paper in 1997 where they examine the performance of three volatility forecasting models to compute the internal value at risk. The three models being the exponentially weighted average, the equally weighted average of squared returns and the Generalized autoregressive conditional heteroskedasticity [18].

From the original GARCH model, many derivations have seen the day in order to improve the results. The Integrated GARCH (IGARCH) of Engle and Bollerslev (1986) is conceived to model to the persistence of conditional variance. The idea is to impose a condition on the expression of the model [23]. From the empirical point of view, the properties of the IGARCH are not highly interesting due to the very slow alleviation of the shock impact upon the conditional variance. Nevertheless, the impacts that fade away show exponential behaviour, which is how the Fractional Integrated GARCH (FIGARCH) proposed by Baillie et al. (1996) behaves [4]. Several papers studying the advantages of the FIGARCH in the framework of the VaR are available [64] [7].
Many proposed GARCH models reflect well on the characteristics of volatility clustering. Yet, improvements were brought to overcome a major defect. In fact, the preceding models do not take into account the asymmetric performance of yields between positive and negative shocks, also called leverage effect. In the previous models, the effect of the positive and the negative innovations are equivalent. However, in practice, we observe a higher volatility rate when the returns are negative compared to positive. As a means to capture

[3] Equation from the book "A. McNeil, R. Frey and P. Embrechts, *Quantitative Risk Management*, Princeton University Press, 2005"

[4] Equation from the book "D. Ruppert, *Statistics and Data Analysis for Financial Engineering*, Springer, 2015"

the leverage effect, several non-linear GARCH formulations have been proposed, with one of them being the Exponential GARCH model, Exponential GARCH (EGARCH) [51]. Sener et al. published the paper *"Ranking the Predictive Performances of Value-at-Risk Estimation Methods"* in 2012 where they compared multiple GARCH models to compute the Value at Risk. They conclude a higher performance for the asymmetric GARCH models such as EGARCH. Other GARCH models have been studied to determine the Value at Risk, with two examples being the Markov-Switching GARCH models [58] and the Treshold-GARCH model [63]. A non-comprehensive list of some GARCH-type models are present in Figure 27, Appendix B.

It is worth noting that there also exist multivariate versions of the GARCH model. The most popular being the Constant Conditional Correlation-Generalized Autoregressive Conditional Heteroskedasticity (CCC-GARCH) proposed by T. Bollerslev in 1990 [14] and Dynamic Conditional Correlation-Generalized Autoregressive Conditional Heteroskedasticity (DCC-GARCH) proposed by Robert F. Engle [24]. The paper *"Value-at-Risk with Application of DCC-GARCH Model"* of Tomas Meluzin et al. treats the computation of the Value at Risk with the help of a DCC-GARCH model.

4.2 Non parametric Method: Nadaraya-Watson

Another method to define the underlying dynamics of the returns is a non parametric method called Nadaraya-Watson used in the work of Mattias Bengtsson and Viktor Olsbo in *Value at Risk Using Stochastic Volatility Models* (2003) [54]. They define the volatility by the squared of de-meaned returns, X_t^2, and denote

$$X_t^2 = \sigma_t^2 + \epsilon_t, \quad with \quad E[\epsilon] = 0$$

with σ_t^2 the volatility to estimate, and ϵ_t the error term with a mean of 0. Since it is a non parametric method, they aim to estimate σ^2 without any distribution assumption on the returns but we rather use a sum of weights on the square returns;

$$\sigma_t^2 = \sum_{i=0}^{n} w_i(t,h) X_i^2$$

$$with \quad w_i(t,h) = \frac{K(t-t_i)/h)}{\sum_{j=1}^{n} K(t-t_i)/h)}$$

where K(.) is an appropriate kernel chosen beforehand and h is a defined bandwidth. The quality of the estimator depends highly on the bandwidth chosen by the statistician.

M. Bengtsson and V. Olsbo used the following estimator,

$$\sigma_t^2 = \frac{K((t-t_i)/h)X_i^2}{\sum_{j=1}^{n} K((t-t_i)/h)}$$

with $X_t = R_t - \mu_t$, $\mu_t = \frac{1}{t}\sum_{t=1}^{T} R_t$ and $K_h(.) = h^{-1}K(./h)$ where $K(x) = e^{-x^2}$. R_t represents the returns at time t. The characters h and n are respectively known as the bandwidth and the window length. Again, with the estimation of the volatility, it is possible the compute the VaR with the use of Monte Carlo simulations.

4.3 Stochastic Volatility Model

The stochastic volatility model has been proposed by Taylor in 1982 in his paper *"Financial Returns Modelled by the Product of Two Stochastic Processes"* [62]. The Stochastic Volatility model is defined as follows,

$$X_t = \mu_t + \sqrt{h_t}Z_t, \quad with \quad Z_t \sim \mathcal{N}(0,1)$$
$$log(h_{t+1}) = \alpha + \phi log(h_t) + \eta_t, \quad with \quad \eta_t \sim \mathcal{N}(0,\sigma_n)$$

where μ_t represents the conditional mean of the risk factor's returns. h_t represents the conditional variance, and both Z_t and η_t are white-noise processes.

Such as for the GARCH-type family, other models have been developed to take into account the large memory over time with the paper *"The detection and estimation of long memory in stochastic volatility"* of Breidt et al. (1998), and the leverage effect with the work of Harvey and Shephard (1996) and So et al. (2002). Multiple stochastic volatility models used for the VaR measure were studied by González-Rivera et al. (2004) where they came to the conclusion that modelling the conditional standard deviation instead of the variance seems to be a dominant model [5]. Fleming and Kirby (2003), as well as others, also studied the relevance of the stochastic volatility models in the context of the VaR [37].

4.4 Neural Networks

The field of machine learning, and especially deep learning, also welcomed the idea to model the returns to retrieve the so-called Value at Risk. While still in development, multiple articles were published attesting the computation of the VaR using neural networks.

Xiaoliang Chen, Kin Keung Lai and Jerome Yen (2009) published a paper where they compared multiple methods to compute the VaR. They used the GARCH model as well as an artificial neural network [67]. In this study, they compared their artificial neural network to the benchmark AutoRegressive Moving Average (ARMA) model combined with a GARCH model where they came to the conclusion that the use of an ANN proved to be difficult for a number of reasons. However, they suggest introducing exogenous variables in the model to improve it.
In the paper *"Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models"*, the authors used a Long Short-Term Memory

neural network with, as parameters, the GARCH coefficients [39]. Later, Zhichao Du, Menghan Wang & Zhewen Xu (2019) proposed a recurrent neural network model based on the previously cited paper to compute the VaR. The method differs from Kim and Won's paper by using only the data from the S&P500 and no GARCH parameters. In fact, they used a univariate Long Short-Term Memory to compute the VaR and then compared it to the GARCH model as well as the historical simulations [68]. Their conclusion is very optimistic as they obtain similar if not better results than the usual models. They finished their work by proposing to train a deeper model to improve the results in further studies.

4.5 Contribution to the literature

Following the review of the literature, the contribution of this work is to develop a Long Short-Term Memory neural network model to compute the conditional Value at Risk and Expected Shortfall. As advised by Xiaoliang Chen, Kin Keung Lai and Jerome Yen, we introduce the use of exogenous variable in our model, making it an evolution of their work. Our model is therefore multivariate, using only the returns data of different times series.
Moreover, to react on the proposition of Zhichao Du, Menghan Wang & Zhewen Xu to train a deeper model, we introduce a novelty in the neural network. Instead of increasing only the depth, we introduce a feature extraction block to capture best the dynamics of the risk factors. The results will be displayed for the one-day ahead Value at Risk and for the 10-days ahead Value at Risk. Based on the paper of Zhichao Du, Menghan Wang, Zhewen Xu (2019), we also compare our results to a benchmark model. In our case, it is the multivariate version of the ARMA - GARCH, known as Vector ARMA - DCC-GARCH.

The aim of this master thesis is to model the conditional mean and volatility of a risk factor with two different models to be able to simulate multiple scenarios via Monte Carlo. The work intents to give a comprehensive comparison between both models and their performance on the measure of Value at Risk and Expected Shortfall.

5 Empirical Properties of financial data

Our analysis mainly focuses on the Standard & Poor's Index. Known as the S&P500, it is a stock market index that groups the stock performances of the 500 largest companies traded on the stock exchange in the United States. This index was first developed in 1926 with a total of 90 stocks. In March 4th 1957, it was expended to 500 companies as it remains today.

The S&P500 is a capitalisation-weighted-index which means that the companies are weighted according to their market values. Companies have entered and left the index over the years. As of April 2021, the 10 largest companies by market capitalisation present in the index are : *Apple Inc.*, *Microsoft*, *Amazon.com*, *Facebook*, *Alphabet Inc. Class A*, *Alphabet Inc. Class C*, *Tesla*, *Berkshire Hathaway*, *JPMorgan Chase & Co.* and *Johnson&Johnson*.

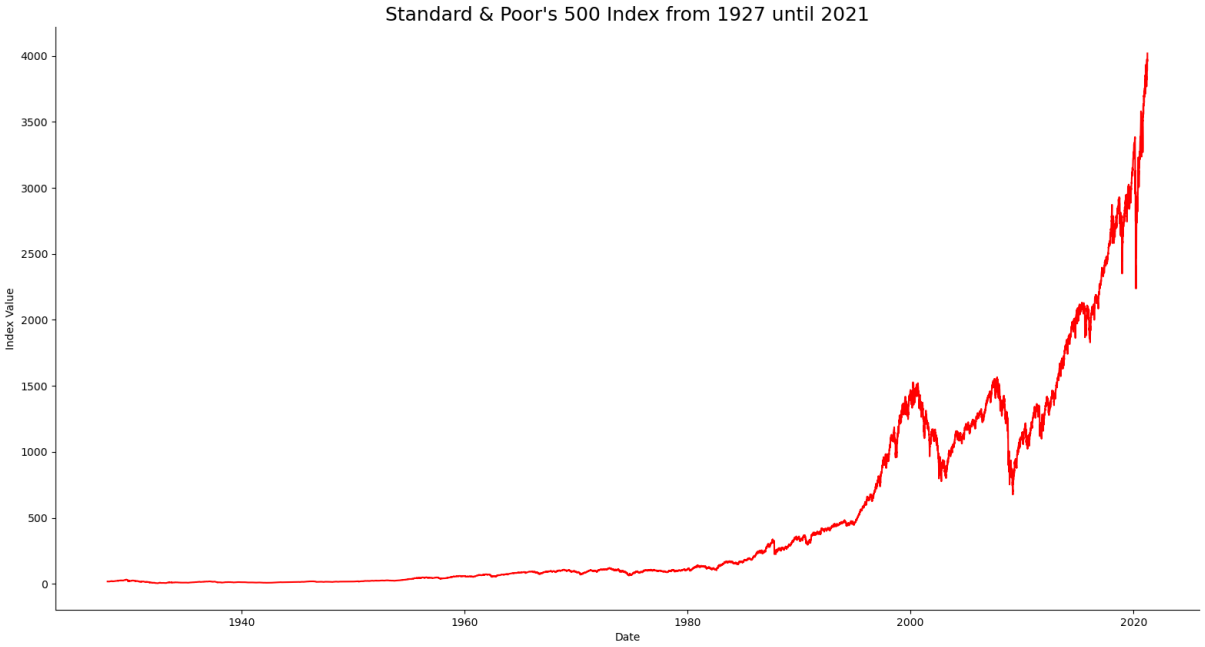

Figure 1: Evolution of the Standard & Poor's Index from 1927 until 2021

As of April 2021, the S&P500 is at an all-time high with a value above 4.000 points. It went through multiple crisis periods, represented by the downward trends such as the 2008 crisis or more recently the "Covid-19 pandemic" crisis.

Many investors and economists consider the S&P500 as an indicator of the overall performance of the US equity market. This is due to the large number of companies taken into account as well as the diversified range of sectors taken into consideration, from information technology and Consumer Services to Financial and Healthcare sectors.

5.1 Stylized Facts about Returns

We consider the S&P500 as a financial time series $\mathbf{X} = x_1,x_2,...,x_n$, represented by the closing price each day. We can compute the returns over a period of time from $t-1$ until t in two different approach, called respectively the simple returns and the log-returns;

Simple returns | *Log-returns*

$$r_t = R_t + 1 = \frac{x_t}{x_{t-1}} \qquad r_t = log(R_t + 1) = log(\frac{x_t}{x_{t-1}})$$

where x_t is the price at time t, x_{t-1} is the price at $t-1$ and r_t is the value of the return at time t. Note that the rate of return is defined as R_t where $r_t = R_t + 1$.

For the remainder of our analysis, we use the log-returns. They are very popular in the financial academic world as their main advantage is the *time-additivity*. In fact, the log return over a k periods is

$$\begin{aligned} r_t(k) &= log[R_t(k) + 1] \\ &= log \prod_{j=0}^{k-1} (R_{t-j} + 1) \\ &= \sum_{j=0}^{k-1} r_{t-j} \end{aligned}$$

which makes it very convenient but also efficient in terms of computational resources.

5.1.1 Volatility Clustering

The first stylized fact about the returns displayed next page, Figure 2, is the distinctive volatility clusters, noted first by Mandelbrot (1963) [45]. The volatility is a measure of the dispersion of the returns around their mean. Volatility is also described as a measure of risk where, the more volatile, the riskier the security is. It can be grouped in time-stamps. The periods of low volatility are followed by periods of low volatility and periods of high volatility are followed by periods of high volatility. The volatility is significantly higher in periods of crisis while in calm periods, it is lower.
There are multiple approaches to measuring the volatility of a risk factor. For the sake of our work, the volatility for a period of time Δ is equal to the standard deviation of the returns for this period horizon. We will go into more details about this computation in the further sections. It is worth noting the major downside to this method. Computing the volatility as a standard deviation over a period of time is like to "chasing a running train". This means that the measure of volatility at time t is always bias compared to the real value because of the multiple days taken into account in our calculation. Therefore, sudden high peaks of volatility are always underestimated in the computation.

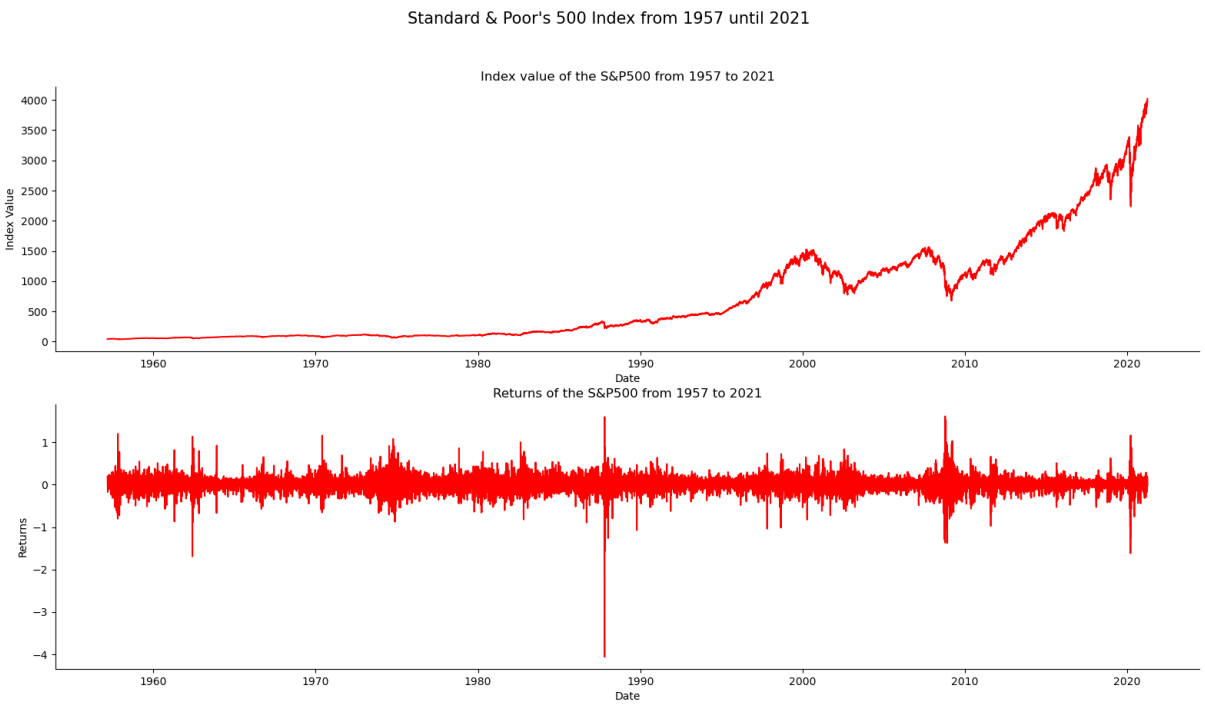

Figure 2: Evolution of the closed price and the returns of the S&P500 from 1957 until 2021

Periods of high volatility clusters are easily observed around the time of crisis. Periods of high market volatility are followed by periods of high volatility and equivalent for the periods of low volatility.

5.1.2 Non-Normality of the Returns

To study the second stylized fact about the returns, we display the histogram along the *Normal* density curve, Figure 3 on the next page. We based our observation from 1957 until 2021 to withdraw facts about the mean, variance and the overall characteristics of the distribution. As shown below, the histogram of the returns does not match the *Normal* distribution with the respective mean and variance of the returns. We state the second stylized fact as the *Non Normality* of the returns. The returns distribution has a higher density around the mean while also having fat tails. This was first noticed by Mandelbrot in his book *"The Variation of Certain Speculative Prices", 1963* [45] and Fama in *"The Behavior of Stock Market Prices",1965* [25]. This observation is confirmed by the QQ plot, Figure 4 next page, which reveals larger tails than the *Normal* distribution. We can distinguish extreme values on the QQ plot around the coordinates {-4, -4} and on the opposite of the graphic, around {2, 4}.

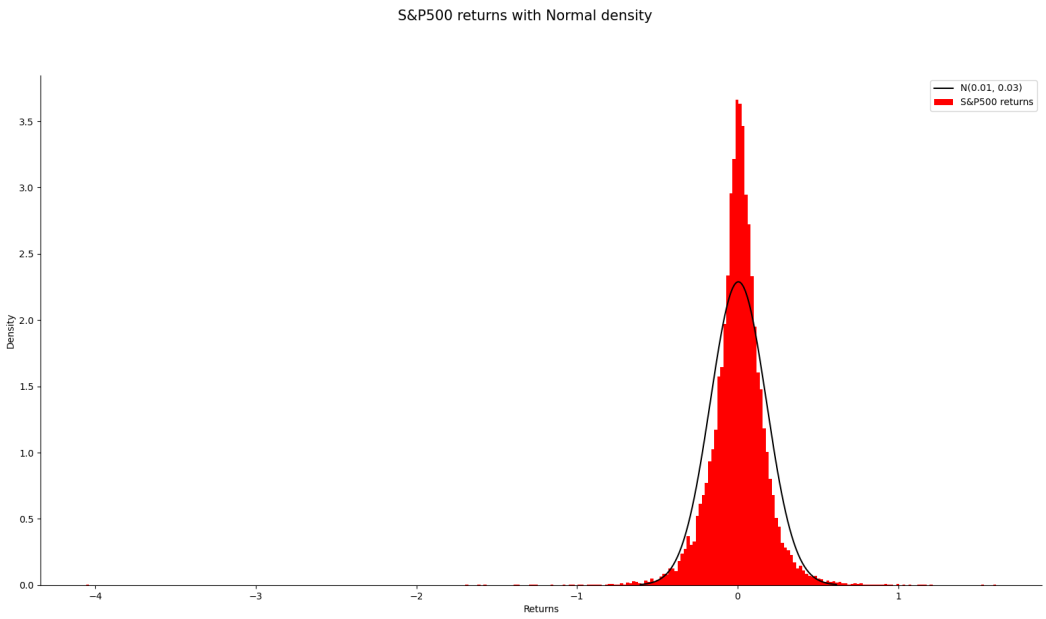

Figure 3: Histogram of the returns of the S&P500 against the Normal Distribution, $\mathcal{N}(0.01, 0.03)$

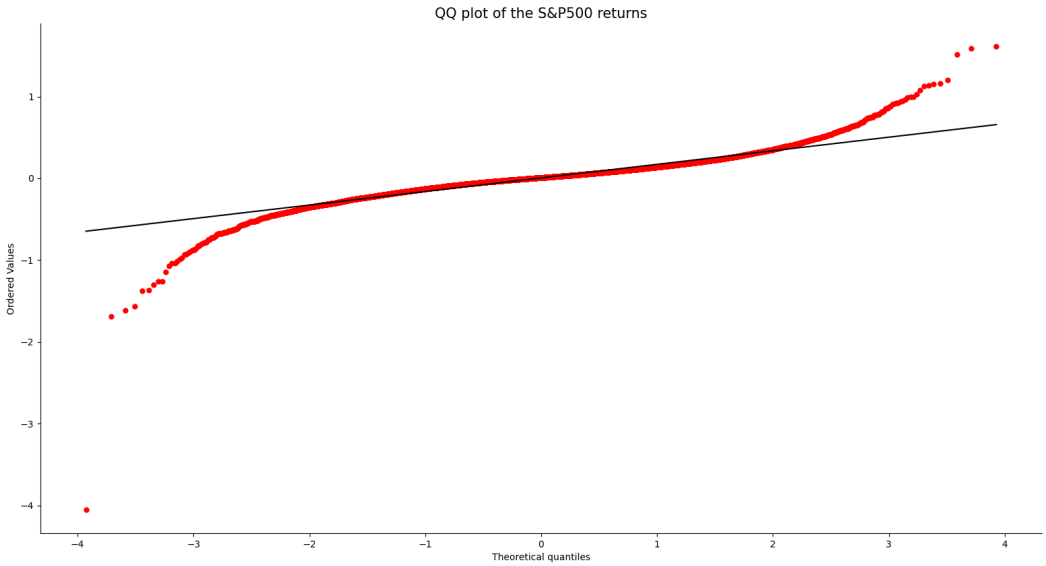

Figure 4: QQ-Plot of the returns of the S&P500

The study of tails of the distribution can also be measured via the Kurtosis coefficient, where the Kurtosis computes the impact of the distribution of the tails compared to the *Normal* distribution.

$$K(X_t) = \frac{E[(X_t - \mu)^4]}{\sigma^4}, \quad t \in \mathbb{Z}$$

where $K(X_t)$ is the value of the Kurtosis coefficient, μ is the mean and σ, the standard deviation of the returns, X_t.

A value of 3 corresponds to the *Normal* distribution while a value higher than 3 is called *Leptokurtic* distribution, which corresponds to fat tails. The Value of the Kurtosis coefficient for the log-returns of the S&P500 is equal to 25.386, which confirms the presence of fatter tails for the distribution of returns.
It is also worth mentioning that the tail corresponding to the negative returns is bigger than the tail corresponding to the positive returns. There is a Gain/Loss asymmetry which can be explained by the leverage effect, already mentioned in section 4. Below are the descriptive statistics about the distribution of the returns.

Descriptive Statistics	
	S&P500
Observations	16132
Mean	0.005030
Std. Dev.	0.174260
Min	-4.057080
25%	-0.074683
50%	0.008079
75%	0.088870
Max	1.610991
Jarque Bera	435452.018
p-value	≈ 0.00

Table 1: Descriptive Statistics of the S&P500

The Jarque-Bera test seeks to test for Normal distribution of the data using the value of the Kurtosis coefficient, K, the number of observations n and the value of the skewness coefficient S. Asymptotically, the Jarque Bera test follows a Chi-Square distribution with 2 degrees of freedom.

$$JB = \frac{n}{6}\left(S^2 + \frac{1}{4}(K-3)^2\right), \quad JB \sim \chi^2_{2ddl}$$

As the p-value is similar to 0.00, we can reject the null hypothesis and confirm that the observations do not follow a Gaussian distribution.

The returns of financial time series are the core data of our financial models in this master thesis. There are numerous benefits of using returns instead of prices in our models. First, we obtain a stationary time series, which is an important hypothesis as stated later in the work. Second, the use of returns, enables us to compare different series as they are scale-free. However, they are time dependent. Lastly, the computation of the correlation between securities is made easier when comparing the same scaled series.

6 Feature Selection

The framework of this master thesis is a multivariate analysis on the S&P500. To retrieve the multiple variables used in our model, we proceed to a feature selection of pre-selected time series. Feature selection, also known as variable selection is "the process of selecting the most useful features to use in model construction" [5]. For this matter, we use both the correlation and the mutual information.

We deliberately choose a set of ten daily times series which can be related to the Standard and Poor's 500 Index: *FTSE 100*, *Nikkei 225*, *Eurostoxx 50*, *Dow Jones*, *VIX*, *EUR/USD*, *GBP/USD*, *JPY/USD*, *10 Year Treasury Yield* and *Gold*.

All the indexes were chosen as the dependencies between markets over the world are increasing [35]. Therefore, we chose indexes over major economies in the world as well as over many time zones which might have strong links with the S&P500. Another reason for this choice is the size and influence of these indexes and rates.
The *FTSE100*, the *Nikkei225*, the *Eurostoxx50* and the *Dow Jones* are share indexes that represents the 100, 225, 50 and 30 companies with the highest market capitalisation, respectively, on the London Stock Exchange, the Tokyo Stock Exchange, in the Eurozone and in the United States. The *VIX* is a measure of the volatility of the financial market, also considered as a measure of fear. The higher the value, the more volatile the market.

The *EUR/USD*, *GBP/USD* and the *JPY/USD* are the exchange values of the euro, British pound sterling and the japan yen with respect to the US dollar. The reason for choosing the floating exchange rates as possible variables is their macro-economic impact. They are good indicators of an economy's performance and the major exchange rates traded over the world. A currency's value can directly impact the economy of a country through different channels: Merchandise, Capital flows, Inflation or Interest Rates.
Merchandise trade refers to the imported and exported goods of one country. Usually, a stronger currency induce more expensive exports to other countries, while weaker money stimulates import for other countries [16]. Therefore, the strength of the currency can gradually impact the country's trade deficit or surplus. The capital flows concerns the transaction of capital in and out of the country. A non stable currency might discourage foreign investors to move money in the country which leads to a decrease in the capital flows. In the end, the impact on the country's economy may be larger or smaller [55]. Another aspect is the inflation which is defined by the devaluation of the currency. If the

[5] Aurélien Géron, "Hands-On Machine Learning with Scikit-Learn, Keras Tensorflow", O'Reilly, 2019, p.27

currency is devalued, the results on the importers lead to higher costs [20]. Lastly, we also have the impact of the interest rate. As the central banks monitors the monetary policies, with a strong currency leads to higher interest rates [19]. All these aspects can therefore impact the US economy and therefore, the S&P500 Index.

The *10 year Treasury Yield* is the yield on a debt obligation issued by the United States government with a maturity of 10 years upon initial issuance. A 10-year Treasury note pays interest at a fixed rate once every six months and pays the face value to the holder at maturity. The U.S. government partly funds itself by issuing 10-year Treasury notes which impact the economy. Finally, we chose the *Gold* which is considered as a safe value in periods of crisis.

Before actually selecting the feature needed in our analysis, it is important to pre-process the time series. As we don't have any missing values, it is not necessary to proceed to interpolations. All the time series do not have the same length, hence we merge all the dates in the series to obtain the intersection that appears in all series. Eventually, we obtain financial series with the same length and no missing values.

In order to select our features, we proceed to two different dependence computations: the correlation and the Mutual Information.

6.1 Correlation

The correlation is defined as a linear dependency between two variables, x and y. It is computed as follows:

$$\rho_{xy} = \frac{E[(x - E(x)).(y - E(y)]}{\sqrt{E[(x - E(x))]^2 E[(y - E(y))]^2}}$$

For the feature selection, we compute the correlation between the S&P500 and all the other variables. We obtain the correlation heatmap on Figure 5 and the correlation values for the S&P500 on Table 2.

	S&P500	Dow Jones	Nikkei255	FTSE100	JPY/USD	Eurostoxx50
Corr.	1	0.99	0.94	0.72	0.54	0.48
	Gold	Sanghai Comp.	VIX	Treasury rate	GBP/USD	EUR/USD
Corr.	0.44	0.2	-0.29	0.55	-0.64	-0.65

Table 2: Table representing the correlation of the financial series with the S&P500

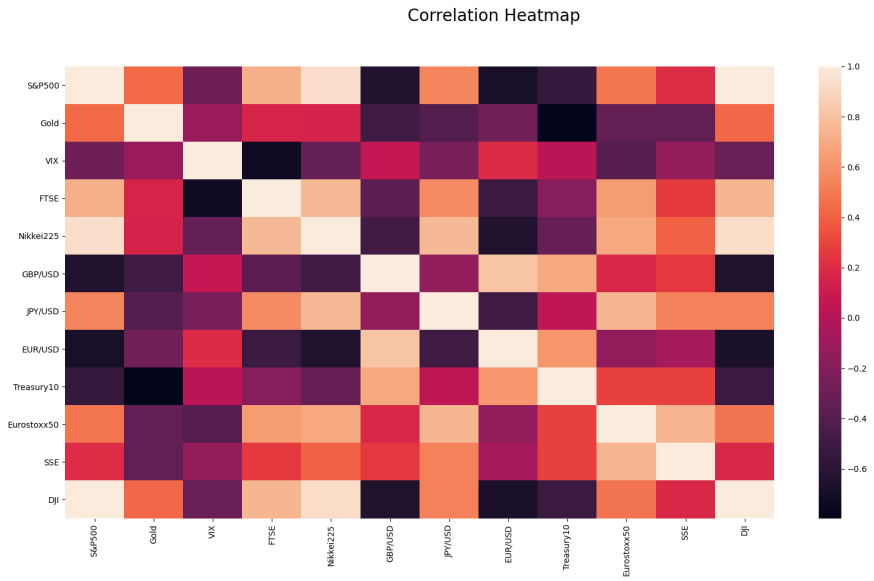

Figure 5: Correlation Heatmap of the different financial series with the colours representing the correlation values

The three times series with the highest correlation with the S&P500 are the Dow Jones, the Nikkei225 and the FTSE100.

6.2 Mutual Information

The mutual information is a measure of the dependency between two variables, x and y. The difference between the mutual information and the correlation is the nature of the dependence. In fact, the mutual information can measure the non linear dependencies. Mutual information between a random variable x and a random variable y measures how the uncertainty on y is reduced when x is known.
To understand the mutual information, we introduce the entropy of a variable y as the measure of uncertainty comprised between 0 and 1. It is computed as follows:

$$H(y) = -E[log(P(y)] = -\sum_{y} log(P(y))P(y)$$

with $P(y = 1) = p$ and $P(y = 0) = 1 - p$. The uncertainty is maximal when both events have the same probability to occur.
The conditional entropy is the measure of uncertainty on y when x is known. It is computed as follows:

$$H(y|x) = H(x, y) - H(x)$$

Note that if both x and y are independent, $H(y|x) = H(y)$, the uncertainty on y is the same as if we don't know x. Therefore we can compute the mutual information using both the entropy and the conditional entropy as being the difference between the entropy of y and the entropy of y knowing x.

$$I(y, x) = H(y)H(y|x) = H(x)H(x|y)$$

The mutual information is equal to 0 if both x and y are independent. The variables in order of mutual information are displayed Table 3, next page.

	S&P500	Dow Jones	Nikkei255	FTSE	JPY/USD	Eurostoxx50
MI	6.73	3.48	1.93	1.77	1.68	1.66
	Gold	GBP/USD	EUR/USD	Treasury rate	Sanghai Comp.	VIX
MI	1.61	1.57	1.47	1.46	1.34	0.98

Table 3: Table representing the mutual information of the financial series with the S&P500

The first three variables with the highest mutual information are the Down Jones, the Nikkei225 and the FTSE100.

Both the correlation and the mutual information are measures of dependence between variables. The choice of method to select our feature depends on the model used. In fact, it does not make sense to compute a non-linear dependence between variables if the model can't capture the non-linear dependencies. As our first model can only capture the linear tendencies and the second the non-linear tendencies, we fall between two stools. However, the first three variables are the same for both methods. Consequently, we choose to work with these first three variables being the **Dow Jones**, the **Nikkei225** and the **FTSE100** in addition to the **S&P500**.

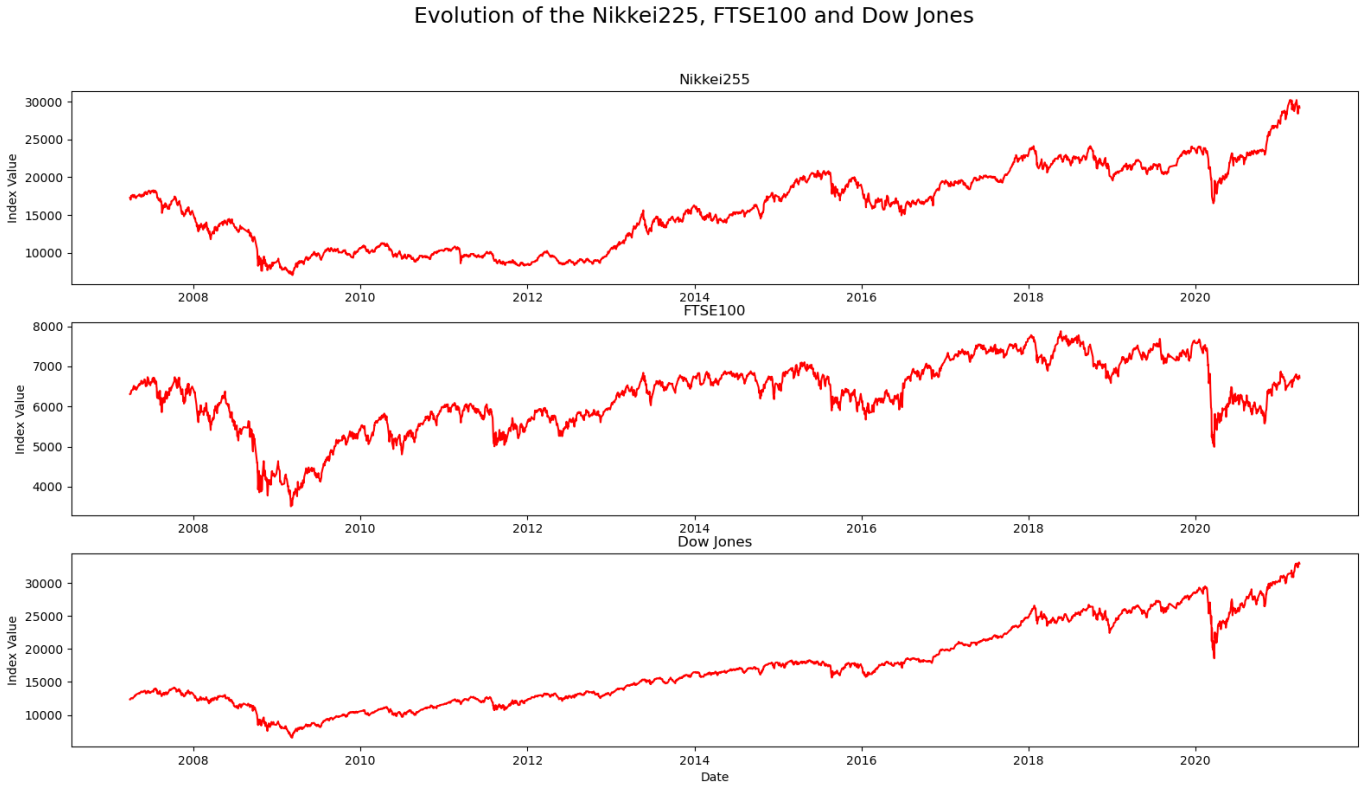

Figure 6: Representation of our three financial series being the Nikkei225, the FTSE100 and the Dow Jones

7 Benchmark Econometrics models

The first methodology adopted is the *Vector AutoRegressive Moving Average (VARMA)*, combined with the *Dynamic Conditional Correlation Generalized AutoRegressive Conditional Heteroskedasticity, (DCC-GARCH)*. The method is considered as our baseline model which is compared to a neural network later on.

The VARMA model is a multivariate version of the AutoRegressive Moving Average model, ARMA. The latter being popularised by George E. P. Box and Gwilym Jenkins in their book *Time Series Analysis: Forecasting and Control* [15]. The VARMA process allows modelling the structure of multiple time series. Intuitively, it can be seen as one ARMA process taking into account each time series lagged values. The VARMA model is used with the aim to estimate the conditional mean, μ_t of our target financial time series, S&P500. To represent the volatility of the S&P500, σ_t, we use the DCC-GARCH. This model is a multivariate version of the Generalized AutoRegressive Conditional Heteroskedasticity model developed in 1982 by Robert F. Engle.
As we work on a multivariate statistical model with time series, it is necessary to state several mandatory hypotheses before estimating our models. The mathematical notations in this section come from the book *"A. McNeil, R. Frey and P. Embrechts,Quantitative Risk Management,Princeton University Press, 2005, pg 164-177"*.

7.1 Basics of Multivariate Time Series Analysis

We consider a d-dimensional multivariate set of stochastic processes of log-returns $(\mathbf{X}_t)_{t\in\mathbb{Z}} = (X_1, X_2, ..., X_d)'$, with $X_t = (x_{t1}, x_{t2}, .., x_{tn})$, as a family of random vectors defined by the probability space $(\Omega, \mathcal{F}, P)$ where

- Ω is the sample space, the set of all possible outcomes.
- $\mathcal{F}$ is the event space, which is the set of outcomes in the sample space.
- P is the probability assigned to each event in the event space.

Assuming the first and the second moments of the random vectors of returns, $\mathbf{X}_t$, exist, we define them as the mean $\boldsymbol{\mu}_t$ and the covariance matrix function, $\boldsymbol{\Sigma}_t$. The first moment is specified by,

$$E[\mathbf{X}_t] = (E[\mathbf{X}_1], E[\mathbf{X}_2], ..., E[\mathbf{X}_d])'$$

$$with\ E(\mathbf{X}_t) = \boldsymbol{\mu}(t), \quad t \in \mathbb{Z}$$

And the second moment, known as the covariance, can be determined as,

$$cov(\mathbf{X}_k, \mathbf{X}_s) = E((\mathbf{X}_k - E(\mathbf{X}_k))(\mathbf{X}_s - E(\mathbf{X}_s))')$$

$$with\ cov(\mathbf{X}_k, \mathbf{X}_s) = \Gamma(k, s) \quad t, s \in \mathbb{Z}$$

with $\Gamma(s, s) = var(\mathbf{X}_s)$.

7.1.1 Strict Stationarity

The first hypothesis states that the multivariate set of time series $(\mathbf{X}_t)_{t\in\mathbb{Z}}$ is strictly stationary. The stationarity is defined as,

$$(\mathbf{X}'_{t_1}, \mathbf{X}'_{t_2}, ..., \mathbf{X}'_{t_d}) \stackrel{\mathrm{d}}{=} (\mathbf{X}'_{t_{1+k}}, \mathbf{X}'_{t_{2+k}}, ..., \mathbf{X}'_{t_{d+k}})$$

for all $t_1, ...t_d, k \in \mathbb{Z}$ and for all $n \in \mathbb{N}$

This equality implies that the joint distribution of the $X'_{t1}, X'_{t2}, ..., X'_{tn}$ is the same as $X'_{t1+k}, X'_{t2+k}, ..., X'_{tn+k}$. In other words, strict stationarity means that the joint distribution only depends on k and not the time. To evaluate the hypothesis of stationarity, we use the *Dickey-Fuller* test. It states for the null hypothesis that a unit root is present in the auto-regressive model where a unit root is defined as a drift in the time series. Therefore, it is non stationary. For the alternative hypothesis, no unit root is present and hence the series is stationary.

7.1.2 Covariance Stationarity

The time series $(\mathbf{X}_t)_{t\in\mathbf{Z}}$ is covariance stationary if both the first and the second moments exist and satisfy the conditions whereby the values are stable over time. Mathematically, the conditions come down to,

$$E(\mathbf{X}_t) = \boldsymbol{\mu}, \quad \text{for all } t \in \mathbb{Z}$$

$$\Gamma(t, s) = \Gamma(t + k, s + k), \quad \text{for all } t, s, k \in \mathbb{Z}$$

The covariance stationarity suggests that both times series t and s, the covariance depends only on the temporal separation. This is defined as the lag.

7.1.3 Multivariate White Noise

Intuitively, a white noise process is a series that has no discernible structure. Hence, we defined the white noise as a series with constant mean, constant variance and the autocovariance equal to zero, except at lag zero.
The series $(\mathbf{X}_t)_{t\in\mathbb{Z}}$ is called multivariate white noise such that it is covariance stationary and the correlation matrix function is given by

$$P(h) = \begin{cases} \mathrm{P}, & \mathrm{h}{=}0 \\ 0, & \mathrm{h} \neq 0 \end{cases}$$

for some positive-definite correlation matrix P. The multivariate white noise process with a mean of 0 and a covariance matrix Σ is denoted as $\mathbf{WN}(0, \Sigma)$. Similar to the univariate white noise process, there is no cross-correlation between series except for the lag 0.

The multivariate strict white noise is a special case of multivariate white noise where the series is constructed of independent and identically distributed random variables with a finite covariance matrix.

7.1.4 Multivariate Martingale Difference

The martingale difference notion requires the introduction of the concept of martingale and filtration $\mathcal{F}_t$. The filtration is a notion used to model the information over time. We say that X_t is adapted to some filtration $\mathcal{F}_t$ if X_t is measurable with respect to $\mathcal{F}_t$. This means that the information known about X_t changes over time. A sequence of random variables, X_t, is said to be martingale if the most probable value at time t, knowing the information at time k is the value X_k. Mathematically, we define the martingale as,

$$E[X_t|\mathcal{F}_s] = X_s$$

We say that $(\mathbf{X}_t)_{t\in\mathbb{Z}}$ has the multivariate martingale difference property with respect to $\mathcal{F}_t$ if $E[\mathbf{X}_t] < \infty$ and

$$E[\mathbf{X}_t|\mathcal{F}_{t-1}] = 0, \quad \forall t \in Z$$

The unconditional mean of this process is zero. Lastly, we can define the the multivariate white noise process as a martingale-difference if $cov(\mathbf{X}_t) < \infty$ for all t and if the covariance matrix function satisfies $\Gamma(t, s) = 0$ for $t \neq s$.

Recall the financial series of returns, X_t. In an effort to model the process, it is useful to observe that the time series are constructed using white noise as a building block,

$$X_t = \mu_t + \epsilon_t, \quad t \in \mathbb{Z}$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma_t^2)$. Thereafter, we make the assumptions that the white noise process ϵ_t is a martingale-difference with respect to $\mathcal{F}_t$. Like so, we can state that $E[X_t|\mathcal{F}_{t-1}] = \mu_t$. To put it another way, the main purpose of the first model is to use the VARMA process to model the conditional mean of the returns and retrieve the white noise series with a specific variance dynamic. Later on, we focus on modelling the dynamics of the conditional variance, $Var[\epsilon_t|F_{t-1}]$.

7.2 VARMA Process

The Vector Autoregressive Moving Average process with order p and q is a statistical process that provides a multivariate description of time series in terms of polynomials. The first polynomial is the AutoRegressive (AR) component, of the time series and the second is the Moving Average (MA) component. The AR component implies modelling the series on its own lagged values, $X_0, X_1, ..., X_{t-1}$. The MA component implies modeling the error component resulting from the lagged values in the past, $\epsilon_0, \epsilon_1, ..., \epsilon_{t-1}$. Both components are defined by their order which delimits how many lags to take into account in the model. Therefore, the VARMA model is defined by the orders p and q which corresponds respectively to the AR order and the MA order.

Let $\boldsymbol{\epsilon}_{t\in Z}$ be a multivariate $WN(\mathbf{0}, \boldsymbol{\Sigma}_\epsilon)$ process. $(\mathbf{X}_t)_{t\in\mathbf{Z}}$ is a mean-μ VARMA(p,q) process if it is a covariance-stationary process satisfying the equation

$$\mathbf{X}_t - \boldsymbol{\Phi}_1\mathbf{X}_{t-1} - ... - \boldsymbol{\Phi}_p\mathbf{X}_{t-p} = \boldsymbol{\epsilon}_t + \boldsymbol{\Theta}_1\boldsymbol{\epsilon}_{t-1} + ... + \boldsymbol{\Theta}_q\boldsymbol{\epsilon}_{t-q}, \quad \forall t \in \mathbb{Z}, \quad \epsilon_t \sim WN(0, \sigma_\epsilon^2)$$

$$\mathbf{X}_t = \boldsymbol{\Phi}_1\mathbf{X}_{t-1} + ... + \boldsymbol{\Phi}_p\mathbf{X}_{t-p} + \boldsymbol{\Theta}_1\boldsymbol{\epsilon}_{t-1} + ... + \boldsymbol{\Theta}_q\boldsymbol{\epsilon}_{t-q} + \boldsymbol{\epsilon}_t, \quad \forall t \in \mathbb{Z}, \quad \epsilon_t \sim WN(0, \sigma_\epsilon^2)$$

for matrix $\boldsymbol{\Phi}_i$ and $\boldsymbol{\Theta}_i$ in $\mathbb{R}^{dxd}$. $(\mathbf{X}_t)_{t\in\mathbf{Z}}$ is a vector of values at time t for each time series in $\mathbb{R}^{dx1}$.

Following the theoretical point above, we can easily apply it to the case of the time series of returns. Let the series of returns $(\mathbf{X}_t)_{t\in\mathbf{Z}}$ be covariance-stationary and $\boldsymbol{\epsilon}_{t\in Z}$ be a multivariate $WN(0, \boldsymbol{\Sigma}_\epsilon)$ process,

$$\mathbf{X}_t = \boldsymbol{\mu}_t + \boldsymbol{\epsilon}_t, \quad t \in \mathbb{Z}$$

$$where \quad \boldsymbol{\mu}_t = E[\mathbf{X}_t|\mathbf{X}_{t-1}] = \boldsymbol{\Phi}_1\mathbf{X}_{t-1}+...+\boldsymbol{\Phi}_p\mathbf{X}_{t-p}+\boldsymbol{\Theta}_1\boldsymbol{\epsilon}_{t-1}+...+\boldsymbol{\Theta}_q\boldsymbol{\epsilon}_{t-q}+\boldsymbol{\epsilon}_t, \quad \forall t \in \mathbb{Z}, \quad \forall t \in \mathbb{Z}$$

7.2.1 Model Results

The analysis has been made using the language Python and the package statsmodels. The times series start on April 2nd of 2007 and end on April 1st, 2021 (2954 observations). Our test set is equal to 10% of the total series, and thus starts on November 11th, 2019 (296 observations).

We assume we have a random sample of returns $(X_1, X_2, X_3, X_4)'$ from a stationnary time series model $(X_t)_{t\in\mathcal{Z}}$. We can verify this hypothesis using the *Dickey Fuller test.*

Dickey Fuller Test		
	ADF statistic	p-value
S&P500	-9.506	≈ 0.00
Nikkei225	-12.993	≈ 0.00
FTSE100	-10.224	≈ 0.00
Dow Jones	-9.9	≈ 0.00

Table 4: Results of the Dickey Fuller test on the four times series

All the series have a p-value approximately equal to 0.00. Therefore we can reject the null statement of non-stationarity and confirm the hypothesis stated on our set of returns.

The analysis implies inferring orders (p, q) about the VARMA process. To do so, we compute estimations of the autocorrelations from this series of returns and use these estimates to make a hypothesis about the dependence structure of the underlying process. The autocovariance $\gamma(t, s)$ is defined as the interdependence between the observations of a series.

$$\gamma(t, s) = E[(x_t - \mu_t)(x_s - \mu_s)], \quad \forall t, s \in \mathbb{N}$$

From the autocovariances, we calculate the autocorrelation function:

$$\rho(h) = \gamma(h)/\gamma(0), \quad 0 \leq h < n$$

We proceed to the same operations for the partial autocorrelation function. The partial autocorrelation function contrasts with the autocorrelation function since it does take into account its own lagged values. To facilitate the interpretations, we plot both the autocorrelation and partial autocorrelation. The order inference of the VARMA process can be deduced from the value of the autocorrelation and partial autocorrelation at each lag.

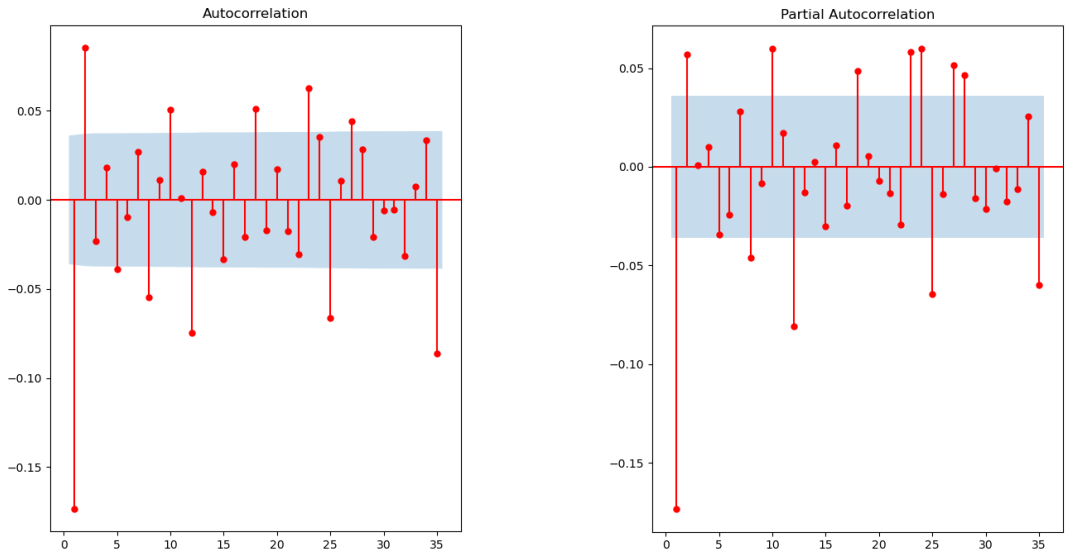

Figure 7: ACF and PACF plots of the returns of the S&P500

Above are displayed the graphs of the Autocorrelation Function (ACF) and the Partial Autocorrelation Function (PACF) of the S&P500. The autocorrelation shows a value above the 95% confidence intervals for the first two lags while quickly dropping under the significant threshold. The lags 8, 10 and 11 are also above the confidence intervals. However they don't seem recurrent along the weeks.
Considering the partial autocorrelation graph, we can also distinguish a very significant

negative value for the first lag and a value for the second lag being also above the minimum confidence level. Looking at the ACF and PACF graphs of the other financial times series, we end up with the same conclusions. All things considered, we can make inferences on the lags to take into account in our VARMA model. As the first lags in both graph is very high, we can make a hypothesis on a VARMA(1, 1) and a VARMA(2, 2). It would be a plus to test also a combination of VARMA(1, 2) and VARMA(2, 1).

In order to determine the best order in our model, we can perform multiple estimations using statistical criteria such as the Aikike Criterion (AIC) or the Bayesian Information Criterion (BIC). Both indicators inform us about the quality of the model. The Aikike Criterion is computed using the number of parameters in the model as well as the value of the maximum likelihood, while the Bayesian criterion penalises the addition of parameters in the model. In both measures, the lower the value, the better the model is.

AIC and BIC for multiple VARMA models		
	AIC	BIC
VARMA(1, 1)	-80727.476	-80455.858
VARMA(1, 2)	-80741.752	-80394.279
VARMA(2, 1)	-80751.563	-80404.090
VARMA(2, 2)	-80721.473	-80278.145

Table 5: AIC and BIC of the four different VARMA models

We can observe a lower value of the BIC for the model VARMA(1, 1). The lower value for the AIC is for the VARMA(2, 1). Following the famous saying "the simpler the model is, the better", we keep as our baseline VARMA, the orders (1, 1). We support this decision based on the value of the BIC and the fact that the AIC does not penalise the addition of parameters. This means that our forecast of returns in t depends on the returns in $t-1$ and the forecast error of $t-1$. The vector of output is of size $4x1$ and the matrices of parameters are both of size 4×4.

As the orders are cleared, we can train the model via maximum likelihood. A clear explanation of the Maximum likelihood estimation is displayed in the Appendix A. In the framework of our analysis, the model is re-trained multiple times. Therefore, we can't analyse each parameter of every model. Knowing this, we take a look at the parameters trained on the full set of data for a pure observation purpose, Table 13, Appendix B. In order to not overload the Appendix, we only transcript the coefficient relative to the S&P500 returns. We observe a negative coefficient for the lag over the S&P500 and over the Nikkei and FTSE error terms. We also notice overall low z-values and high standard errors for the S&P500 and Dow Jones components. The model estimated a grand total of 42 coefficients to retrieve the forecast over the four times series as well as the residuals.

As we are making our analysis on two different time horizons, one-day ahead and ten-days ahead, the methodologies are slightly different. First, for the one-day ahead conditional mean estimation, we train a model until time t to estimate the conditional mean in time $t + 1$. We made the choice to retrain the model after each estimation of $t + 1$ to have the best forecast possible. The Figure 8, next page, represents our results on the one-day ahead forecasts.

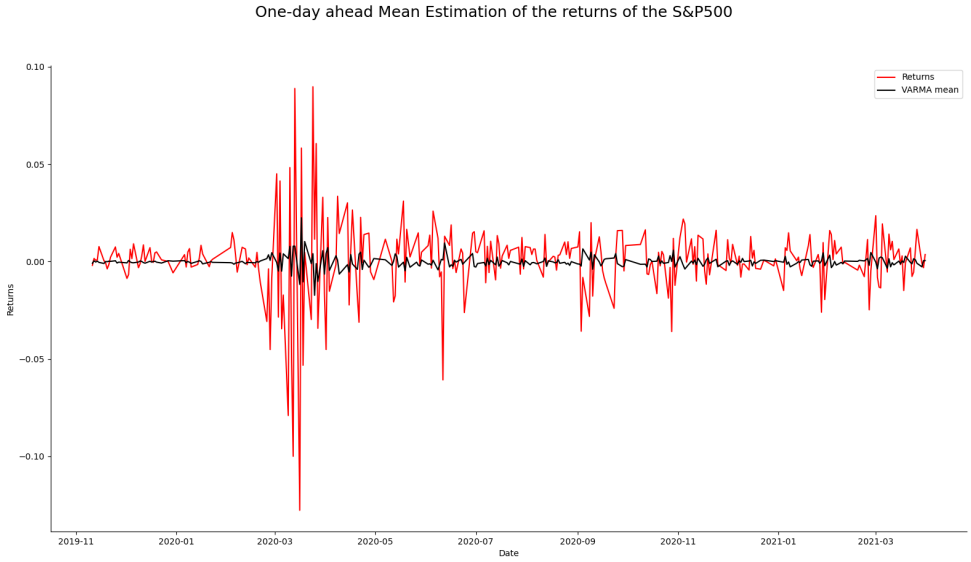

Figure 8: One-day ahead Mean estimation of the S&P500 with the VARMA(1, 1) model

We can observe a mean estimation around the value of 0. It is close to being constant in periods of low volatility and more dissipated in the periods of high volatility. The Covid-19 period results the highest peaks of variance. The punctual and abrupt shocks of June and September 2020 do not seem to be detected, as opposed to repetitive ups and downs visible in March 2020. We conclude that a couple of days are needed for the model to adapt its mean in periods of crisis. This is due to the impact of the coefficient of the error components that acts in a "delayed manner" over the output.

In an attempt to compare the models with each other, we compute the Sum of Squared Errors (SSE) between the returns and the forecast.

$$SSE = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2, \quad y_i \, being \, the \, target \, value \, and \, \hat{y}_i \, the \, forecast.$$

The SSE for the one-day ahead estimation of the conditional mean is equal to 0.1058.

As for the 10-days ahead estimation, we have to train the model until t such as the preceding experience but we estimate the conditional mean from $t+1$ until $t+10$. On the results, Figure 9 next page, we can observe a significant change in the dynamics of the estimated conditional mean. The value of the Sum of Squared Errors over all time steps is equal to 1.11666.

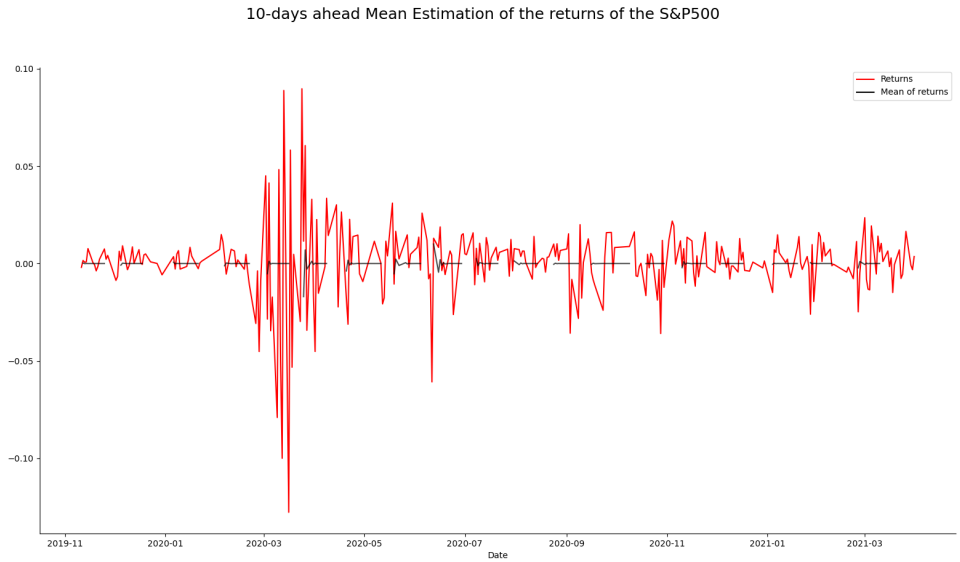

Figure 9: 10-days ahead mean estimation of the S&P500 with the VARMA(1, 1) model

Looking at a time step of 10 days, the first couple of values are not constant and are estimated with a variance higher in periods of volatility. In periods of low volatility, the first values are closer to zero. However, most values after the third day in the time step are estimated as 0. The further we go in the days, the closer the mean is estimated being equal to 0. This is valid no matter the period which we lay in. This is explained by the *law of iterated expectation*.

Suppose that $\mathcal{F}_{t-k} \subset \mathcal{F}_t$ for all $k \geq 1$ and all t. The information one has at time t contains the information one has at time $t-k$. Then, for some random variable X_t, $k \geq 1$ and $0 \leq i \leq k$,

$$E[X_t \,|\mathcal{F}_{t-k}] = E\{E[X_t \,|\mathcal{F}_{t-i}] \,|\mathcal{F}_{t-k}\}$$

This is equivalent to

$$E\{X_t - E[X_t \,|\mathcal{F}_{t-i}] \,|\mathcal{F}_{t-k}\} = 0$$

$$E[X_t \,|\mathcal{F}_{t-2}] = E\{E[X_t|\mathcal{F}_{t-1}] \,|\mathcal{F}_{t-2}\}$$

The forecast you do the day before yesterday for today is the same as the expectation of the forecast you would do yesterday. Taking the limits of the expression gives,

$$lim_{k\to\infty} E[X_t \,|\mathcal{F}_{t-k}] = E[X_t]$$

We observe that when decreasing the information set, the expectation of the conditional mean tends to the expectation of the unconditional mean whereby this value is close to zero. Therefore, we can confirm the behaviour of the VARMA for 10-days ahead forecasting. The further we try to forecast, the smaller the information set is and therefore, the closer we get to the unconditional mean.

The conditional mean of our multivariate set of returns modelled, we have to focus on our residuals, $\hat{\epsilon}_t$. They should behave as a realisation of white noise process, since this is our model assumptions for the innovations. This can be assessed by constructing the plot of ACF and PACF of the residuals, Figure 28, Appendix B. The first lags are not significant and therefore, the value of the residuals do not have any autocorrelation.
We can also perform the Ljung-Box test on the residuals to highlight the asymptotic behavior of the residuals following a white noise process. The Ljung-Box test has a null hypothesis of strict white noise.

$$\mathcal{Q}_{LB} = n(n+2)\sum \frac{\hat{\rho}(j)^2}{n-j}, \quad \mathcal{Q}_{LB} \sim \chi^2_h$$

which has an asymptotic chi-squared distribution with h degrees of freedom.
With a p-value of 0.087, for a Ljung-Box test over the 5 first lags, we can keep the null hypothesis being the absence of autocorrelation of the residuals. So we can assume the following,

$$\boldsymbol{\epsilon}_t \sim WN(\mathbf{0}, \boldsymbol{\Sigma}_\epsilon), \quad \forall t \in \mathbb{Z}$$

Since the residuals follow a white noise process, confirmed by the Ljung-Box test, by definition, we can not make any VARMA inference on it. However, we can focus on the variance dynamics of the process. We use a multivariate GARCH process, known as DCC-GARCH to model the variance behavior of those residuals. This, with the aim of modeling the volatility clustering and simulate the series.

7.3 DCC-GARCH process

Recall the definition of the returns,

$$\mathbf{X}_t = \boldsymbol{\mu}_t + \boldsymbol{\epsilon}_t, \quad t \in \mathbb{Z}$$

$$where \quad \boldsymbol{\epsilon}_t \sim WN(\mathbf{0}, \Sigma_\epsilon), \quad \forall t \in \mathbb{Z}$$

We can construct the white noise process, $\boldsymbol{\epsilon_t}$ driven by a mean of 0 and a conditional variance of $\mathbf{\Sigma}_t$, using a strict white noise process with mean 0 and the identity matrix as the variance-covariance matrix. Let's consider $\mathbf{Z}_t$ to be a multivariate strict white noise process, $\mathbf{Z}_t \sim SWN(\mathbf{0}, \mathbf{I}_d)$. The process $(\boldsymbol{\epsilon}_t)_{t\in\mathbb{Z}}$ is a multivariate GARCH process if it is strictly stationnary and satisfies the equation,

$$\boldsymbol{\epsilon}_t = \Sigma_t^{1/2}\mathbf{Z}_t, \quad t \in \mathbb{Z}$$

where $\Sigma_t^{1/2} \in \mathbb{R}^{dxd}$ is the Cholesky factor of the positive covariance matrix Σ_t, measurable with respect to $\mathcal{F}_{t-1}$. We can easily demonstrate the property of martingale-difference characterizing this type of covariance stationnary process.

$$E[\boldsymbol{\epsilon}_t \,|\mathcal{F}_{t-1}] = E[\Sigma^{1/2}\mathbf{Z}_t \,|\mathcal{F}_{t-1}] = \Sigma^{1/2}E[\mathbf{Z}_t] = 0$$

$(\boldsymbol{\epsilon}_t)_{t\in\mathbb{Z}}$ must therefore be a multivariate white noise process. Σ_t is considered as the conditional covariance matrix of $(\boldsymbol{\epsilon}_t)_{t\in\mathbb{Z}}$ since,

$$cov(\boldsymbol{\epsilon}_t \,|\mathcal{F}_{t-1}) = E[\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_t^{'} \,|\mathcal{F}_{t-1}] = \Sigma_t^{1/2}E[\mathbf{Z}_t\mathbf{Z}_t^{'}](\Sigma_t^{1/2})^{'} = \Sigma_t^{1/2}(\Sigma_t^{1/2})^{'} = \Sigma_t$$

In a comparison framework between the multivariate and univariate process, the conditional covariance matrix Σ_t is similar to the σ_t^2 in a univariate GARCH model. We denote the elements of Σ_t by $\sigma_{t,ij}^2$ and also use the notation $\sigma_{t,i} = \sqrt{\sigma_{t,ii}^2}$ to denote the conditional standard deviation, or volatility at time t of the ith component in the multivariate set of series $\mathbf{X}_t$. In practice, the innovations of our model are taken from a Gaussian distribution, $Z_t \sim N(0,1)$ because we only simulate one series of returns. However, we could use the multivariate Gaussian distribution to simulate multiple series. The use of other distributions is also possible and might represent a more realistic distribution of innovations.

Recall the covariance matrix Σ_t, we can develop the expression as

$$\Sigma_t = \Delta_t P_t \Delta_t$$

where Δ_t is the diagonal matrix, known as the volatility matrix and P_t is the conditional correlation matrix. In order to understand the development of the Dynamic conditional correlation model, DCC-GARCH, we need to introduce the Constant conditional correlation model, CCC-GARCH. The process X_t is a CCC-GARCH if it is a process with a conditional covariance matrix of the form

$$\Sigma_t = \Delta_t P_c \Delta_t$$

where P_c is a constant, positive-definite correlation matrix and Δ_t is a diagonal volatility matrix with elements $\sigma_{t,k}$ satisfying,

$$\sigma_{t,k}^2 = \alpha_{k,0} + \sum_{i=1}^{p_k} \alpha_{ki} X_{t-i,k}^2 + \sum_{j=1}^{q_k} \beta_{kj} \sigma_{t-j,k}^2, \quad k = 1, ..., d$$

where $\alpha_{k0} > 0$, $\alpha_{ki} \geq 0, i = 1, ..., p_k, \beta_{kj} \geq 0, j = 1, ..., q_k$.

Hence, P_c is constant and Δ_t is modeled over a simple univariate GARCH processes. The correlation between variable stays constant while the variance of each series is conditional over time. The final equation can be written as $\boldsymbol{\epsilon}_t = \Delta_t P_c^{1/2} \mathbf{Z}_t$, which is equal to $\boldsymbol{\epsilon}_t = \Delta_t \tilde{\mathbf{Z}}_t$ for $\tilde{\mathbf{Z}}_t \sim SWN(0, P_c)$ process. If $\boldsymbol{\epsilon}_t$ is a covariance stationary process, then each component series is a covariance stationary GARCH process for which a necessary and sufficient condition is $\sum_{i=1}^{p_k} \alpha_{ki} + \sum_{j=1}^{q_k} \beta_{kj} < 1$.

The Dynamic Conditional Correlation model, DCC-GARCH generalises the constant conditional correlation model to allow conditional correlations to evolve dynamically. Therefore, the main change is the matrix P_c which was constant in the CCC-GARCH that becomes dependent on time in the DCC-GARCH.

The process $\boldsymbol{\epsilon}_t$ is a DCC-GARCH process where the volatility's comprising Δ_t follows univariate GARCH specifications such as in the previous model, and the conditional matrices P_t, satisfy, for $t \in \mathbb{Z}$, the equation:

$$P_t = \phi((1 - \sum_{i=1}^{p} \alpha_i - \sum_{j=1}^{q} \beta_j) P_c + \sum_{i=1}^{p} \alpha_i Y_{t-i} Y' t - i + \sum_{j=1}^{q} \beta_j P_{t-j})$$

where P_c is a positive-definite correlation matrix, ϕ is the operator that extracts the correlation matrix, Y_t denotes the devolatilized process, and the coefficients satisfy $\alpha_i >= 0, \beta_j >= 0$ and $\sum \alpha_i - \sum \beta_j < 1$.

7.3.1 Model Results

The model has been trained using a plug-in for the language R in the Python script. We use the package rmgarch available on Cran.

Such as with the VARMA(p, q) model, we estimate multiple orders of DCC-GARCH to evaluate their AIC and BIC. Following the results of Table 6, the model orders are chosen to be (1, 1). Consequently, the standard deviation depends both on the value of the standard deviation in $t - 1$ and the error on the forecast in $t - 1$. The model takes into account 4 times series and therefore estimates 20 different parameters presented in Table 14, Appendix B. Most of the coefficients have significant values. There is only one negative coefficient which is not significant and close to zero.

AIC and BIC for multiple DCC-GARCH models		
	AIC	BIC
DCC-GARCH(1, 1)	-29.186	-29.131
DCC-GARCH(1, 2)	-29.186	-29.128
DCC-GARCH(2, 1)	-29.186	-29.128
DCC-GARCH(2, 2)	-29.186	-29.126

Table 6: AIC and BIC of different DCC-GARCH orders

This estimation is realised using the maximum likelihood function, explained in Appendix A, and returns a matrix of size 4×4 corresponding to the variance-covariance matrix of all the variables in time t. The results for the one-day ahead estimation of the residuals' volatility of the S&P500 is displayed on Figure 10 below.

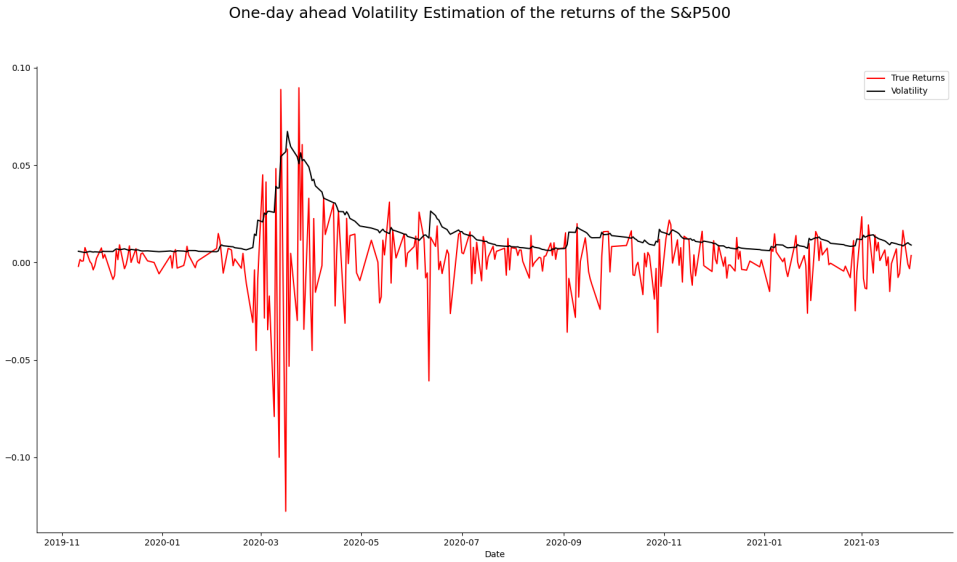

Figure 10: One-day ahead estimation of the volatility with the DCC-GARCH(1, 1) model

The model captures very well the high volatility period of the Covid-19 crisis. The different brutal peaks of volatility in June and September are also forecasted by the model as it adapts well to the high risks periods. The one-day ahead estimations are never evaluated to zero. We can also observe that the estimated conditional volatility increases drastically in periods of high movements and tends to decrease slowly over time to represent the volatility persistence, as the Covid-19 period shows it well. In an attempt to compare the models with each other, we compute the Sum of Squared Error between the forecasted standard deviation of the residuals and the square root of the residuals' conditional variance. The value is equal to 0.1078.

Concerning the 10-days ahead forecast, the estimations are present in Figure 11. Again, we see a radical change in the volatility forecasts as they show constant volatility estimation for the 10 days-ahead values. The standard deviations are slightly higher in periods of crisis, such as in April of 2020. But overall, the estimations are constant

over time, especially from May 2020 until March of 2021. Concerning the constant factor over the time step of 10 days, it is again due to the *law of iteration* where, with a decrease of information, the model tends to estimate the unconditional standard deviation. The value of the SSE for the model is equal to 328.583.

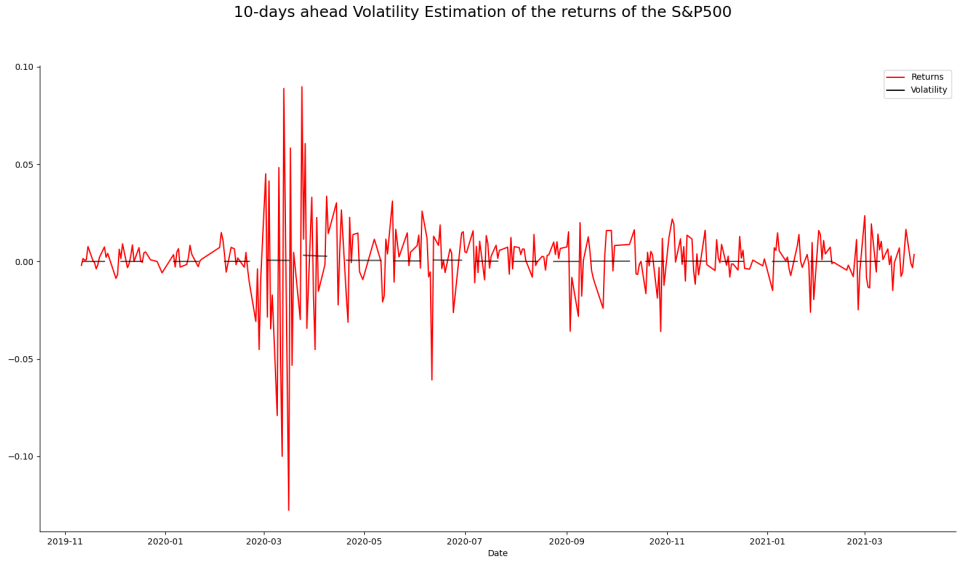

Figure 11: 10-days ahead estimation of the volatility with the DCC-GARCH(1, 1) model

7.4 Conclusion

In this section, we managed to model the conditional mean of returns with the help of a VARMA process and the conditional standard deviation of residuals with the help of the DCC-GARCH. We built our models with the utmost statistical rigour, even though we did not manage to retrieve the confidence intervals out of our models. The results were coherent and explained by statistical laws.
The models above seek to represent the conditional mean of the returns and conditional volatility of the residuals with a classical econometric process. In the next section, we focus on the conditional mean and standard deviation modeled with the help of a multivariate neural network. The process is deliberately the same where we first work on the mean and thereafter the volatility.

8 Deep Learning and Neural Networks

Biomimicry is *the practice of making technological and industrial design copy natural processes* [6]. The discipline studies the biological behaviour and processes to find plausible solutions to the problems humans are facing. Perhaps one of the biggest biomimicry invention to this day is planes. From an economic point of view, decades ago, humans relied on ships to transport cargoes from one point to another. This implies for the two destinations to be accessible by boat, which is not always the case. A revolution in the world of transportation was the introduction of planes, which erased the problem of accessibility encountered by boats. Engineers based their design on bird wings where the air can flow on the surface smoothly. It is shaped such that it creates low pressure above and high pressure under the wing and therefore creates a force to lift the plane [33].
Another example would be the invention of bike helmets. The woodpecker has the astonishing ability to dig in trees at a rate of 20 times per second [47]. This creates impacts 100 times larger than a human brain cell can tolerate. However, their skull is made of a tongue-bone structure, which protects the woodpecker from the massive deceleration forces at work. This is this spongy bone structure that has inspired the design of biker helmets [47].
A last example would be the wingsuits created for base jumping. Highly inspired from flying squirrels, these suits allow humans to modify the airflow along their bodies to increase lift and be able to glide in the sky at speed of around 200 km/h.

Biomimicry is the very core idea of **neural network** as it is inspired by the brain's architecture. The brain is largely composed of cell bodies containing complex biological components, which keep the cells alive and run the complex organ that is our brain. The cells are connected to each other via the *synapses* which are used to conduct electrical pulses to release *neurotransmitters*. When a neuron receives a sufficient number of these neurotransmitters within a few milliseconds, it fires its own electrical impulses in its turn [27]. Thus, individual neurons seem to behave in a rather simple way, but they are organized in a vast network of billions.

The Neural Network (NN) is highly inspired by this architecture. It is composed of neurons and synapses, connected between each other in a layer-based architecture. They were introduced in the year 1943 by the neurophysiologist Warren McCulloch and the mathematician Walter Pitts [48]. Since then, the field has evolved a lot with groundbreaking architecture which applies to their own specific cases, from times series to voice data, images and videos. Today the field of deep learning is very popular with continuous publications and promising results.

[6] Cambridge Dictionnary, https://dictionary.cambridge.org/fr/dictionnaire/anglais/biomimicry

8.1 The Multilayer Perceptron and Backpropagation

8.1.1 Multilayer Perceptron

To introduce the concept of neural network, we start by one of the most popular NN, the *multilayer perceptron* (Multilayer Perceptron (MLP)). The MLP can be thought as a complex non-linear function that maps the set of inputs to the set of outputs. The NN is composed of one input layer, one or more hidden layers, and one final output layer. When an Artificial Neural Network (ANN) contains more than one set of hidden layers, we usually talk about deep neural network. Each layer consists of neurons fully connected via synapses to the following layer. The output layer can be either composed of one neuron and therefore produce one output, or multiple neurons and thereby produce multi-outputs. The MLP can be used for both regression or classification purpose. Also called *feed-forward neural networks*, the MLP is a type of network where the flow of computation goes one way, from the input to the output.

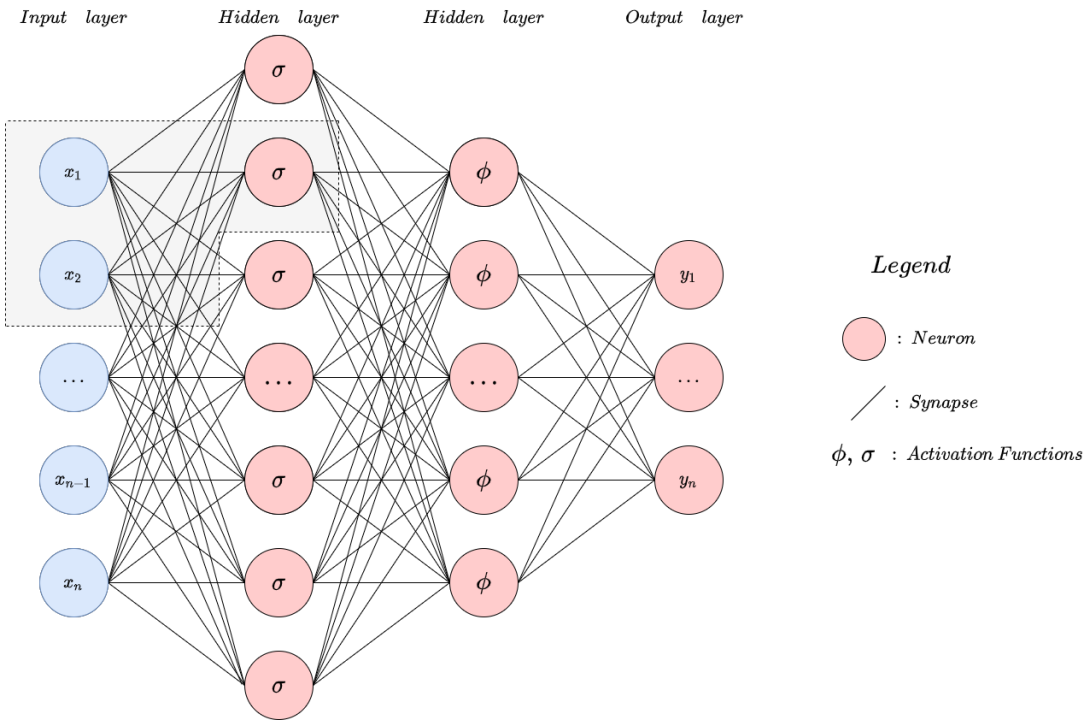

Figure 12: Representation of a Multilayer perceptron.

The above figure displays a multilayer perceptron with one layer of input, 2 hidden layers and a layer of outputs. The inputs are represented by the vector $\mathbf{X}$ and the output by the vector $\mathbf{Y}$. Each synapse is associated to a weight, w^i_{jk} where i is the neuron's layer's number, j is the preceding neuron's number and k is the next neuron's number it is connected to. Each neuron is characterised by an *activation function*, σ or ϕ in our example, and process the inputs to produce an output that is forwarded to the next layer. An *activation function* is a non-linear function applied to the input of each neuron. This function must have the particularity to be continuously differentiable in order for the network to be trained. Note that each layer, except the output, includes a bias neuron. A detailed schema of the grey area in Figure 12 is represented on Figure 13, next page.

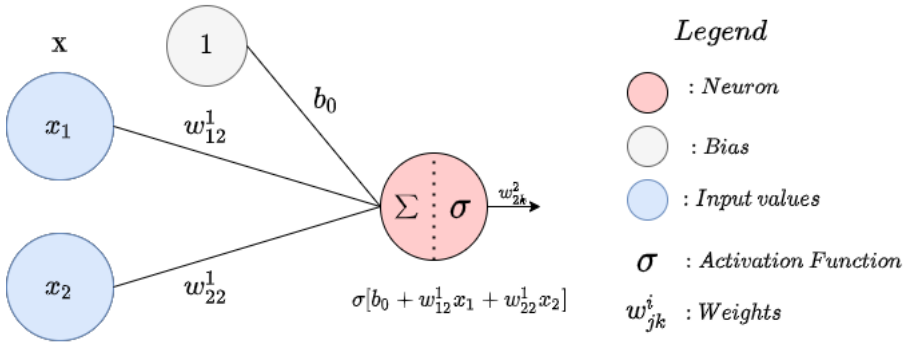

Figure 13: Representation of two connected neurons in the MLP. (Grey area in Figure 12)

The input observations, x_1, x_2, go through the synapses where each value is multiplied by its corresponding weight, w_{12}^1 and w_{22}^1. At the entry of the neuron, all values are summed together with a bias, b_0. This gives us the expression, $b_0 + w_{12}^1 x_1 + w_{22}^1 x_2$. Thereafter, this term is passed into a non-linear activation function chosen beforehand, which results in the output of the neuron and passed to the next layer. This same process is repeated over all the layers and all the neurons until the final output.

Computing the output of the Figure 12 with n features comes down to solving the following equation:

$$\mathbf{Y} = \phi\left[\mathbf{W}^{(2)}\sigma\left[\mathbf{W}^{(1)}\mathbf{X} + \mathbf{b^{(1)}}\right] + \mathbf{b}^{(2)}\right]$$

Where in this equation:

- $\mathbf{X}$ in $\mathbb{R}^{n\times 1}$ represents the matrix of input features with one occurrence per row and one feature per column.
- The weight matrix $\mathbf{W}^{(i)}$, also defined as the matrix of parameters, contains all the weight connections between neurons. As an example, $\mathbf{W}^{(1)}$ is a matrix in $\mathbb{R}^{q\times n}$ with q the number of neurons in the first hidden layer.
- The bias vector $\mathbf{b}$ contains the bias values with $\mathbf{b^{(1)}}$ in $\mathbb{R}^{q\times 1}$.
- σ and ϕ are the activation functions. Few examples are the Hyperbolic Tangent [Figure 30, Appendix B], the Rectified Linear Unit Function, the Softmax, etc.

8.1.2 Backpropagation

The weights corresponding to each connection between neurons are initialised randomly. In order for our MLP to produce the best forecast, we need to change the weights and adapt them to each case. This is called training the feed-forward network. Mathematically speaking, the aim is to minimise the error, defined by an error function between the real value and the forecast produced by the network.

$$\mathbf{W} = \arg\min_{\mathbf{W}} \frac{1}{N}\sum_{i=0}^{n} E_L(y_i, \hat{y}_i)$$

The error function is defined as $E : \mathbb{R}^2 \to \mathbb{R}$ with $E_L(y_i, \hat{y}_i)$ and y_i being the actual output and $\hat{y}_i$ being the forecasted value. It is important to emphasise that the loss function must admit a one order derivative.

In 1986, David Rumelhart, Geoffrey Hinton and Ronald Williams introduced the concept of *backpropagation* [21]. The idea is to minimize the error function, or loss function, of the network to find the best weights for each connection. It has to be stressed that minimising the loss function depending on thousands of parameters is close to impossible. Thus, we use the gradient of the loss function, defined as $\nabla E_L(\mathbf{W}) = \frac{\partial E_L}{\partial \omega_i}$, with ω_i as the $i-th$ weight parameter, to train our model. The gradient informs us about the behavior of the loss function, which indicates on how to change the weights to cause the most rapid decrease in the cost function. This implies only two passes in the network, one forward to compute the forecast and one backward to compute the gradient. The choice of the loss function (Mean Square Error, Mean Absolute Error, Cosine Similarity and others) is therefore capital to have a relevant model. The weights update is driven by Gradient Descent formula, which tweaks all the parameters with respect to the value of the gradient.

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha \frac{\partial E_L}{\partial \omega_i}\bigg|_{\mathbf{W}_t}$$

where the matrix of weights $\mathbf{W}_t$ is updated by subtracting the gradient balanced by the learning rate α, to obtain the updated weights $\mathbf{W}_{t-1}$. The learning rate is a hyperparameter chosen beforehand which controls the speed of the weights update. It is important to keep an eye on the learning rate, which can cause the loss function value to "zig-zag" around its minimum.
For to the gradient computation, we introduce the chain rule, which states that the derivative of a function can be built with the derivatives of the composing function. The error function is built from the input passed into a function then balanced by weights, which itself is passed to a function, etc. Therefore, to compute the gradient over the first weight on the first layer, $\frac{\partial E_L}{\partial \omega_{1,1}^{(1)}}$ can be rewritten as,

$$\frac{\partial E_L}{\partial \omega_{1,1}^{(1)}} = \frac{\partial E_L}{\partial Y}\frac{\partial Y}{\partial H_2}\frac{\partial H_2}{\partial H_1}\frac{\partial H_1}{\partial \omega_{1,1}^{(1)}}$$

where H_i is the output of the ith layer. Regarding the above expression, we can compute $\frac{\partial H_1}{\partial \omega_{1,1}^{(1)}}$ because we know the proper expression, $\mathbf{W}^{(1)}\mathbf{X} + \mathbf{b}^{(1)}$, with its activation function. Eventually, the computation becomes quite easy and we can build back the derivative until $\frac{\partial E_L}{\partial \omega_{1,1}^{(1)}}$. Recall the characteristic of the activation function, it must be differentiable, otherwise we would not be able to compute the derivative.

The backpropagation is the algorithm that regroups the notions explained above. It computes the loss over the outputs of the neural network. Thereafter, the gradient is determined for all the weights, $\frac{\partial E_L}{\partial \omega_i}$ to pass it to the gradient descent step. Each weight is tweaked to the right direction and the algorithm starts over. Each updated weight is used to compute new inputs with a new loss function which leads to new updates. In fine, the loss function converges to a minimum and the weights become stable. It is worth noting that the loss function can reach a local minimum which implies that the model is not at

the best of its performance. Finding the global minimum of a loss function by testing different weight initialisations is the scientist's work.

8.2 Recurrent Neural Networks

With the aim to process times series comes the *Recurrent Neural Network (RNN)*. A RNN is very similar to the MLP , except it also has a connection pointing backwards. Therefore, a recurrent neural network can simultaneously take a sequence of inputs and produce a sequence of outputs. This type of sequence-to-sequence network is useful for predicting time series such as stock prices.
At each time step, t, the recurrent neuron receives two inputs. Firstly, the input x_t of the initial observations sequence. Secondly, the weighted output from the preceding recurrent neuron, h_{t-1}. It is worth mentioning that there is no previous time step at time $t = 0$, therefore h_0 is generally set to 0.

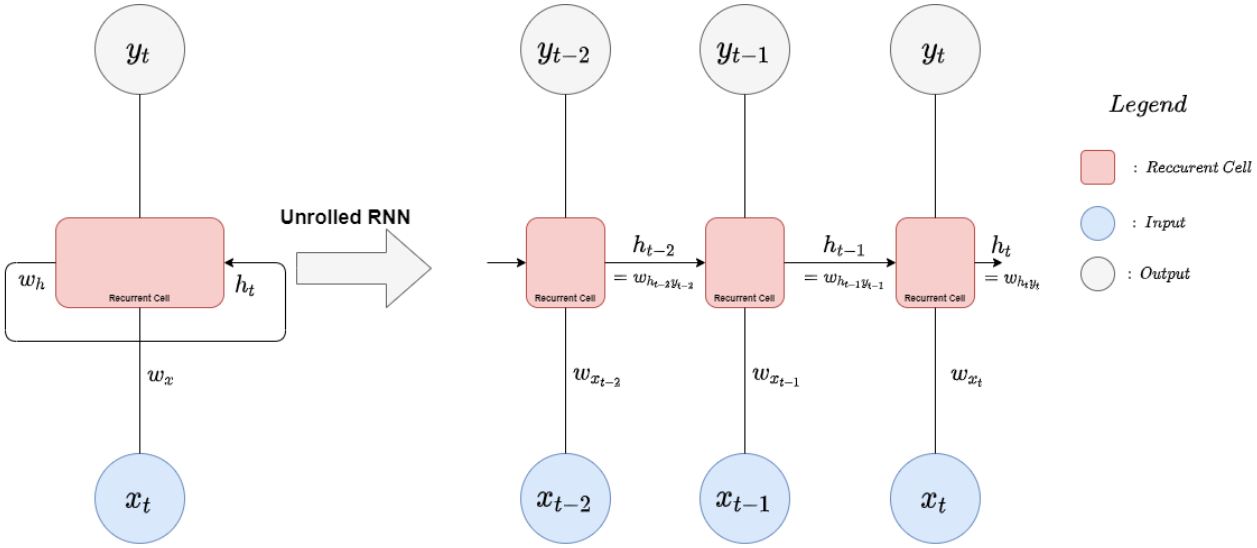

Figure 14: Representation a recurrent neural network with one layer, both rolled and unrolled

In this new architecture, each neuron has two sets of weights: one for the input x_t and the other for the outputs of the previous time step, h_{t-1}. We call these weights matrices, $\mathbf{W}_x$ and $\mathbf{W}_h$. The output of the whole recurrent layer, Figure 14, can be computed in the same manner as the MLP :

$$\begin{aligned}\mathbf{Y}_t &= \phi\left[\mathbf{W}_x\mathbf{X}_t + \mathbf{H}_{t-1} + \mathbf{b}\right] \\ &= \phi\left[\mathbf{W}\left[\mathbf{X}_t\mathbf{Y}_{t-1}\right] + \mathbf{b}\right] \quad with \quad \mathbf{w} = \begin{bmatrix}\mathbf{W}_x \\ \mathbf{W}_h\end{bmatrix}\end{aligned}$$

The value $\mathbf{H}_{t-1}$ is the state of the preceding neuron, equal to $\mathbf{W}_h\mathbf{Y}_{t-1}$. ϕ is the activation function and $\mathbf{b}$ is the bias added in the cell. We can contrast this to a memory that passes throughout the days to influence the value of the next output. The basic recurrent cell is able to learn very short time patterns. We can denote h_t as a function of some

inputs at time step t and its state at the previous time step: $h_t = f(h_{t-1}, x_t)$. Its output at time step t, denoted y_t, is also a function of the previous state and the current input.

We can think about the recurrent neural networks as generalisation of the VARMA process where the dependencies are no longer linear but non-linear throughout the use of the activation functions. The AR component is represented by the input observation feed into the network and the MA component is represented by the intermediate values that pass from recurrent cell to recurrent cell.

8.2.1 Vanishing Gradient Problem

The two main challenges faced by the traditional recurrent neural network are the *loss of information* and the *vanishing gradient*. As the hidden state of the cell passes from day to day, the weight correction $\mathbf{W}_h$ can easily reduce the influence of h_t on the result of the next time step. Therefore, if the value of a weight tends to zero, the information of the past days is reduced. We lose the information of the past days for the next outputs.

The second and most important challenge is the vanishing gradient problem. Gradients can often fade away over the layers. Suppose the gradient descent updates the weights of the first time step to decrease the output. Thereafter, the output of the second time step will decrease and so forth until the output is equal to zero. On a similar note, the gradient itself can also vanish or explode.
In such process, the neural network's weights are updated with respect to the partial derivative of the error function. If the value of the gradient is notably small, the update of the weights will be insignificant, forcefully preventing the network from training.

A solution to tackle these problems was introduced with the appearance of the Long-Short Term Memory cell [27].

8.3 Long Short-Term Memory

In 1997, Sepp Hochreiter and Jürgen Schmidhuber published a ground-breaking paper introducing the Long Short-Term Memory (LSTM) cell [31]. The work has been continuously improved over the years by several researchers, such as Alex Graves [29] who introduced the concept of Connectionist temporal classification LSTM, Haşim Sak [30] who worked on LSTM architecture for large vocabulary speech recognition and Wojciech Zaremba [65] who tackled the regularisation process.

The LSTM cell can be compared to a basic recurrent cell besides the fact that the state of the cell is divided into two different vectors being $\mathbf{h}_t$ and $\mathbf{c}_t$. The cell is also composed of three different gates that control the flow of information. The LSTM has proven to work very well for time series data as it detects longer time dependencies and allows for the information vector c_t to flow along the processing without being impacted. It is widely used in natural language processing, image and video analysis.

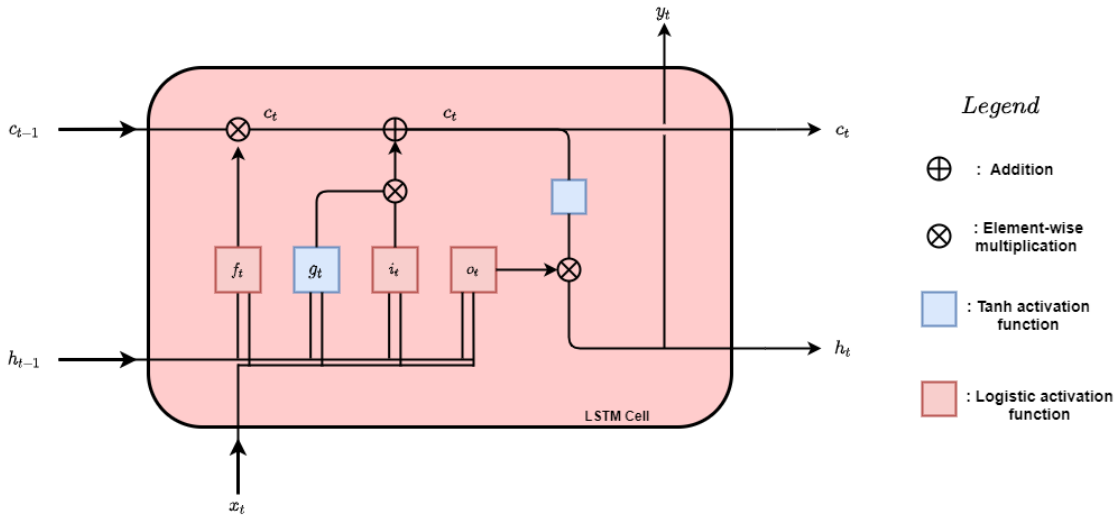

Figure 15: Representation of an LSTM cell.

The inputs of the LSTM cell are the initial observations x_t, the hidden state of the preceding time step, h_{t-1} and the information vector c_{t-1}. The inside of the LSTM is composed of three gates, specifically,

- The *forget gate* (noted as f_t)
- The *input gate* (noted as i_t)
- Finally, the *output gate* (noted as o_t)

With the flow of short-term information h_{t-1} comes the observations data, x_t. Both values are combined together and passed into the forget, input and output gates as well as an usual activation function "door" represented by g_t. The three gates use the same activation function which is the logistic function [Figure 29, Appendix B]. Therefore, the gates are open if the output is 1 and closed if the output is 0.
The long-term information vector c_{t-1} traverses the cell and is updated first by the forget gate where information is removed from the vector by multiplication between c_{t-1} and f_t. Next, the vector is updating by adding information selected from the input gate, i_t and g_t. After being updated by the gates, this vector of long-term information passes through the next time step without any more transformations. In parallel, the long-term information vector is also copied and passed through a *tanh* activation function [Figure 30, Appendix B] to be multiplied by the result of the output gate, o_t. Consequently, the outputs y_t and h_t are influenced by this vector of information c_t.
To update the information via the three gates, the cell uses either the addition or the element-wise multiplication, also called Hadamard Product. Finally, the outputs of the cell are the target estimation y_t, the two vectors of long-term information and short-term information, hence the name Long Short-Term Memory Neural Network.

Mathematically, the LSTM cell is described as below:

$$
\begin{aligned}
i_t &= \sigma(\mathbf{W}_{xi}\mathbf{X}_t + \mathbf{W}_{hi}\mathbf{H}_{t-1} + \mathbf{b}_i) \\
f_t &= \sigma(\mathbf{W}_{xf}\mathbf{X}_t + \mathbf{W}_{hf}\mathbf{H}_{t-1} + \mathbf{b}_f) \\
o_t &= \sigma(\mathbf{W}_{xo}\mathbf{X}_t + \mathbf{W}_{ho}\mathbf{H}_{t-1} + \mathbf{b}_o) \\
g_t &= tanh(\mathbf{W}_{xg}\mathbf{X}_t + \mathbf{W}_{hg}\mathbf{H}_{t-1} + \mathbf{b}_g) \\
c_t &= f_t \otimes c_{t-1} + i_t \otimes g_t \\
y_t &= h_t = o_t \otimes tanh(c_t)
\end{aligned}
$$

where W_{xi}, W_{xf}, W_{xo} and W_{xg} are the weight matrices of each of the three control gates, and the door g_t, for their connection to the input vector x_t. W_{hi}, W_{hf}, W_{ho} and W_{hg} are the four weight matrices of each of the three control gates as well as g_t for their connection to the previous short-term state h_{t-1}. And finally, b_i, b_f, b_o and b_g are the bias terms of each gate. It is worth mentioning that in practice, we initialise b_f to a vector of 1s instead of 0s. This prevents forgetting everything at the beginning of the training.

8.4 Autoencoders

The last architecture needed for our analysis is the Autoencoder (AE) network. According to the history provided in Schmidhuber, "Deep learning in neural networks: an overview", autoencoders were proposed as a method for unsupervised pre-training in "Modular learning in neural networks" (1987).
The autoencoder is a type of neural network that is trained to reproduce its input. Although the task seems to be quite useless, there is more to that. The purpose of this architecture is to learn to represent the initial observations in an efficient approach. Among others, autoencoders are used for dimensionality reduction and feature learning. It is a symmetrical neural network used to learn representation of an input sequence into an intermediate vector, called *latent vector* or *codings*. The symmetry appears with the encoders which is built from the layer of input until the layer of codings and the decoder which is the symmetry of the encoder over the axe of latent codings, until the reconstruction.

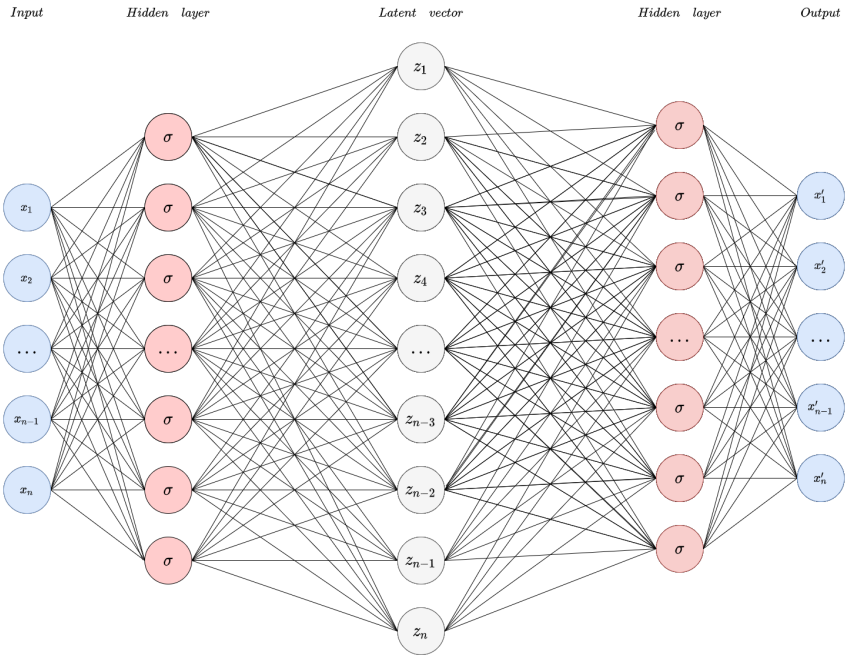

Figure 16: Representation of an autoencoder

The latent vector, $\mathbf{Z}$, is built up with the vector of inputs. Mathematically, we can describe the autoencoder as a function that maps the inputs $\mathbf{X}$ to $\mathbf{Z}$, then a function that maps $\mathbf{Z}$ to $\mathbf{X}$.

$$\mathcal{F} : \mathbf{X} \to \mathbf{Z}$$
$$\mathcal{F}' : \mathbf{Z} \to \mathbf{X}$$

Therefore, the autoencoder can be trained to minimize the difference between the input and the reconstruction,

$$E_L(\mathbf{X}, \hat{\mathbf{X}}) = |\mathbf{X} - \hat{\mathbf{X}}|^2$$

In our work, the dimension of the codings is higher than the size of the initial vector. We are talking about overcomplete autoencoders [34]. The purpose of having a higher dimension introduces the concept of *feature extraction*. Feature extraction is the process of "combining existing features to produce more useful one" [7]. By representing the initial vector into a vector of higher dimensions, the codings can capture the complex dynamics in the times series which is useful for future analysis.

[7] Aurélien Géron, "Hands-On Machine Learning with Scikit-Learn, Keras Tensorflow", O'Reilly, 2019, p.27

9 Our model: AE-Stacked LSTM

Let's recall the definition of returns in the framework of our analysis,

$$X_t = \mu_t + \epsilon_t, \quad t \in \mathbb{Z}$$
$$where \quad \epsilon_t \sim WN(0, \sigma_t), \quad \forall t \in \mathbb{Z}$$
$$with\ \mu_t = f_1(\mathcal{F}_{t-1}, \mathbf{W})$$
$$with\ \sigma_t = f_2(\mathcal{F}_{t-1}, \mathbf{W})$$

where $f(\mathcal{F}_{t-1}, \mathbf{W})$ is a function used to estimate μ_t or σ_t with respect to the information known at time $t-1$ and the parameters of the neural network.

With the aim of modelling the mean and volatility of returns with more precision than the simple vanilla LSTM model from Zhichao Du, Menghan Wang Zhewen Xu, we develop a slightly more complex architecture. Their paper [68] describes the study of the VaR with a recurrent neural network built with three hidden layers. They conclude by proposing evolution to their model by increasing the depth of the NN to improve the accuracy. Increasing the number of layers allows the model to extend the encoding of the times series' characteristics over more layers which captures better the times series' dynamics. Related to deepening the network, we can also increase the width. This means increasing the number of neurons in each layer. Both methods should, to a certain extend, improve the feature extraction and de facto the accuracy of the network. However, it also increases the chances of overfitting the data.
Based on the idea of improving the feature extraction of the inputs, we decide to not only increase the size of the network, but more than that, introduce a specific feature extraction block in our network. This is a choice made to avoid overfitting by having an absurdly deep network and explore novel methods for forecasting instead of a simple deeper network.

The architecture has proven its effectiveness in the paper *Time-series Extreme Event Forecasting with Neural Networks at Uber* [53] by Nikolay Laptev, Jason Yosinski, Li Erran Li & Slawek Smyl. Their model was developed in order to accurately forecast the extreme values of demand during periods of high variance, such as the holidays, sporting events, etc. For this manner, they proposed a Long Short-Term Memory based neural network composed of two main blocks.

- An Autoencoder which is used for the feature extraction.
- A LSTM forecaster used to forecast the demand based on the autoencoder's latent vector.

The architecture used in our work is quite similar to the model proposed by Nikolay Laptev, Jason Yosinski, Li Erran Li & Slawek Smyl. In the first place, an autoencoder is used for the purpose of auto-feature extraction on the S&P500. We call this first block, the **LSTM Autoencoder**. The vector of n *input days* of the S&P500 is passed throughout the autoencoder. The model trains to reproduce the original vector of n days and thus creating a latent vector of higher dimensions in the process which should capture complex

dynamics in the time series. We chose to encode the vector of input of the S&P500 alone and not take into account the exogenous variables. This choice is rather arbitrary. We want to avoid the curse of dimensionality by re-encoding the vector of n input days for the 4 variables into an even higher dimension vector. The curse of dimensionality refers to the surprising phenomena that appears when processing data in high dimensions. This can, among others, cause many problems when training a network.

Thereafter, we retrieve the latent vector produced by the encoder to feed it to the forecaster. We call this second block, the **Stacked LSTM**. Before feeding the latent vector, it is necessary to concatenate the latent vector representing the S&P500 with the input vector of exogenous variables (n input days of Nikkei225, FTSE100 and Dow Jones).
The Stacked LSTM network is made of an arbitrary number of layers and outputs the desired values over a specific time horizon. Intuitively, the architecture can be thought as a simple vanilla LSTM network which takes, as input, a higher dimension vector of the S&P500 plus the exogenous variables. It is worth noting that the neural networks do not require any hypothesis on the data. For the sake of our work, we need to process the returns. However there are no prerequisites over the type of data we have to process.

We call the combination of both blocks, the **Autoencoder- Stacked Long Short-Term Memory Neural Network (AE-Stacked LSTM)** model. A detail schema is displayed below.

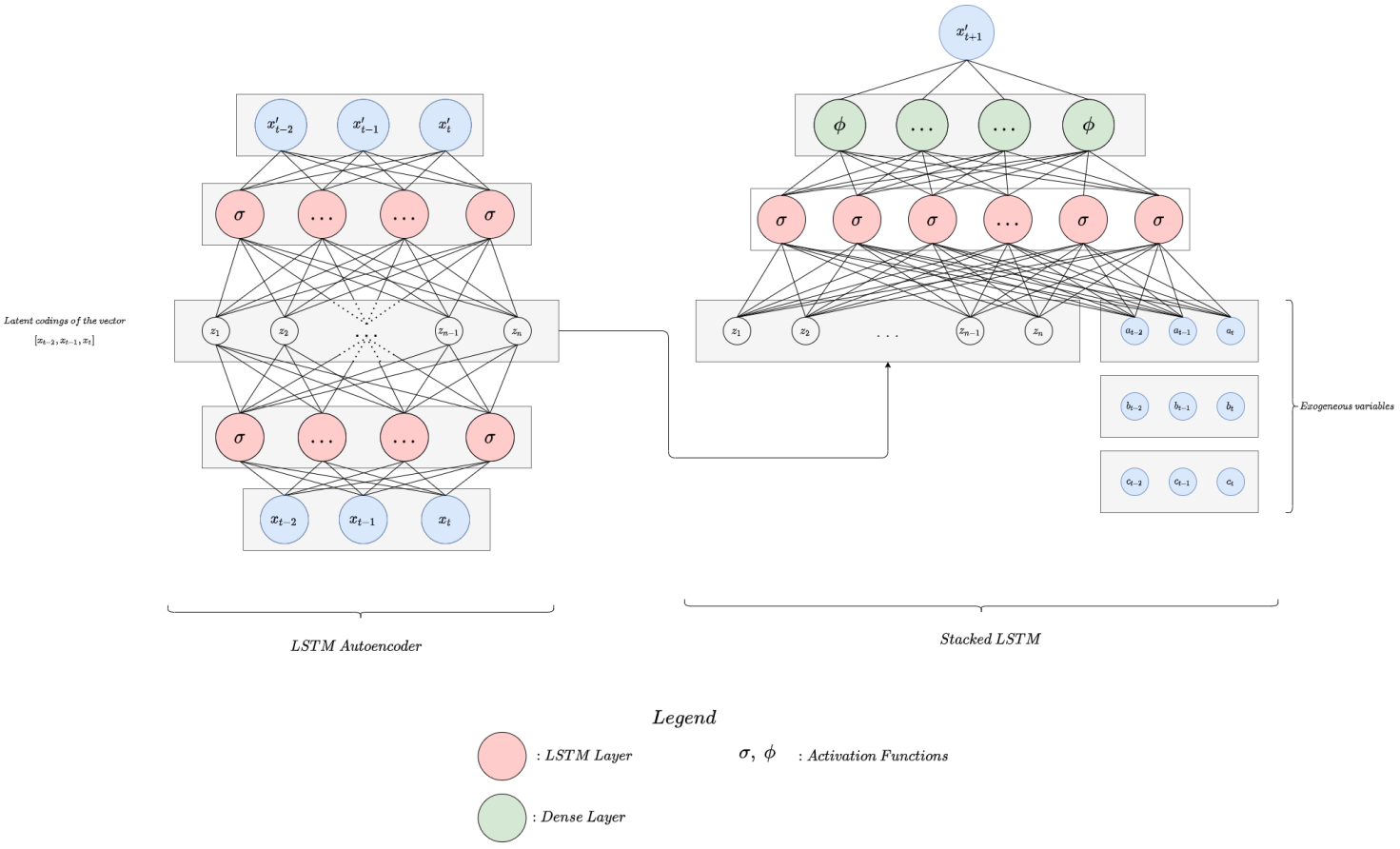

Figure 17: Representation of our AE-Stacked LSTM

As shown in the schema above, each block is built with multiple *layers*. First of all, each block starts with an *input layer*. It is a mandatory layer that takes as input the preprocessed data. In the specific case of the LSTM model, the input is a tensor, also thought as a 3D matrix build with the batch size, the time step and the number of features. The *batch size* is referred to the number of observations used for one iteration in the training step. The *time step* is the number of days taken into account to forecast the values of the next days. And finally, the *number of features* is the number of times series to introduce in the model. In other words, it is the number of variables in the network. Second, we have the *Long Short-Term Memory layers* explained earlier in the work. It is characterized by the *number of neurons* as well as a *dropout rate*. Defined as a regularisation method in the paper *"Dropout: A Simple Way to Prevent Neural Networks from Overfitting"*, the dropout rate corresponds to a probabilistic value between 0 and 1 where neurons are probabilistically excluded while training a network. At every iteration of the training, each neuron has a probability to be included or not in the computation. After the training, the neurons are not supposed to be dropped for forecasting, unless otherwise specified. This dropout method can be compared to a regularised network where regularisation helps to generalize best and avoid overfitting of the model. As stated earlier, we develop a deeper NN with a feature extraction block. A deeper and wider ANN favours overfitting, hence regularization is an important part of the analysis.

Consider a single linear unit layer with one output. We present the usual mean square error of the dropout network ($E_{Dropout}$) as well as the mean square error of all the sub-network possible from the original single input layer (E_{SubNet}) [10],

$$E_{SubNet} = \frac{1}{2}(y - \sum_{i=1}^{N} p_i \omega_i x_i)^2$$

$$E_{Dropout} = \frac{1}{2}(y - \sum_{i=1}^{N} \delta_i \omega_i x_i)^2$$

$$with\ \delta_i \sim Bernoulli(p)$$

where y is the target value, ω_i is the ith weight, x_i the observation and p_i a probability of the weights appearing. The ensemble average of the sub-networks can be computed using $p_i\omega_i$ instead of ω_i. Therefore, when deriving the loss function in the framework of minimisation, we obtain,

$$E_{SubNet} = \frac{1}{2}(y^2 - 2y \sum_{i=1}^{N} p_i \omega_i x_i + (\sum_{i=1}^{N} p_i \omega_i x_i)^2)$$

$$\frac{\partial E_{SubNet}}{\partial \omega_i} = -y p_i x_i + \omega_i p_i^2 x_i^2 + \sum_{j=1, j\neq i}^{N} \omega_j p_i p_j x_i x_j$$

$$E_{Dropout} = \frac{1}{2}(y^2 - 2y\sum_{i=1}^{N}\delta_i\omega_i x_i + (\sum_{i=1}^{N}\delta_i\omega_i x_i)^2)$$

$$\frac{\partial E_{Dropout}}{\partial \omega_i} = -y\delta_i x_i + \omega_i\delta_i^2 x_i^2 + \sum_{j=1,j\neq i}^{N}\omega_j\delta_i\delta_j x_i x_j$$

Knowing that the random variable δ follows a Bernoulli distribution with mean p and variance $p(1-p)$, we compute the expected value of the derived expression above, as follows,

$$\begin{aligned} E[\frac{\partial E_{Dropout}}{\partial \omega_i}] &= -yp_i x_i + \omega_i p_i^2 x_i^2 + \omega_i Var(\delta_i)x_i^2 + \sum_{j=1,j\neq i}^{N}\omega_j p_i p_j x_i x_j \\ &= \frac{\partial E_{SubNet}}{\partial \omega_i} + \omega_i Var(\delta_i)x_i^2 \\ &= \frac{\partial E_{SubNet}}{\partial \omega_i} + \omega_i p_i(1-p_i)x_i^2 \end{aligned}$$

The expectation of the gradient with dropout is equal to the derived expression of the classical neural network plus the regularization parameter induced by the dropout rate. This means that minimizing the neural network's loss function with dropout is equivalent to minimizing the classical neural network's loss function with a regularization term.

$$E_{Dropout} = E_{NN} + \frac{1}{2}\sum_{i=1}^{n}\omega_i^2 Var(\delta_i)x_i^2$$

Note that the maximum regularization possible induces a dropout rate of 0.5 since $p(1-p)$ is maximum with p=0.5.

Third, we have the *Dense layer*. It must be the last layer of the Stacked-LSTM block. It represents the simplest layer to use in a neural network with a classic cell such as described in section 8.1.1. In our case, this layer is used to change the dimension of the vector into the output size $1 \times p$ with p the forecast horizon. The parameters that characterise these layers are mainly the activation function and the layer's number of neurons.

The last hyper-parameter of our models is the number of *epochs*. Also considered as the number of times the model adapts its parameters. In other words, the number of times the Gradient Descent will be called to adapt the weights in the network.

With the aim to clarify the architecture of the model as well as all the variables taken into account in our estimation, we provide next page, Table 7, a representation of all the hyper-parameters of both the AE-Stacked LSTM for the mean and for the volatility estimations.

Hyper-Parameters
Input Days
Target Width
Latent Vector Size
Hidden Layers
Dropout Values
Optimizer
Loss Function
Number of Epochs
Number of Neurons

Table 7: Presentation of the hyper-parameters of the model

9.1 Model Results

The analysis were made on a Windows operating system equipped with a processor Intel Core i7, 2.60 GHz and 8Go of RAM. The code is written in Python using the package Tensorflow.

9.1.1 Preprocessing of the data

The first task in our analysis is the preprocessing of the four financial time series. The neural networks and specifically the LSTM are models that are more efficient when using scaled data [27]. Therefore, we apply a standardisation to our matrix of observations. The standardisation scales the data by subtracting the empirical mean from the observations to set it to zero. Then, it divides the result by the standard deviation of the feature to obtain a unit standard deviation on the scaled data. Mathematically, the standardization is represented as follows:

$$z_t = \frac{x_t - \mu}{\sigma}, \quad t \in \mathbb{Z}$$

where z_t and x_t are respectively the new value and the old value of the return at time t. μ and σ are the mean and standard deviation of the sample data. We made the choice to use the standardisation process not to bound the values in a specific chosen range. In fact, there exists other scaling methods such as the so-called Min-Max which bounds the scaled values into a range, often from 0 to 1. As the results were not as significant, we made the choice not to use it.

Once the data is scaled, we proceed to the division of our dataset into three parts: *training set*, *validation set* and *test set*. This allows us to evaluate our model correctly, using the training set for the only purpose of training the model. The validation set is used to control for performance, overfitting and calibration of the hyper-parameters. With the epoch corresponds a value of the loss. As the training loss can only decrease along the number of epochs, the validation loss value should also decrease. However, when we get to the overfitting step, meaning that the network fits the noise of our training data, the loss values of the validation set increases back. Therefore, we know at which epoch

to stop our training. Finally, the test set is used to evaluate the error of our model over data that was used neither in the training nor the validation step. The initial dataset was divided into 70% for the training set, 20% for the validation set and 10% for the test set.

As mentioned in Section 8.1.2, the choice of the loss function is capital for the good realisation of the specific task. The first logical decision would be to use the log-likelihood function such as with the econometrics models where the error term, ϵ follows a Gaussian distribution with mean 0 and variance σ_ϵ. The likelihood function with θ as a matrix of parameters can be written as follows.

$$L(\mathbf{W}) = \prod_{t=1}^{T} \frac{1}{\sigma_\epsilon} exp(-\frac{1}{2}(\frac{z_t - \hat{z}_t}{\sigma_\epsilon})^2)$$

with $\hat{z}_t = f_1(\mathcal{F}_{t-1}, \mathbf{W})$. The log-likelihood of the observation z_t is therefore of the form

$$log(L(z_t|\mathbf{W}, x_{t-1})) = -\frac{1}{2\sigma_\epsilon^2}(z_t - \hat{z}_t)^2 + c$$

with $c \in \mathbb{R}$ as a constant. Maximising the log-likelihood is comparable to minimising the mean square error between the estimated and the observed values. Therefore, we use the Mean Squared Error (MSE) function as the cost function for our model of the mean or returns.

Thus the model is optimised on the mean squared error loss function, using the *Adam* algorithm (Adaptive moment estimation). The Adam algorithm was introduced by Kingma and Ba [22] with the purpose to optimise the gradient descent algorithm developed in section 8.1.2. The main idea is to adapt the learning rate, α, using the first and second moment of the gradient.

$$\mathbf{v}_t = \beta_1 \mathbf{v}_{t-1} - (1 - \beta_1)\nabla E_L(\mathbf{W})$$

$$\mathbf{s}_t = \beta_2 \mathbf{s}_{t-1} + (1 - \beta_2)(\nabla E_L(\mathbf{W}))^2$$

where $\mathbf{v}$ and $\mathbf{s}$ are two vectors build over the first and second moments of the gradient. Thereafter, the vectors are scaled such as,

$$\hat{\mathbf{m}} = \frac{\mathbf{m}}{1 - \beta_1^t}, \quad \hat{\mathbf{s}} = \frac{\mathbf{s}}{1 - \beta_2^t}$$

and the backpropagation algorithm can be updated as,

$$\mathbf{W_t} = \mathbf{W_{t-1}} + \frac{\eta \hat{\mathbf{m}}}{\sqrt{\hat{\mathbf{s}} + \epsilon}}$$

where ϵ is used to avoid a division by zero and η is the learning rate parameter. The two vectors $\mathbf{v}_t$ and $\mathbf{s}_t$ are the expression using the first and second moments of the gradient. Therefore, using the Adam algorithm optimize the convergence speed and quality.

To tune our hyper-parameters, we choose to process to a 10-fold cross validation. The concept of the cross validation is to divide the data set into k different folds. It allows us to train our model on $k-1$ folds and evaluate our performance on the last fold. By doing this process on all the combination of $k-1$ fold, we can average our measures of performances and adapt our hyper-parameters to the best performance. This method also allows us to detect for overfitting and optimize our hyper-parameters. Knowing the high number of hyper-parameters in our models, we tested the input width in a range from 1 to 100 with steps of 5 days. The number of layers was tested going from 1 to 5 layers with a number of units from 2 to 50 by steps of 2. The dropout rate was tested from 0.1 to 0.5 and the latent vector size from size 2 to 70 units.

9.1.2 AE-LSTM Model for the Mean

The first task, such as with our baseline model, is to estimate the conditional mean of the S&P500. It is important to state that the neural networks process tensor with shape [batch size, input with, features]. However, to ease the explanation, we talk about vector or matrices in the model, therefore not taking the batch size into account.

First, we describe the one-day ahead model which is different from the 10-days ahead neural network. We use a vector of inputs containing the 45 preceding days to forecast the mean of returns in t. The LSTM-Autoencoder, Table 15 Appendix B, is built with 3 hidden layers composed of 2, 3 being the latent codings and 2 units. Therefore, the latent codings is of size 45×3. This can be thought as 45 days of one feature re-described into 3 features of 45 days.
It is important to understand the difference between units and cells in a LSTM neural network. Recall the LSTM cell defined in section 8.2, a LSTM neural network is built of units organised in cells. A matrix of input $n \times p$ is modelled by a layer of p units formed of n cells. In our specific case, the layer of codings is built with 3 units to create a matrix of size 45×3. In order to better understand the difference between units and cells, the figure 39 in Appendix B shows both a 2D and a 3D representation of a 2 units LSTM network that takes as input 2 series of 45 days to forecast one value in t.

We train a grand total of 155 parameters over the mean squared error loss function with a dropout of 0.3. The graph of loss values, Figure 31, Appendix B, for both the training and validation set shows decreasing MSE along the epochs. This means that the model trained well and increased the accuracy of the estimations over time. There is no sign of overfitting but the loss value stagnates from epoch 60. Hence, we can stop the training around this epoch.

The latent codings (45x3) combined with the exogenous variables(45x3) produces an input matrix of size 45 by 6 feed into the Stacked-LSTM. It is built with 3 hidden layers with respectively 15, 10 and 5 units. Figure 32, Appendix B shows a validation set that does not decrease with the number of epochs. This might raise questions on the performance of the model. As the network trains, the accuracy over the validation set is not improving. However, the loss becomes less stable from epoch 30. Therefore, we stop the training around the epoch 30. The dropout rate of the LSTM layers is chosen to be 0.5. A description of the Stacked-LSTM model is displayed in Table 16 Appendix B. The results showing the conditional mean for a one-day ahead framework is represented below.

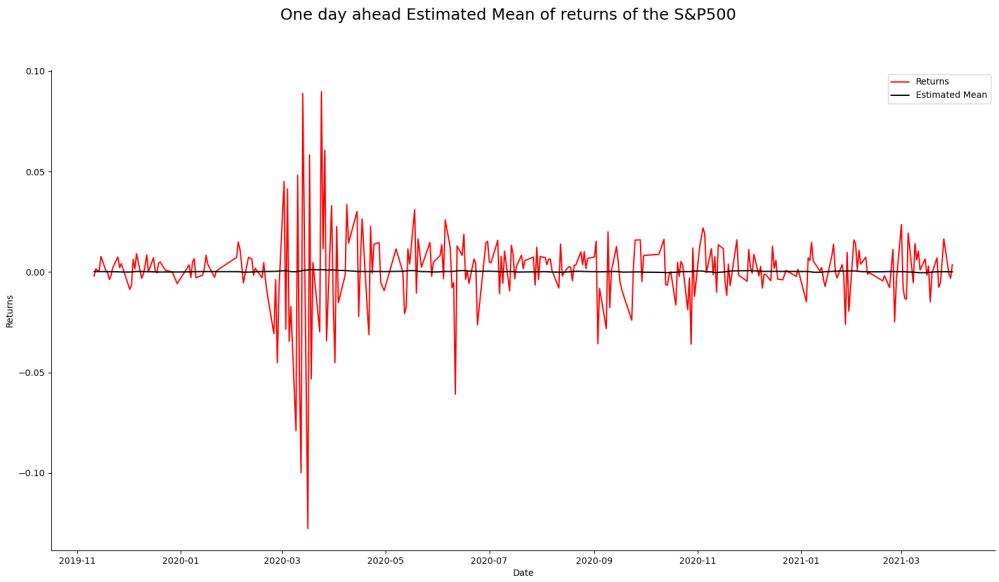

Figure 18: One-day ahead mean estimation of the S&P500 returns with the AE-Stacked LSTM model

The estimated conditional mean is different from the benchmark model's estimations. In fact, the period of Covid-19 which was very distinct with the VARMA(1, 1) goes unnoticed. The mean is almost constant along the 251 days of forecasting with a slight dent in March 2020. The network seems to approximate the unconditional mean over the whole period of testing. The value of the SSE is equal to 0.11342. We can conclude that the estimation of the conditional mean is worse than with the VARMA model (0.10580). However, we do not conclude yet that the AE-Stacked LSTM is worse than the benchmark model.

For the 10-days ahead forecasting, the LSTM-Autoencoder, Table 19 Appendix B, is composed of 5 hidden layers with 3, 5, 7, 5 and 3 units. The matrix of latent encoding is built over 45 days by 7 columns. We train a total of 976 parameters. The loss function, Figure 35 Appendix B, shows a validation loss decreasing along with the training loss after 25 epochs. Therefore the training benefited both sets. The dropout value chosen is equal to 0.3.
The Stacked-LSTM, Table 20 Appendix B, is created by 4 hidden layers to process the data and return the mean vector of the S&P500 over 10 days. The loss function, Figure 36 Appendix B, shows a validation loss that increases from epoch 10. This means that, as the model trains the performance does not increase but rather decreases. No matter the hyper-parameters, we did not manage to obtain a decreasing validation loss. The conditional mean of returns for the 10-days ahead forecast is displayed on Figure 19, next page.

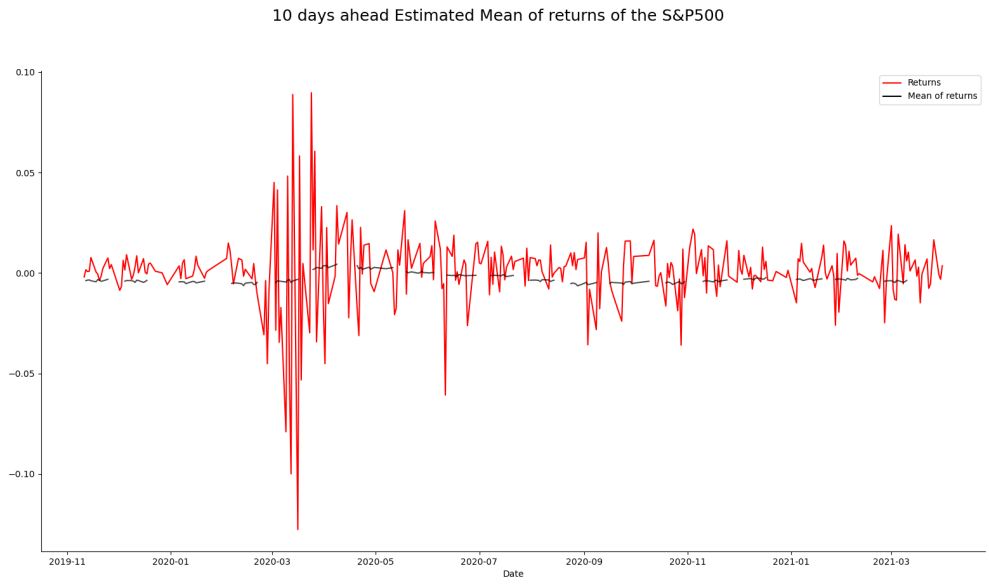

Figure 19: 10-days ahead mean estimation of the S&P500 returns with the AE-Stacked LSTM model

For each set of 10-days, the estimated mean is overall constant over the time step. This differs from the benchmark model with a higher variance for the first estimated values in the time step. The conditional mean before the Covid-19 period is forecasted being negative with a slight increase in April 2020. The Covid-19 period is definitely captured by the model, in contrast to the one-day ahead model. After the crisis the conditional mean comes back to its initial value predicted before the crisis. The value of the SSE is equal to 1.18632 which confirms a worse performance than the VARMA model (0.10580).

Such as with our econometric model, we retrieve the residuals from our estimated conditional mean to compute the standard deviation.

9.1.3 AE-LSTM Model for the Volatility

Recall the definition of the returns,

$$X_t = \mu_t + \epsilon_t, \quad t \in \mathbb{Z}$$

$$where \quad \epsilon_t \sim WN(0, \sigma_t), \quad \forall t \in \mathbb{Z}$$

$$with\ \mu_t = f_1(\mathcal{F}_{t-1}, \mathbf{W})$$

$$with\ \sigma_t = f_2(\mathcal{F}_{t-1}, \mathbf{W})$$

The second part of the analysis is to represent the volatility of the S&P500 residuals, σ_t. However, the way of doing things is quite different from the first benchmark model. In fact, we represent the volatility of the residuals using the standard deviation over a window of n residuals value. This helps the network to learn patterns easier as we reduce the noise in the series. We choose a sliding window of 3 residuals to slightly smooth the volatility but keep as much the extreme values intact.

$$\sigma_t = \sqrt{\frac{\sum_{i=t-3}^{t}(\epsilon_t - \mu_\epsilon)^2}{n-1}}, \quad t \in \mathbb{Z}$$

where ϵ_t is the value of the residual at time t, μ_ϵ is the value of the mean of the residuals and n is the number of data in the set. This is one type of computation among others. As stated in section 5.1.1, the main drawback of this method is the loss of information over the days. We can compare it to "chasing a running train". In fact, the sudden high peaks of volatility in t are smoothed by values of $t-2$ until $t-1$. Hence, we do not capture the actual volatility in t.

We obtain a new time series defined as the volatility. Feeding this series to our model, we try to forecast the volatility in $t+1$ using the series of volatility itself, as well as the other exogenous variables. Note that this analysis is supervised, as we feed the volatility we want to estimate.

The LSTM autoencoder, Table 17, Appendix B, is composed of 3 hidden layers with a latent matrix of size 45×5. The autoencoder takes, as input, 45 days of volatility of the S&P500 such as for the mean estimation. The loss function of the autoencoder, Figure 33, Appendix B, shows a validation loss that is decreasing along the training loss. We can stop the training around epoch 40 as the weights converged and the loss is not improving. We train a total of 332 parameters of which zero are non trainable.
The Stacked-LSTM, Table 18 Appendix B, is composed of 2 hidden layers, with respectively 20 and 10 units. It takes as input a matrix of 45 days by 8 features (5 for the S&P500 and 3 for the exogenous series). The model is composed of 3.571 parameters to train over the mean square error loss function. Figure 34 shows a decrease in the validation loss until epoch 50 where the MSE becomes somewhat stable.

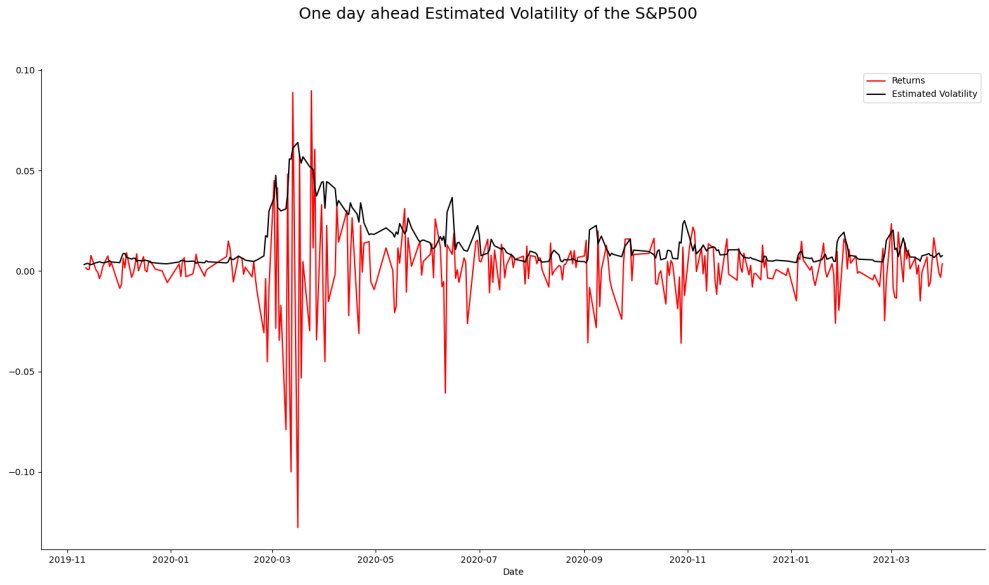

Figure 20: One-day ahead volatility estimation of the S&P500 returns with the AE-Stacked LSTM model

The volatility is well represented in general as it increases in periods of high volatility and decreased in periods of low volatility. The highest volatility forecasted is in period of Covid-19 where the model adapts its estimations very quickly as we can observe with the beginning of the crisis in March 2020. The estimation with the neural network seems more precise than the benchmark model as we can distinguish multiple peaks of volatility in March, June, September and November 2020 as well as February 2021. In the DCC-GARCH model, we could only highlight a distinct peak in the Covid period and in June 2020. As the volatility adapts very quickly to the periods of crisis and it also decreases quickly compared to the benchmark model. We can also notice that the statistical model's estimations are smoother than the neural network's estimations. The value of the NN's SSE is equal to 0.0402. Hence we confirm that the performance on the volatility estimation with the neural network is better than the DCC-GARCH model (0.1077).

Concerning the 10-days ahead forecast of the volatility, the LSTM-Autoencoder, Table 21 Appendix B, is composed of 5 hidden layers with a latent matrix of size 55×6. The 55 days of S&P500's volatility are represented into a matrix of 55 days and 6 features. The loss function, Figure 37 Appendix B, shows a validation loss that decreases with the number of epoch. We stop the training at epoch 70 to retrieve the latent codings and feed it to the Stacked-LSTM, Table 22 Appendix B. Build with 3 hidden layers, the Stacked-LSTM model shows a validation loss function that stagnates around 3.25, Figure 38 Appendix B. Again, such as with the mean model, we find it difficult to train the model and improve result for the validation loss. The model does not learn any patterns and the weights do not converge through a better performance.

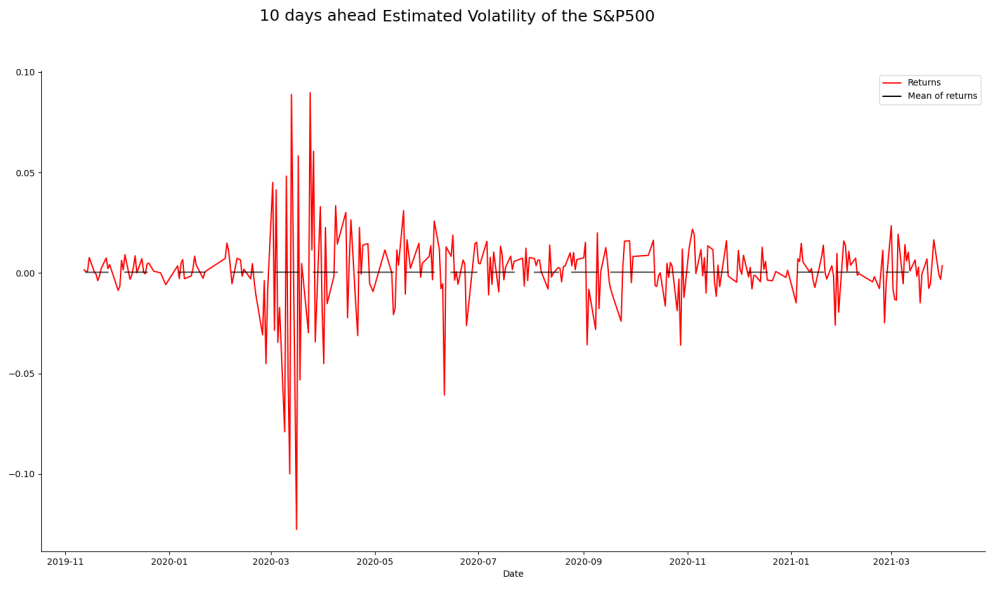

Figure 21: 10-days ahead volatility estimation of the S&P500 returns with the AE-Stacked LSTM model

The values, although seemingly equal to zero, are actually positive. We have similarities between the AE-Stacked LSTM and the benchmark model results. However, the DCC-GARCH showed a slight increase in volatility in the Covid-19 period. In the results

above, we do not capture any changes in volatility. We do not have conclusive results for the standard deviation estimations. The values are close to being constant over the whole test set. The SSE is equal to 64062.835 which is far from being better than the DCC-GARCH (328.583).

9.1.4 Conclusion

The results obtained with the AE-Stacked LSTM are coherent. However, the mean models as well as the 10-days ahead volatility model, based on the SSE performs worse than the benchmark model. The one-day ahead volatility model, on the other hand, performs better than the benchmark and seems to capture more volatility patterns.

SSE on Test set		
Mean	SSE -One day ahead	SSE - 10 days ahead
VARMA- DCC-GARCH	**0.10580**	**1.11666**
AE-Stacked LSTM	0.11342	1.18632
Volatility		
VARMA- DCC-GARCH	0.10773	**328.583**
AE-Stacked LSTM	**0.04021**	64062.835

Table 8: Representation of the Sum of Squared Errors for all the models

Based on the results, we can state the hypothesis that the Value at Risk will perform worse than the benchmark model, especially for the 10-days ahead model.

10 Simulations for the Value at Risk

The last part of our work is the Monte Carlo simulation over our forecasts. Recall one last time the definition of the returns,

$$X_t = \mu_t + \epsilon_t, \quad t \in \mathbb{Z}$$

$$where \quad \epsilon_t \sim WN(0, \sigma_t), \quad \forall t \in \mathbb{Z}$$

which can be rewritten as,

$$X_t = \mu_t + \sigma_t Z_t, \quad t \in \mathbb{Z}$$

$$where \quad Z_t \sim SWN(0, 1), \quad \forall t \in \mathbb{Z}$$

We defined the conditional mean of returns, μ_t and the conditional volatility, σ_t of the S&P500 both using an econometric model and a deep learning model. Therefore, we have all the parameters needed to process to the Monte Carlo simulations. In our case we simulate 10.000 scenarios of the S&P500. The objective is to create an arbitrary number of financial scenarios which are by definition all different but based on the same mean and same volatility. This is completed by summing the conditional mean with a strict white noise process multiplied by the conditional volatility.

As an example, we display a set of simulations for the 10 days-ahead forecast with the VARMA- DCC-GARCH model.

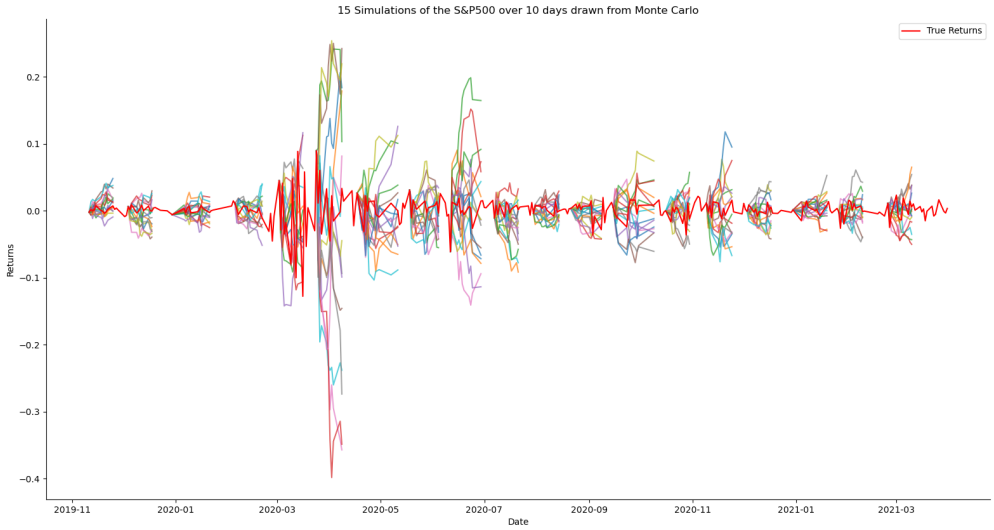

Figure 22: Simulations of the 10 days-ahead scenarios using VARMA- DCC-GARCH

As shown on the 10-days ahead example above, the simulations have their volatility increasing drastically over the days. This is due to the accumulation of random values, Z_t, over the days. While for the one-day ahead, Figure 40 Appendix B, the extreme values are not as high because we forecast over a one-day horizon. Hence, the random values do not stack up over the days. The simulations for the AE-Stacked LSTM are presented in Figure 41 and 42, Appendix B.

10.1 Value at Risk and Expected Shortfall

The final goal of our analysis is to compare the performance of both models on the computation of the VaR. The retrieve the VaR_α, we compute the α quantile of the simulations at time t. This returns a maximum value of return l for which the probability of having a loss higher than l is $1 - \alpha$. In the framework of this master thesis, we compute the 95% VaR. Therefore, we define the loss-threshold at which there is a 5% chance to go over. The 95% VaR and the ES for the one-day ahead computed using both the VARMA-DCC-GARCH and the AE-Stacked LSTM model are displayed on Figure 23 and 24, next page.

The VaR and the ES for the one-day ahead estimation using the VARMA- DCC-GARCH is displayed below.

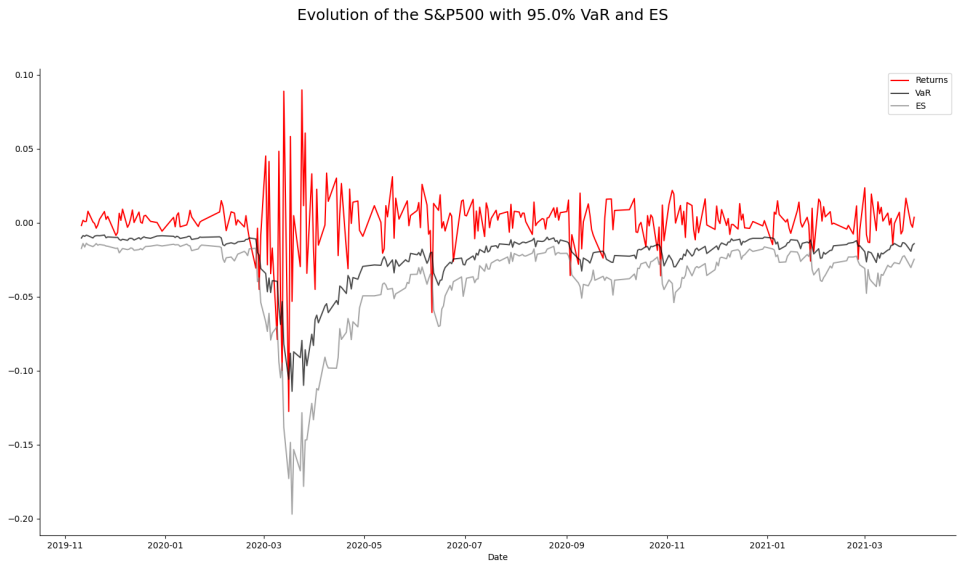

Figure 23: Value at Risk for the VARMA- DCC-GARCH for the one day ahead forecast

While for the neural network model, we obtain the following results.

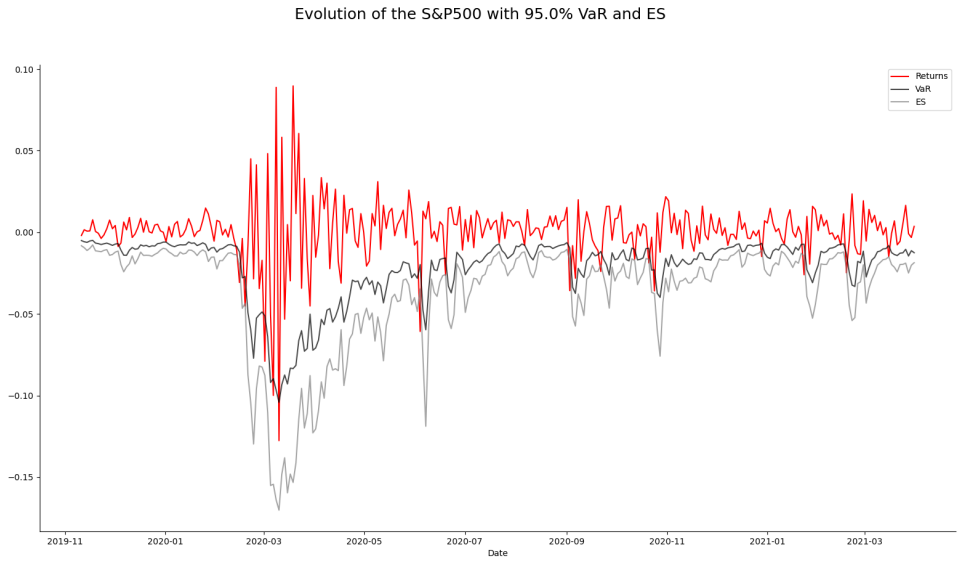

Figure 24: Value at Risk for the AE-Stacked LSTM for the one day ahead forecast

Comparing both graphs, we distinguish a very noticeable characteristic, the VaR of the benchmark model is smoother than the neural network model. This is due to the very sharped volatility estimated by the NN. We can distinguish exceedances for both models during the Covid-19 crisis as well as during the down pic in June 2020. Although they adapt quickly, the most brutal changes cannot be detected right away by the models. The neural network model seems to adapt quicker than the econometric model but estimates less extreme values for the Covid-19 period.

There are multiple ways to compute the performance of a VaR models, however, we choose to focus on the number of exceedances.

Exceedance 1-day ahead Forecast		
	VARMA- DCC-GARCH	AE-Stacked LSTM
VaR	16	16
ES	8	8

Table 9: Number of Exceedances for the 1-day ahead forecasts

Following the results of the Value at Risk exceedances, both models seem to be worthwhile. The number of exceedances for both the VaR and ES are the same. There is only one difference in exceedance. The econometric model does not make an error in December 2019 while the AE-Stacked LSTM does. On the other hand, the AE-Stacked LSTM adapts quicker to the increase in September 2020 which allows it not to make two exceedances in a row. The AE-Stacked LSTM seems also better suited for low volatility periods as it is closer to the returns. Both models having the same number of exceedance is not surprising as 16 and 8 might be the hardest values to detect. Therefore, despite a difference in the model performance or estimations, the exceedances can be the same.

	S&P500		
	Observed	VARMA- DCC-GARCH Simu.	AE-Stacked LSTM Simu.
Mean	0.08004%	-0.01853%	0.02336%
Std. Deviation	1.957%	1.802%	1.792%
5% percentile	-2.882%	-2.446%	-2.429%
95% percentile	2.260%	2.432%	2.515%

Table 10: Descriptive statistics about the observed and simulated data over the one day ahead models

The values of the unconditional mean, standard deviation, 5 and 95 percentiles are represented in the Table above. 90% of the simulations of the benchmark model evolves between -2.446% and 2.432% while for the NN, between -2.429% and 2.515% . The simulations values are close to the observed data with a slight difference in the 5% percentile that might be due to the Covid-19 crisis. In fact, besides the 2008 crisis, the models did not have much crisis data to train on. Which means once we face another crisis, it is hard to forecast good results.

Concerning the 10-days ahead, the VaR and the ES computed using the VARMA-DCC-GARCH is displayed below.

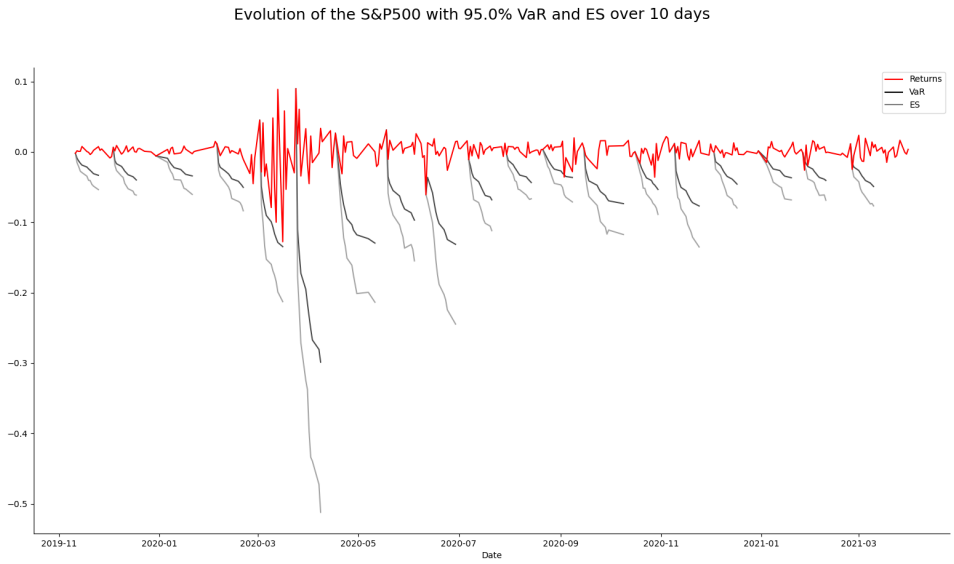

Figure 25: Value at Risk for the VARMA- DCC-GARCH for the 10 days ahead forecast

While the results for the 10-days ahead computed using the AE-Stacked LSTM is displayed below.

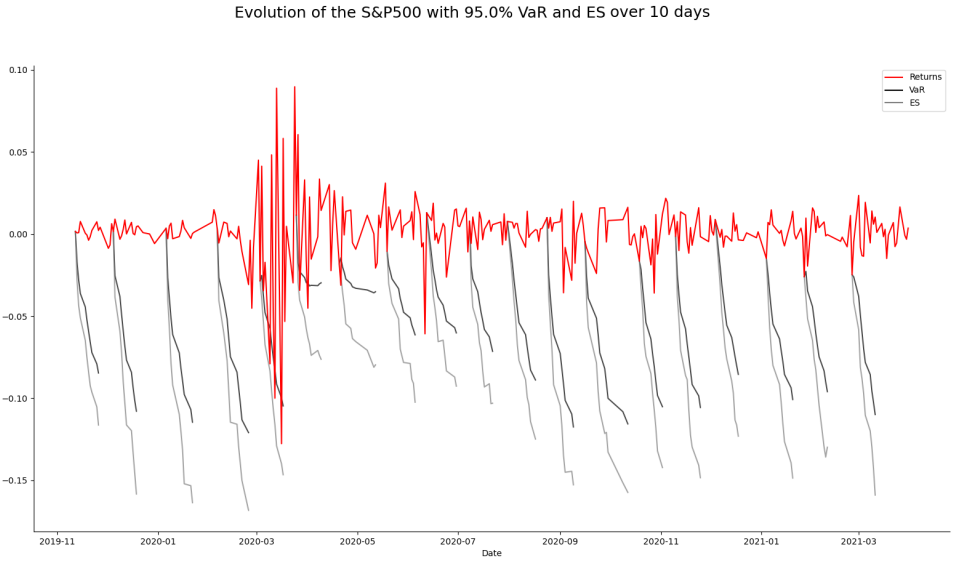

Figure 26: Value at Risk for the AE-Stacked LSTM for the 10 days ahead forecast

The bad results obtained via the neural network model are not surprising given the poor forecast of the mean and volatility. We observe that the VaR is badly estimated with very high values in calm periods. The Covid-19 crisis does not seem to have any impact on the Value at Risk forecasts. On the other hand, the benchmark model's values are very coherent with the periods. Low values of VaR are estimated in calm periods and higher values are estimated in and after crisis periods. The number of exceedances is lower for the benchmark model which confirms the superiority of the model.

Exceedance 10-days ahead Forecast		
	VARMA- DCC-GARCH	AE-Stacked LSTM
VaR	72	82
ES	18	30

Table 11: Number of Exceedances for the 10-days ahead forecasts

There are less exceedances in the benchmark model for both the VaR and the ES. Most of the additional exceedances in the neural network are due to a non adaptation to the period of high volatility. Therefore, the VaR is just badly estimated every day until the volatility decreases again.

	S&P500 - 10 Days Ahead		
	Observed	VARMA- DCC-GARCH Simu.	AE-Stacked LSTM Simu.
Mean	0.08004%	-0.00616%	-1.66101%
Std. Deviation	1.957%	4.149%	3.295%
5% percentile	-2.882%	-5.576%	-7.537%
95% percentile	2.260%	5.580%	3.277%

Table 12: Descriptive statistics about the observed and simulated data over the 10 days ahead models

All the estimations of the neural network are centred over a negative mean which is far from the observed data. We can notice the low value of the 5% percentile with the neural network.

With these results, it is worth noting that the neural network was trained once and forecasted the whole test set while the statistical model was re-trained over each day for the next. It is an implementation choice that we made for this work, however, the neural network could have been re-trained every month, for example. For both one-day ahead models, there is a $\hat{\alpha}$=5.4% VaR exceedance and 2.7% ES exceedance. For the 10-days ahead VARMA- DCC-GARCH, there is a $\hat{\alpha}$=24.3% VaR and 6.1% ES exceedance. While for the AE-Stacked LSTM, we have a $\hat{\alpha}$=27.7% and 10.1% ES exceedance [8]. Following theses values, the econometric model provides a better risk coverage as the expression $|\alpha - \hat{\alpha}|$ is closer to zero.

[8] These values of $\hat{\alpha}$ are computed using the number of exceedances over the total number of observations forecasted

11 Conclusion

Financial forecasting may be difficult but is a very useful task in the field of market risk management. Statistical models such as the VARMA- DCC-GARCH have already proven their worth whereas the machine learning models are being researched and developed continuously. Unlike statistical models, machine learning models do not rely on assumptions in order to process the data. Furthermore statistical models often suffer from the curse of dimensionality and need recurrent retraining to keep performing while the machine learning models can process more data and features before lacking efficiency.

The objective of our master thesis was to provide a comprehensive comparison between the benchmark econometric model and a LSTM neural network on the computation of the Value at Risk. For the benchmark econometric model, we used a Vector AutoRegressive Moving Average to estimate the conditional mean of the S&P500. To estimate the conditional volatility, we adopted the Dynamic Conditional Correlation model. The architecture of the neural network used is built with two blocks. The first one used for feature extraction and the second one used to forecast the parameters needed based on the latent codings and the exogenous variables. From the conditional estimations, we simulate 10.000 scenarios of returns and compute the VaR and ES. The neural network developed for the one-day ahead forecasting is worth the statistical model. They both return coherent simulations with good results on VaR exceedances. The main advantage of the AE-Stacked LSTM is its ability to adapt quicker to changes in the underlying returns dynamics. However, concerning the 10-days ahead model, it returns bad results as it is not capable of learning patterns in the data. We have major problems forecasting the mean and volatility over 10-days as the estimated values are almost constant over the whole test set. The results on the estimated Value at Risk are not conclusive which highlights the superiority of the statistical model.

In an effort to improve the models, one can gather more exogenous variables and feed them all into the autoencoder to create a latent matrix that takes information from all the variables at once. This might emphasize the importance of the exogenous variables in the model. Moreover, we can introduce the Transformers instead of the LSTM. The Transformers is an evolution of the LSTM layers introduced in 2017 which replace the recurrent feature by an attention mechanism to process time series [3].
Going out of the framework of this master thesis, we introduce the Generative Adversarial Neural Networks (GAN). This is a type of neural network that focus on generating artificial data from the original subset of observations. GANs are based on a game scenario where there are two players. First, the generator network produces samples $z = g(x, \theta^g)$ based on the original data. Then, the discriminator which goals is to detect the fake sample produced by the generator from the original data, with a probability $p = d(z, \theta^d)$. Ultimately, the goal of the generator is to produce samples that are not distinguishable from the discriminator therefore considered as simulations [44].

Machine learning models have a lot of room to evolve. This implies to do researches with results not as satisfying as desired. However, it does not discredit the possibilities of progress through machine learning models.

Appendices

A Proof of Hypothesis and Theorems

Intuitive example about the VaR being non-subadditive

Let's consider 2 different outcomes possible for X and Y.

$$X = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, Y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

with a probability of occurrence is $p = 0.80$ for $X = Y = 0$ and $1 - p = 0.20$ for $X = Y = 1$. Calculating the Value at Risk for X and Y at level 0.75, we obtain,

$$VaR_{X,0.75} = 0 \quad and \quad VaR_{Y,0.75} = 0$$

However, $P(X + Y = 0) = p^2 = 0.64$, $P(X + Y = 1) = 2p(1 - p) = 0.32$ and $P(X + Y = 2) = (1 - p)^2 = 0.04$. Thus calculating the VaR at level 0.75 for $X + Y$ is returns,

$$VaR_{X+Y,\,0.75} = 1$$

which proves the non-subadditivity of the Value at Risk.

Description of the Maximum likelihood estimation for the VARMA model.
The following explanations come from the book *Helmut Lütkepohl, "New Introduction to Multiple Time Series Analysis", Springer, pg 467-469.*
The likelihood function used to estimate the parameters depends on the mean μ, the matrix of parameters γ and a white noise matrix of variance-covariance Σ_ϵ. Let's consider a de-meaned time series, the likelihood is equivalent to,

$$L(\mu, \gamma, \Sigma_\epsilon) = \prod_{t=1}^{T} \frac{1}{\sqrt{(2\pi)^n |\Sigma_\epsilon|}} exp\bigg(- \frac{\epsilon_t(\mu,\gamma)' \Sigma_\epsilon^{-1} \epsilon_t(\mu,\gamma)}{2} \bigg)$$

$$with \quad \epsilon_t(\mu,\gamma) = (x_t - \mu) - \sum_{i=1}^{t-1} \prod_i (\gamma)(x_{t-i} - \mu)$$

$\sum_{i=1}^{t-1} \prod_i(\gamma)$ being the coefficient of the VAR components. The log-likelihood function is defined, to a constant, as,

$$log(L(\mu, \gamma, \Sigma_\epsilon)) = -\frac{T}{2} log(|\Sigma_\epsilon|) - \frac{1}{2} \sum_{t=1}^{T} \epsilon_t(\mu,\gamma)' \Sigma_\epsilon^{-1} \epsilon_t(\mu,\gamma)$$

Thereafter, we can estimate the parameters by computing the partial derivative of the log-likelihood with respect to the parameters μ, γ and Σ_ϵ.

$$\frac{\partial log(L)}{\partial \mu} = - \sum_{t=1}^{T} \mu_t \Sigma_\mu^{-1} \frac{\partial \mu_t}{\partial \mu}$$

$$\frac{\partial log(L)}{\partial \gamma} = - \sum_{t=1}^{T} \mu_t \Sigma_\mu^{-1} \frac{\partial \mu_t}{\partial \gamma}$$

$$\frac{\partial log(L)}{\partial \Sigma_\epsilon} = -\frac{T}{2} \Sigma_\epsilon^{-1} + \frac{1}{2} \Sigma_\epsilon^{-1} (\sum_{t=1}^{T} \epsilon_t \epsilon_t') \Sigma_\epsilon^{-1}$$

A recursive formula for computing $\partial \mu_t / \partial \gamma$ ca be computed. This goes out of the framework of the master thesis. The lemma to solve this optimization problem is present in *Helmut Lütkepohl, "New Introduction to Multiple Time Series Analysis", Springer, pg 468.*

Description of the Maximum likelihood estimation for the DCC-GARCH [11]
We focus on the case of multivariate normal standardized residuals with the parameters noted as $\theta = (\phi, \psi, P_c)$ with $\psi = (\alpha_1, ..., \alpha_p, \beta_1, ..., \beta_q)$ and $\phi = (\phi_1, ..., \phi_d)$. The likelihood is given as,

$$L(\theta) = \prod_{t=1}^{T} \frac{1}{\sqrt{(2\pi)^n |\Sigma_t(\theta)|}} exp\Big(- \frac{X_t' \Sigma_t(\theta) X_t}{2} \Big)$$

Recall the covariance function, $\Sigma_t = \Delta_t P_t \Delta_t$, we can write the log-likelihood as,

$$log(L(\theta)) = -\frac{1}{2} \sum_{t=1}^{T} \Big(n\, log(2\pi) + log(|\Sigma_t(\theta)|) + X_t^T \Sigma_t(\theta) X_t \Big)$$

$$= -\frac{1}{2} \sum_{t=1}^{T} (d\, log(2\pi) + 2log(|\Delta_t(\phi)|) + log(|P_t(\psi, P_c)|)$$

$$+ X_t^T \Delta_t^{-1}(\phi) P_t^{-1}(\psi, P_c) \Delta_t^{-1}(\phi) X_t)$$

where Δ_t and P_t depends on ϕ, ψ and P_c. The process of minimizing this function with respect to θ is too complicated to be done in one step. Therefore, the usual technique is divided into two different steps.

First step: We replace the term $|P_t(\psi, P_c)|$ with the identity matrix as it simplifies the equation. And we estimate the term of volatility.

$$log(L(\theta)) = -\frac{1}{2} \sum_{t=1}^{T} (d\, log(2\pi) + 2log(|\Delta_t(\phi)|) + X_t^T \Delta_t^{-1}(\phi) I_d \Delta_t^{-1}(\phi) X_t)$$

$$= -\frac{1}{2} \sum_{t=1}^{T} \left(d\, log(2\pi) + \sum_{i=1}^{n} \left[log(\sigma_{t,i}^2) + \frac{X_{t,i}^2}{\sigma_{t,i}^2} \right] \right)$$

$$= \sum_{i=1}^{n} \left(- \frac{1}{2} \sum_{t=1}^{T} [log(\sigma_{t,i}^2) + \frac{X_{t,i}^2}{\sigma_{t,i}^2}] + C \right)$$

with C as a constant. Maximizing this equation comes down to fitting each univariate models separately. After calculating the parameter ϕ, we extract the conditional volatilities $\sigma_{t,i}$ and compute the standardized residuals noted Z_t. Thereafter, we can estimate the correlation matrix P_c using classical estimators. After these steps have been performed, it only remains to estimate ψ.

Second step: In the second step, we treat the log-likelihood function as a function of ψ, ignoring constants, returns,

$$log(L(\psi)) = -\frac{1}{2} \sum_{t=1}^{T} (log(|P_t|) + Z_t^T P_t^{-1} Z_t)$$

Then, we can maximize this function with respect to ψ.

B Graphs and Tables

	Formulations	Restrictions
EGARCH (1,1)	$\log(\sigma_t^2) = \alpha_0 + \gamma\left(\frac{\varepsilon_{t-1}}{\sigma_{t-1}}\right) + \alpha_1\left(\left\lvert\frac{\varepsilon_{t-1}}{\sigma_{t-1}}\right\rvert - \sqrt{\frac{2}{\pi}}\right) + \beta \log(\sigma_{t-1}^2)$	$\alpha_1 + \beta < 1$
GJR-GARCH (1,1)	$\sigma_t^2 = \alpha_0 + \alpha_1\varepsilon_{t-1}^2 - \gamma\varepsilon_{t-1}^2 S_{t-1}^- + \beta\sigma_{t-1}^2$ $S_{t-1}^- = 1$ for $\varepsilon_{t-1} < 0$ and $S_{t-1}^- = 0$ otherwise	$\alpha_0 > 0,\ \alpha_1,\ \beta > 0$ $\alpha_1 + \beta < 1$
TS GARCH (1,1)	$\sigma_t = \alpha_0 + \alpha_1\lvert\varepsilon_{t-1}\rvert + \beta\sigma_{t-1}$	$\alpha_0 > 0,\ \alpha_1,\ \beta > 0$ $\alpha_1 + \beta < 1$
TGARCH	$\sigma_t = \alpha_1 + \alpha_1\lvert\varepsilon_{t-1}\rvert + \gamma\varepsilon_{t-1}S_{t-1}^- + \beta\sigma_{t-1}$ $S_{t-1}^- = 1$ for $\varepsilon_{t-1} < 0$ and $S_{t-1}^- = 0$ otherwise	$\alpha_0 > 0,\ \alpha_1,\ \beta > 0$ $\alpha_1 + \beta < 1$
PGARCH (1,1)	$\sigma_t^\delta = \omega + \alpha\lvert\varepsilon_{t-1}\rvert^\delta + \beta\sigma_{t-1}^\delta$	$\omega > 0,\ \alpha \geq 0$ $\beta \geq 0,\ \delta > 0$
APGARCH (1,1)	$\sigma_t^\delta = \alpha_0 + \alpha_1(\lvert\varepsilon_{t-1}\rvert + \gamma\varepsilon_{t-1})^\delta + \beta\sigma_{t-1}^\delta$	$\alpha_0 > 0,\ \alpha_1 \geq 0,\ \beta > 0,$ $\delta > 0 - 1 < \gamma < 1$
AGARCH (1,1)	$\sigma_t^2 = \alpha_0 + \alpha_1\varepsilon_{t-1}^2 + \gamma\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2$	$\alpha_0 > 0,\ \alpha_1, \beta > 0$ $\alpha_1 + \beta < 1$
SQR-GARCH	$\sigma_t^2 = \alpha_0 + \alpha_1(\gamma\sigma_{t-1} + \varepsilon_{t-1}/\sigma_{t-1})^2 + \beta\sigma_{t-1}^2$	$\alpha_0 > 0,\ \alpha_1, \beta > 0$ $\alpha_1 + \beta < 1$
Q-GARCH	$\sigma_t^2 = \alpha_0 + \alpha_1(\varepsilon_{t-1} + \gamma\alpha_{t-1})^2 + \beta\sigma_{t-1}^2$	$\alpha_0 > 0,\ \alpha_1, \beta > 0$ $\alpha_1 + \beta < 1$
VGARCH	$\sigma_t^2 = \alpha_0 + \alpha_1(\gamma + \sigma_{t-1}^{-1}\varepsilon_{t-1})^2 + \beta\sigma_{t-1}^2$	$\alpha_0 > 0, 0 < \beta < 1$ $0 < \alpha_1 < 1$
NAGARCH (1,1)	$\sigma_t^2 = \alpha_0 + \alpha_1(\varepsilon_{t-1} + \gamma\sigma_{t-1})^2 + \beta\sigma_{t-1}^2$	$\alpha_0 > 0, \alpha_1, \beta > 0$ $\alpha_1 + \beta < 1$
MS-GARCH (1,1)	$r_t = \mu_{s_t} + \varepsilon_t = \mu_{s_t} + \sigma_t z_t$ with z_t iid $N(0,1)$ $\sigma_t^2 = \omega_{s_t} + \alpha_{s_t}\varepsilon_{t-1}^2 + \beta_{s_t}\sigma_{t-1}^2$	s_t: state of the process at the t $\omega_{s_t} > 0, \alpha_{s_t} \geq 0,$ and $\beta_{s_t} \geq 0$
RS-APARCH	$\sigma_{s_t}^{\delta_{st}} = \omega_{s_t} + \alpha_{s_t}(\lvert\varepsilon_{t-1}\rvert + \gamma_{s_t}\varepsilon_{t-1})^{\delta_{s_t}} + \beta_{s_t 2}\sigma_{t-1}^{\delta_{s_t}}$	$\omega_{s_t} > 0, \alpha_{s_t} \geq 0, \beta_{s_t} > 0$ $\delta > 0 - 1 < \gamma_{s_t 1} < 1$

Figure 27: A non-exhaustive list of some GARCH-type models used in the framework of VaR. The table comes directly from the paper *"A comprehensive review of Value at Risk methodologies "*.

Results for equation S&P500			
	coef	std err	z
L1.S&P500	-0.1904	2.866	-0.066
L1.Nikkei	0.0215	0.088	0.244
L1.FTSE	0.0155	0.073	0.212
L1.DowJones	0.0054	3.124	0.002
L1.e(S&P500)	0.0028	2.871	0.001
L1.e(Nikkei)	-0.0201	0.091	-0.222
L1.e(FTSE)	-0.0104	0.075	-0.139
L1.e(DowJones)	0.0168	3.129	0.005

Table 13: VARMA(1,1) coefficient for the S&P500 with the standard error and z-values over the entire data set.

Coefficient for the DCC-GARCH(1, 1)			
	Estimate	Std. Error	p-value
[X1].mu	0.000724	0.000170	0.000021
[X1].omega	0.000003	0.000004	0.461651
[X1].alpha1	0.145569	0.019523	0.000000
[X1].beta1	0.830412	0.028464	0.000000
[X2].mu	0.000384	0.000223	0.084390
[X2].omega	0.000007	0.000002	0.000777
[X2].alpha1	0.155128	0.017652	0.000000
[X2].beta1	0.810763	0.021158	0.000000
[X3].mu	-0.000022	0.000165	0.893983
[X3].omega	0.000003	0.000003	0.328200
[X3].alpha1	0.126300	0.034970	0.000304
[X3].beta1	0.852832	0.041636	0.000000
[X4].mu	0.000746	0.000153	0.000001
[X4].omega	0.000003	0.000003	0.243123
[X4].alpha1	0.146839	0.018915	0.000000
[X4].beta1	0.829217	0.022879	0.000000
[Joint]dcca1	0.029179	0.006456	0.000006
[Joint]dccb1	0.938159	0.018262	0.000000
[Joint]mshape	7.396388	0.442681	0.000000

Table 14: VARMA(1,1) coefficient for the S&P500 with the standard error and z-values over the entire data set.

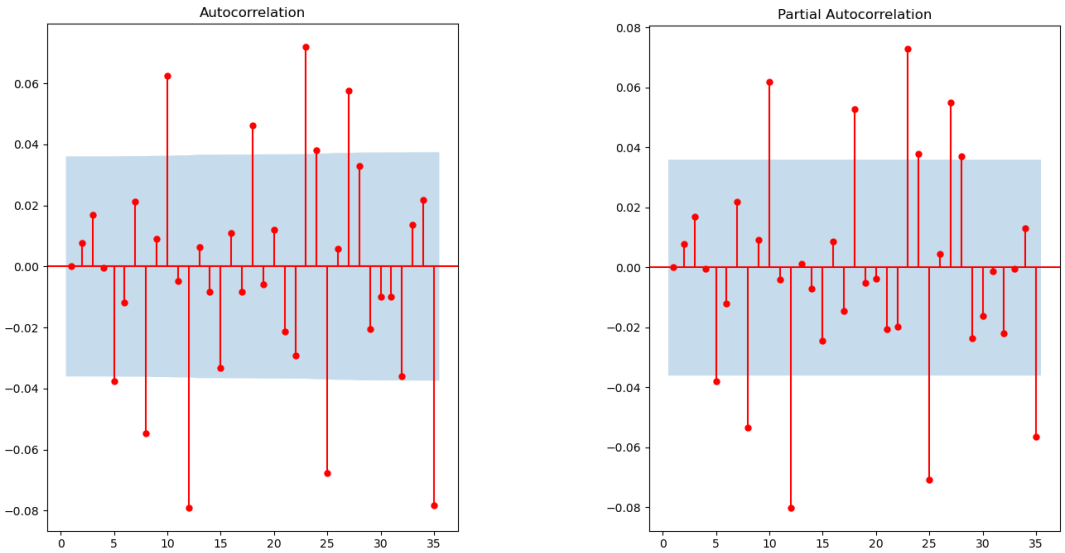

Figure 28: ACF and PACF plots of the residuals of the VARMA(1, 1)

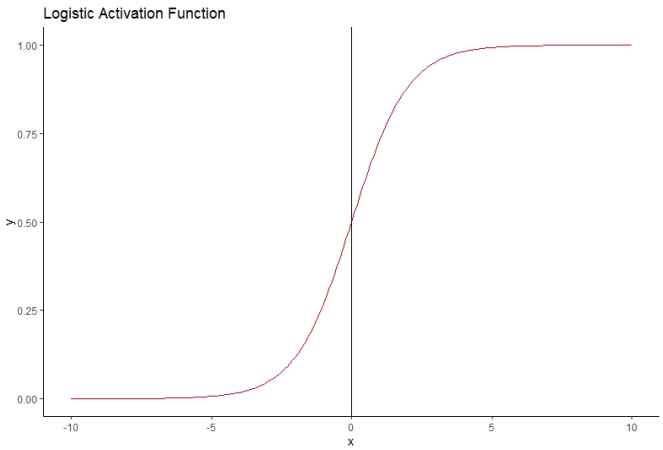

Figure 29: Representation of the Logistic activation function.

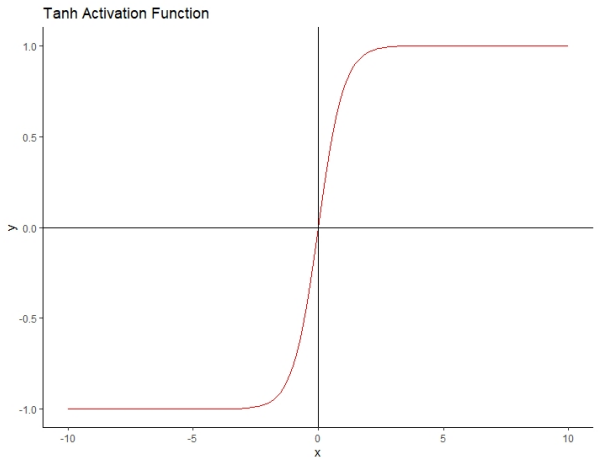

Figure 30: Representation of the Tanh activation function.

AE-LSTM Mean model- One day ahead		
Layers	Output Shape	Param #
Input Layer	(None, 45, 1)	0
LSTM	(None, 45, 2)	32
LSTM	(None, 45, 3)	72
LSTM	(None, 45, 2)	48
TimeDistributed	(None, 45, 1)	3
Total params: 155		
Trainable params: 155		
Non trainable params: 0		

Table 15: AE-LSTM architecture to model the mean one day ahead

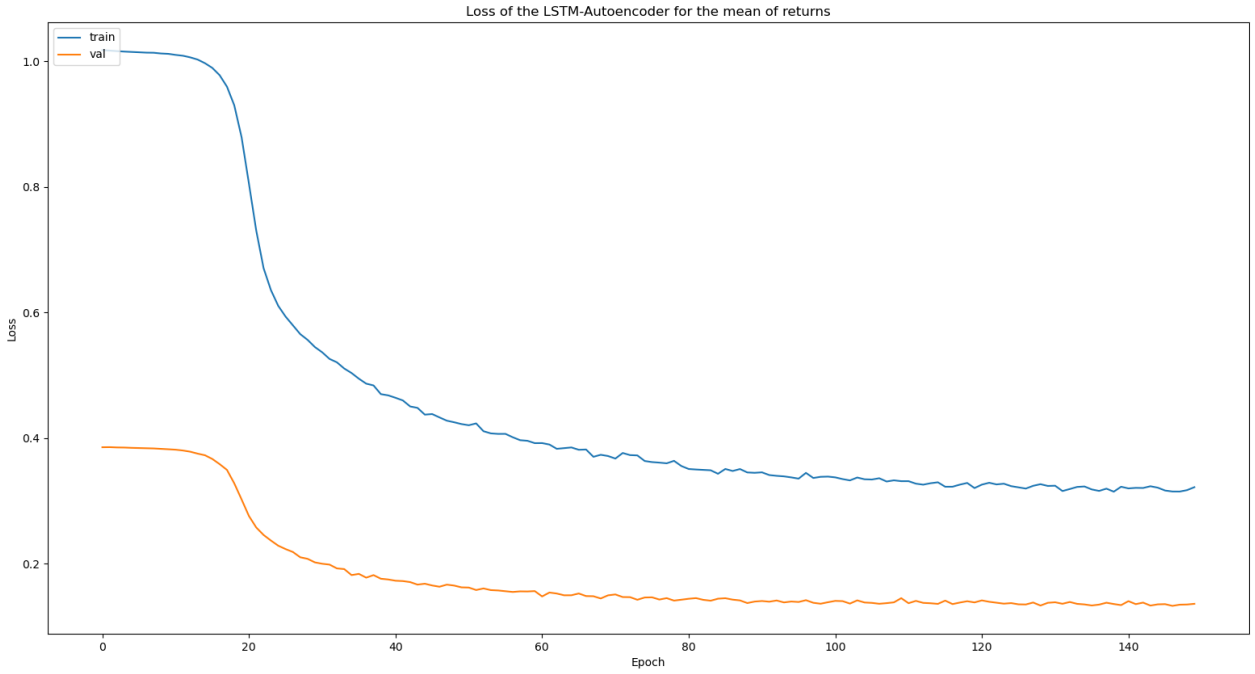

Figure 31: Representation of the Loss function for the LSTM-Autoencoder, one-day ahead mean model.

Stacked LSTM Mean model- One day ahead		
Layers	Output Shape	Param #
Input Layer	(None, 45, 6)	0
LSTM	(None, 45, 15)	1320
LSTM	(None, 45, 10)	1040
LSTM	(None, 5)	320
Dense	(None, 1)	6
Total params: 2,686		
Trainable params: 2,686		
Non trainable params: 0		

Table 16: Stacked LSTM architecture to model the mean one day ahead

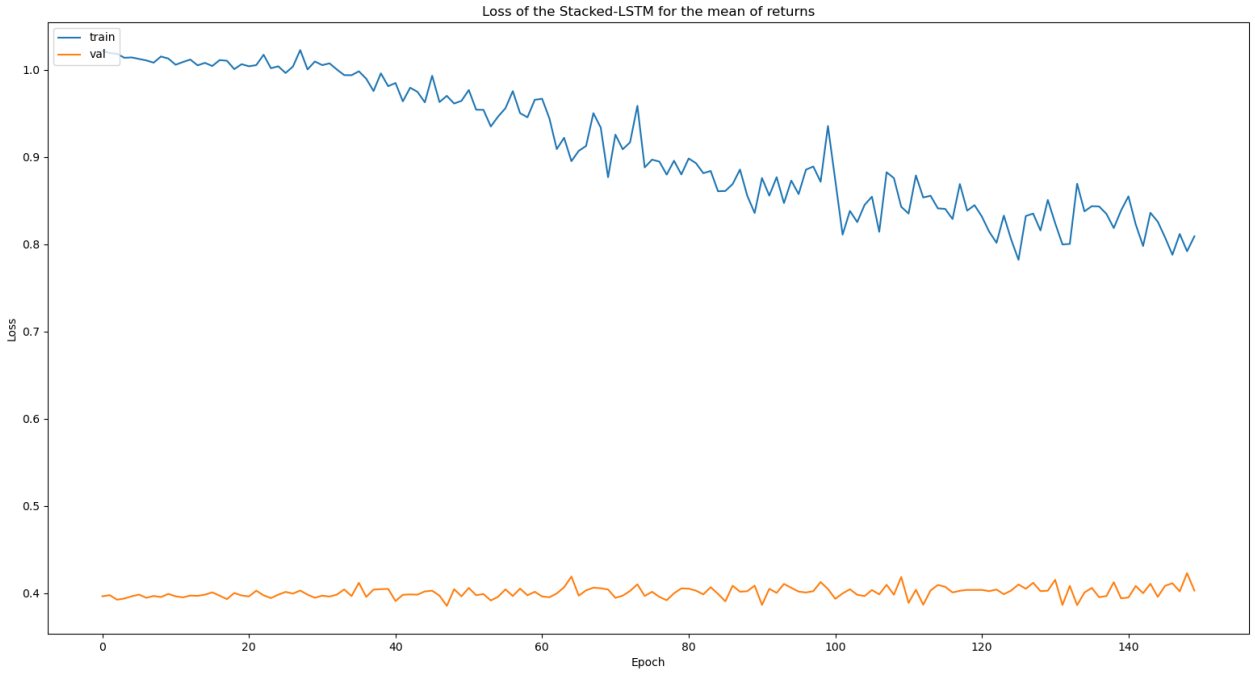

Figure 32: Representation of the Loss function for the Stacked-LSTM, one-day ahead mean model.

AE LSTM Volatility model- One day ahead		
Layers	Output Shape	Param #
Input Layer	(None, 45, 1)	0
LSTM	(None, 45, 3)	2320
LSTM	(None, 45, 5)	1240
LSTM	(None, 45, 3)	11
TimeDistributed	(None, 45, 1)	4
Total params: 332		
Trainable params: 332		
Non trainable params: 0		

Table 17: AE-LSTM architecture to model the volatility one day ahead

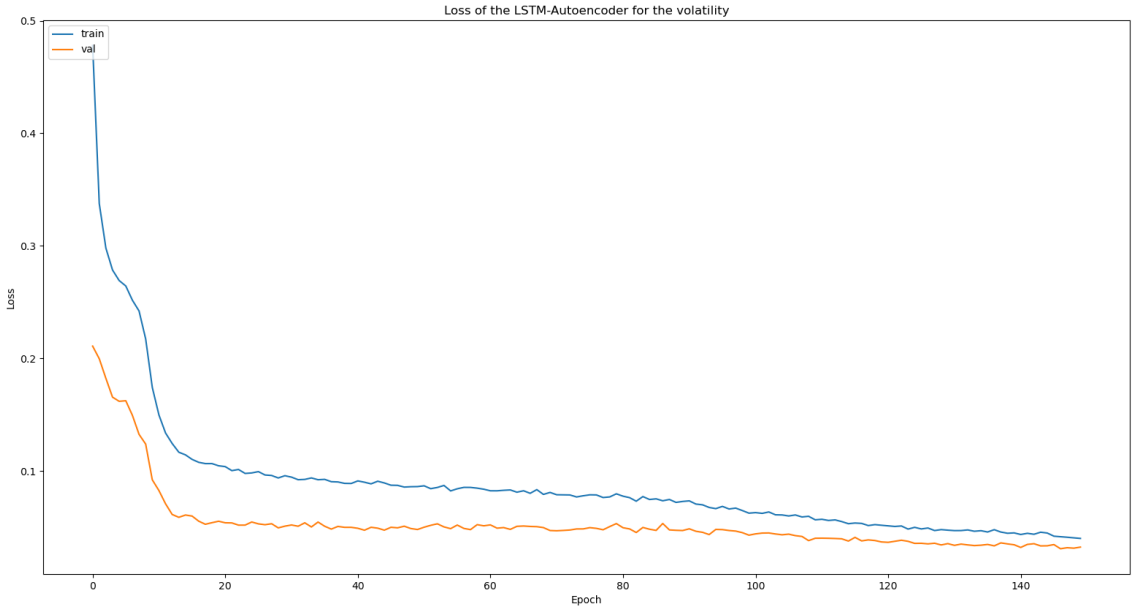

Figure 33: Representation of the Loss function for the LSTM-Autoencoder, one-day ahead volatility model.

Stacked LSTM Volatility model- One day ahead		
Layers	Output Shape	Param #
Input Layer	(None, 45, 8)	0
LSTM	(None, 45, 20)	2320
LSTM	(None, 10)	1240
Dense	(None, 1)	11
Total params: 3,571		
Trainable params: 3,571		
Non trainable params: 0		

Table 18: Stacked LSTM architecture to model the volatility one day ahead

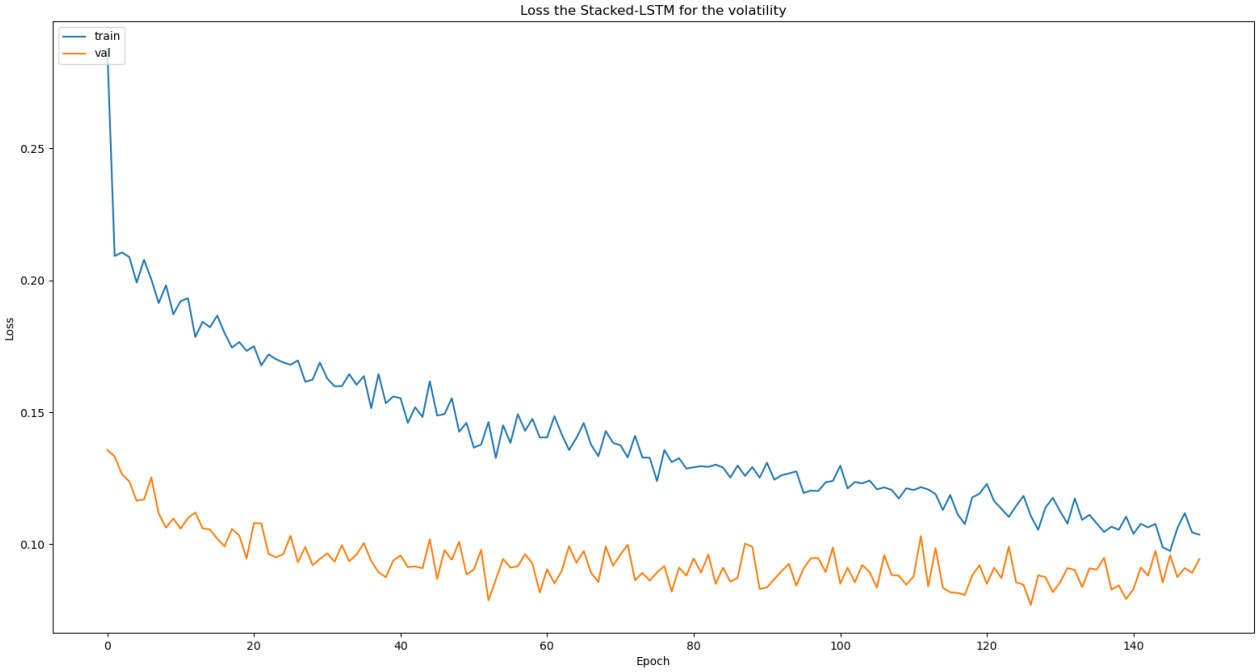

Figure 34: Representation of the Loss function for the Stacked-LSTM, one-day ahead volatility model.

AE LSTM Mean model- 10 days ahead		
Layers	Output Shape	Param #
Input Layer	(None, 55, 1)	0
LSTM	(None, 55, 3)	60
LSTM	(None, 55, 5)	180
LSTM	(None, 55, 7)	364
LSTM	(None, 55, 5)	260
LSTM	(None, 55, 3)	108
TimeDistributed	(None, 55, 1)	4
Total params: 976		
Trainable params: 976		
Non trainable params: 0		

Table 19: AE-LSTM architecture to model the mean 10 days ahead

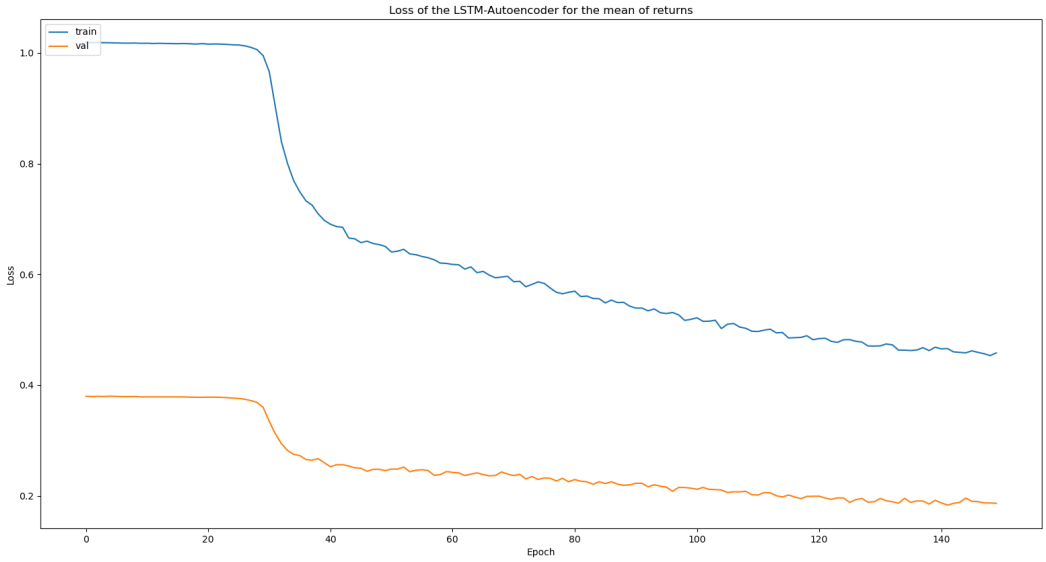

Figure 35: Representation of the Loss function for the LSTM-Autoencoder, 10-days ahead mean model.

Stacked LSTM Mean model- 10 days ahead		
Layers	Output Shape	Param #
Input Layer	(None, 55, 10)	0
LSTM	(None, 55, 35)	6440
LSTM	(None, 55, 20)	4480
LSTM	(None, 45)	11880
Dense	(None, 20)	920
Dense	(None, 10)	210
Total params: 23,930		
Trainable params: 23,930		
Non trainable params: 0		

Table 20: Stacked-LSTM architecture to model the mean 10 days ahead

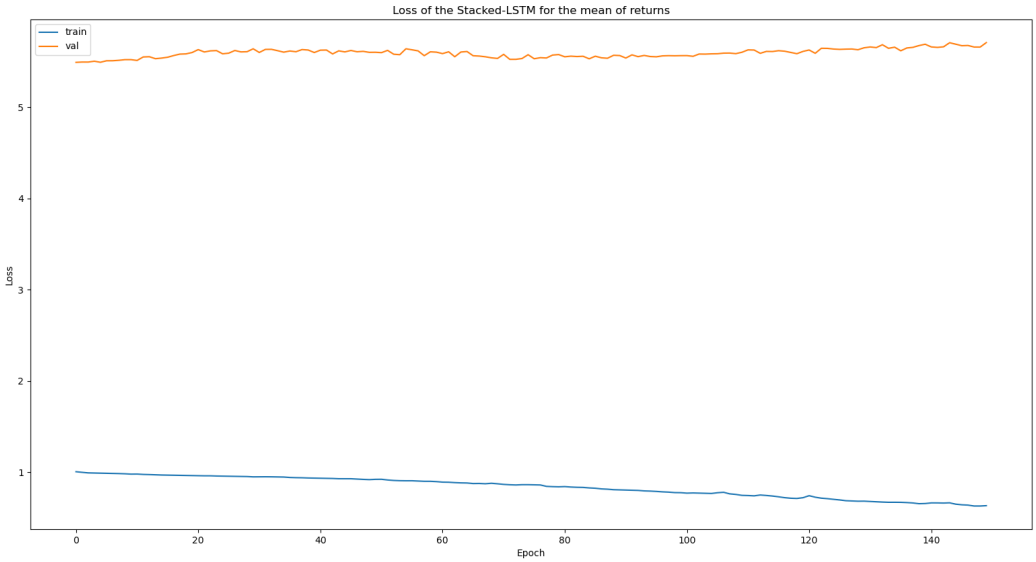

Figure 36: Representation of the Loss function for the Stacked-LSTM, 10-days ahead mean model.

AE LSTM Volatility model- 10 days ahead		
Layers	Output Shape	Param #
Input Layer	(None, 55, 1)	0
LSTM	(None, 55, 2)	32
LSTM	(None, 55, 4)	112
LSTM	(None, 55, 6)	264
LSTM	(None, 55, 4)	176
LSTM	(None, 55, 2)	56
TimeDistributed	(None, 55, 1)	3
Total params: 643		
Trainable params: 643		
Non trainable params: 0		

Table 21: AE-LSTM architecture to model the volatility 10 days ahead

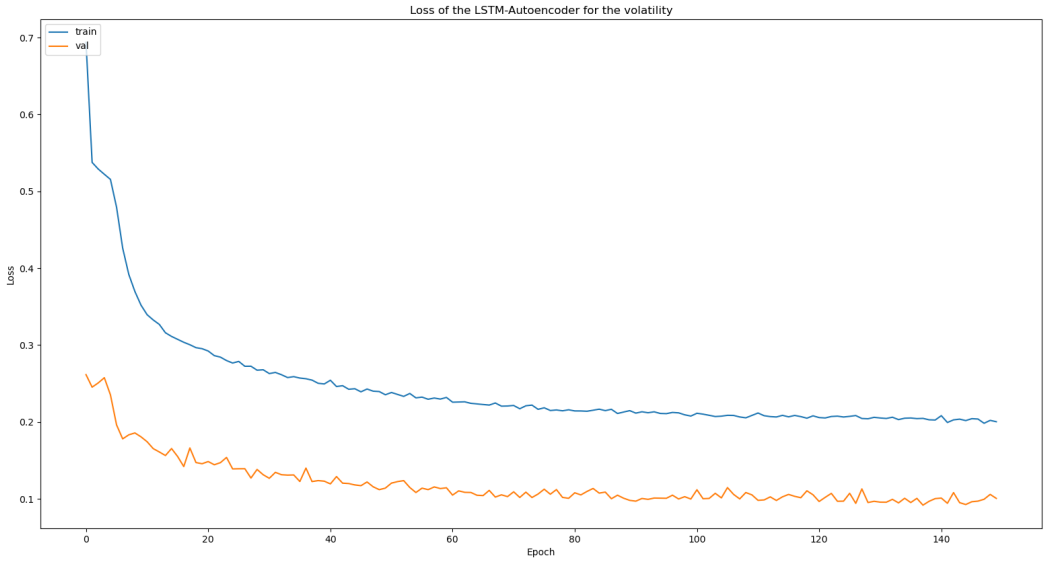

Figure 37: Representation of the Loss function for the LSTM-Autoencoder, 10-days ahead volatility model.

Stacked LSTM Volatility model- 10 days ahead		
Layers	Output Shape	Param #
Input Layer	(None, 55, 9)	0
LSTM	(None, 55, 25)	3500
LSTM	(None, 55, 15)	2460
LSTM	(None, 20)	2880
Dense	(None, 10)	210
Total params: 9,050		
Trainable params: 9,050		
Non trainable params: 0		

Table 22: Stacked-LSTM architecture to model the volatility 10 days ahead

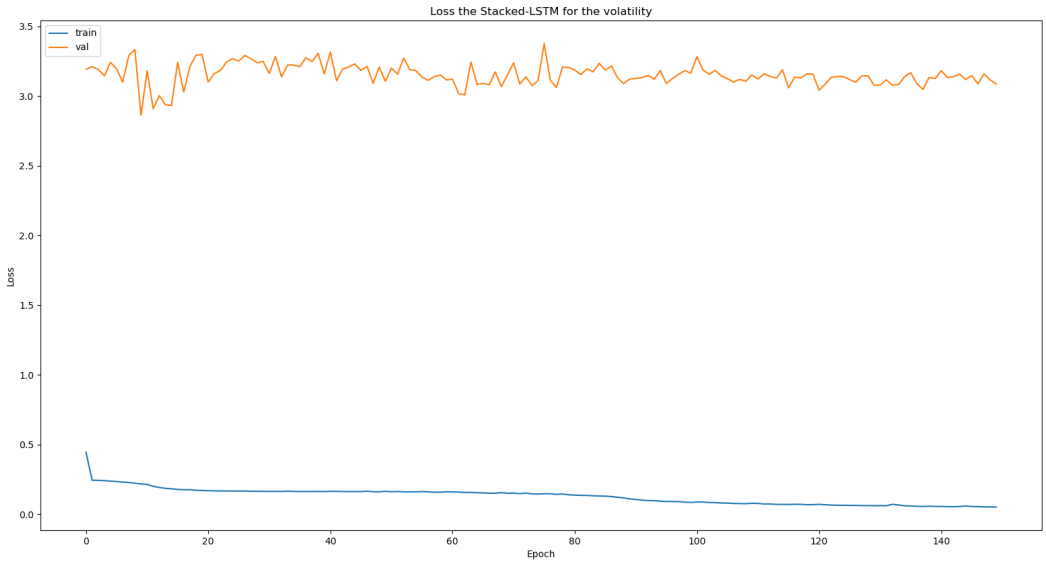

Figure 38: Representation of the Loss function for the Stacked-LSTM, 10-days ahead volatility model.

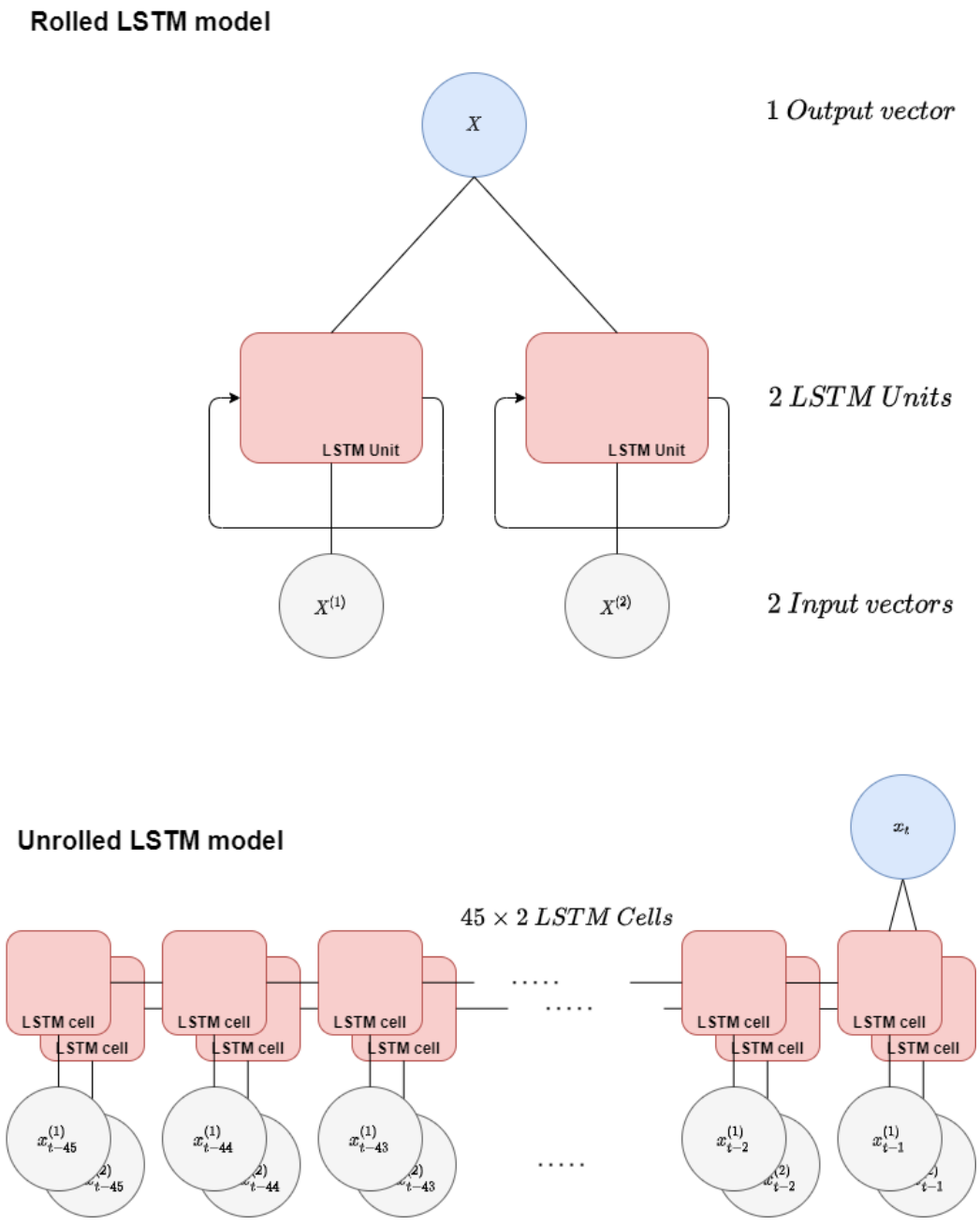

Figure 39: Representation of both rolled and unrolled LSTM models taking as input (45x2) and outputs (45x1).

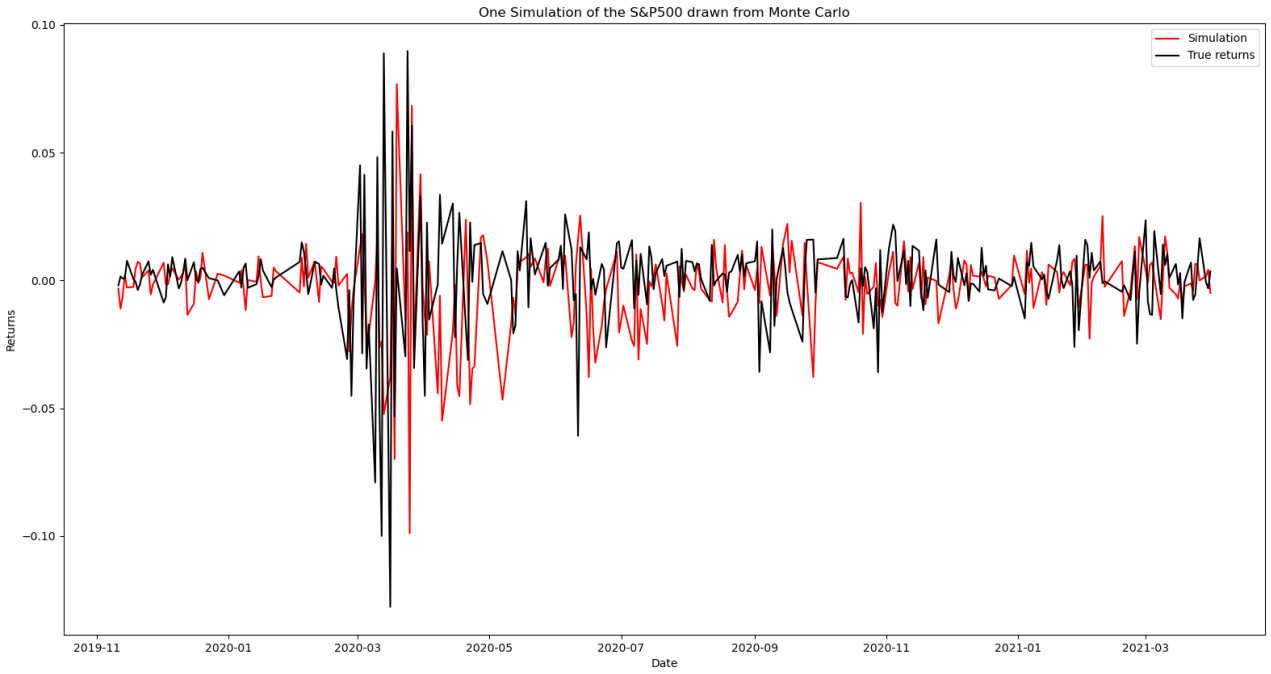

Figure 40: Representation of one simulation of the S&P500 with the VARMA- DCC-GARCH.

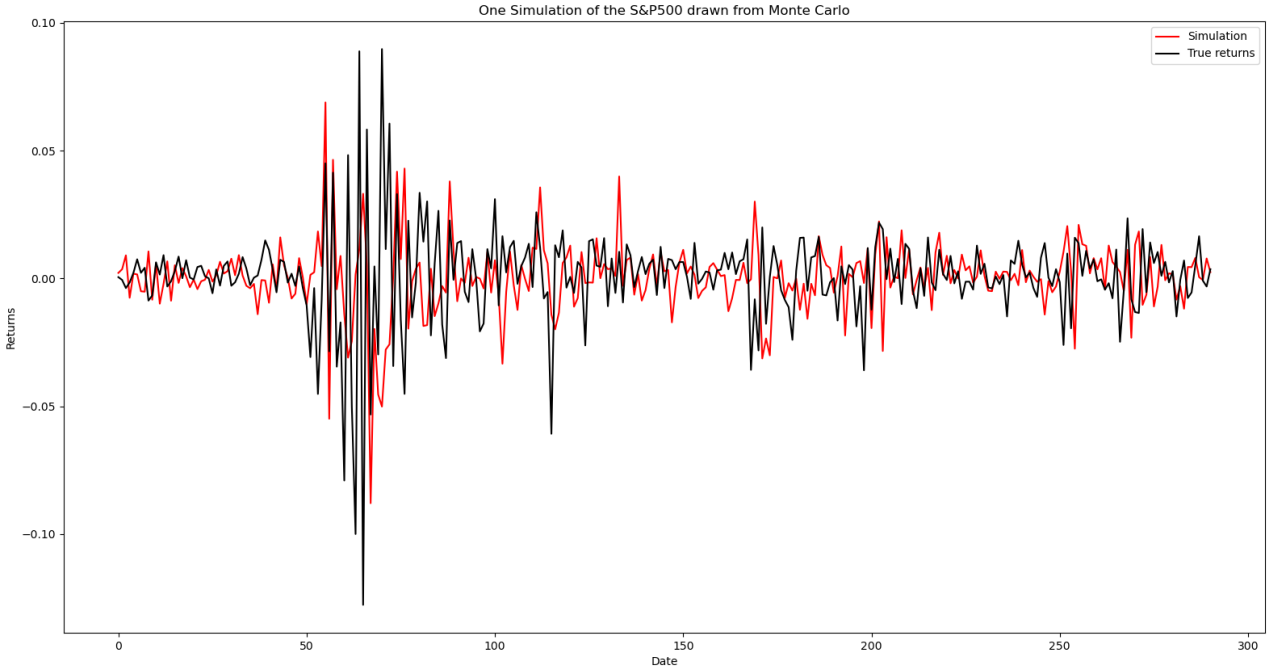

Figure 41: Representation of one simulation of the S&P500 with the AE-Stacked LSTM.

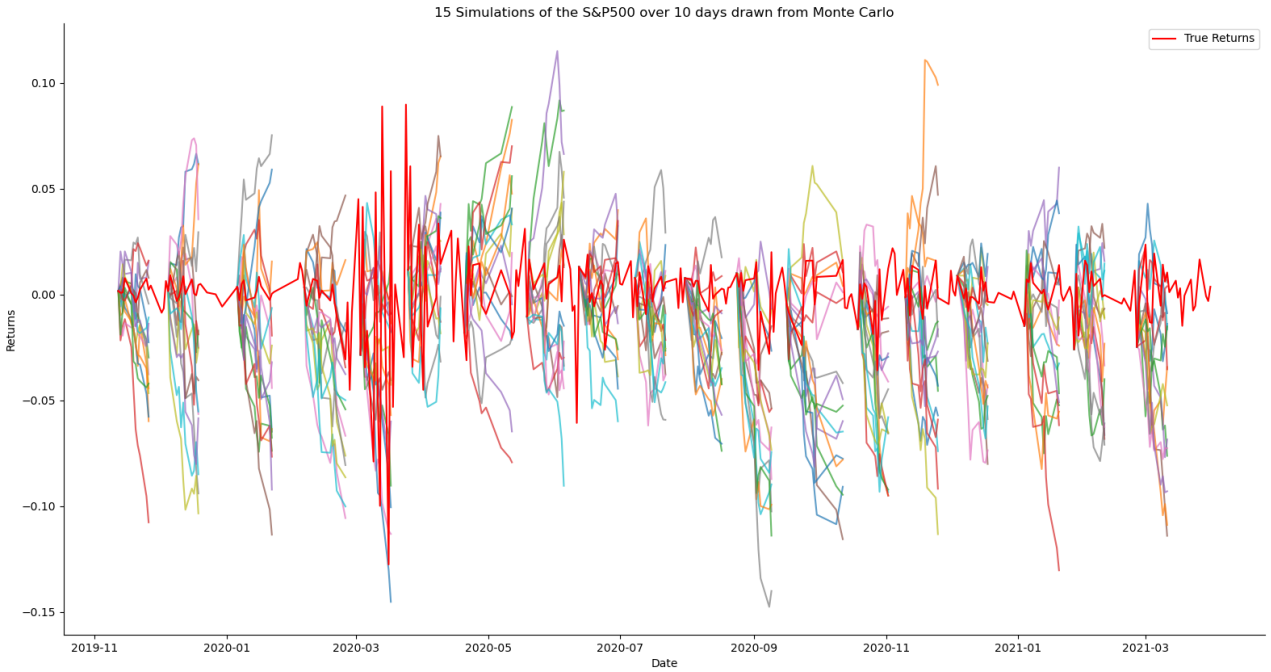

Figure 42: Representation of 10 simulations of the S&P500 10 days-ahead with the AE-Stacked LSTM.

References

[1] *Aircraft flight will be revolutionized by biomimicry.* URL: https://www.airplanes.com/blog/aircraft-flight-will-be-revolutionized-by-biomimicry/. Accessed on July 31, 2021.

[2] Artzner et al. "Coherent Measures of Risk". In: *Mathematical Finance* (July 1999).

[3] Ashish Vaswani et al. "Attention Is All You Need". In: *31st Conference on Neural Information Processing Systems* (2017).

[4] Baillie et al. "Fractionally integrated generalized autoregressive conditional heteroskedasticity." In: *Journal of Econometrics* (1996).

[5] Gloria González-Rivera et al. "Forecasting Volatility: a Reality Check Based on Option Pricing, Utility Function, Value-at-Risk, and Predictive Likelihood". In: *International Journal of Forecasting* (2004).

[6] Hinton et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* (2014).

[7] S. Firman et al. "Estimating the Value-at-Risk for some stocks at the capital market in Indonesia based on ARMA-FIGARCH models". In: *Journal of Physics Conference Series* (2017).

[8] Rüdiger Frey Alexander J.McNeil and Paul Embrechts. *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton University Press, 2005.

[9] Saqib Aziz and Michael Dowling. "Machine Learning and AI for Risk Management". In: *Disrupting Finance* (2019).

[10] Pierre Baldi and Peter Sadowski. "Understanding Dropout". In: *Journal of Machine Learning Research* (2014).

[11] Lucas Benedikt. "Risk Management with Generative Adversarial Networks". In: *Thesis* (2019).

[12] Fischer Black and Myron Scholes. "The Pricing of Options and Corporate Liabilities". In: *The Journal of Political Economy* (June 1973).

[13] T. Bollerslev. "Generalized autoregressive conditional heteroscedasticity." In: *Journal of Econometrics* (1986).

[14] T. Bollerslev. "Modelling the Coherence in Short-Run Nominal Exchange Rates: A Multivariate Generalized Arch Model." In: *The Review of Economics and Statistics* (1990).

[15] Gwilym Box George; Jenkins. *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day., 1970.

[16] Valentina Bruno and Hyun Song Shin. *Dollars and exports: The effects of currency strength on international trade*. 27 July 2021. URL: https://voxeu.org/article/effects-currency-strength-international-trade. Accessed on August 10, 2021.

[17] Julie Steinberg By Ryan Tracy and Telis Demos. *Bank Bailouts Approach a Final Reckoning*. December 2019. URL: https://www.wsj.com/articles/ally-financial-exits-tarp-as-treasury-sells-remaining-stake-1419000430. Accessed Mai 18, 2021.

[18] C. Leigh C. Alexander. “On the Covariance Matrices Used in Value at Risk Models”. In: *Economics* (1997).

[19] Mark Carney. *Dutch Disease*. URL: https://www.bankofcanada.ca/2012/09/dutch-disease/. Accessed on August 10, 2021.

[20] Thomas D. Corrigan. “The Relationship Between Import Prices and Inflation in the United States”. In: *Sacred Heart University* (2005).

[21] Geoffrey Hinton David Rumelhart and Ronald Williams. “Learning representations for backpropagating errors”. In: *Letters to Nature* (1986).

[22] Jimmy Ba Diederik P. Kingma. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference for Learning Representations* (2015).

[23] Engle and Bollerslev. “Modeling the persistence of conditional variances.” In: *Econometric Reviews* (1986).

[24] Robert F. Engle. “Dynamic Conditional Correlation - A Simple Class of Multivariate GARCH Models.” In: *Journal of Business and Economic Statistics* (2002).

[25] Eugene F. Fama. *The Behavior of Stock-Market Prices*. The Journal of Business Vol. 38, No. 1, pp. 34-105, (Jan. 1965).

[26] Keith D. Foote. *A Brief History of Deep Learning*. February 7, 2017. URL: https://www.dataversity.net/brief-history-deep-learning/. Accessed on June 12, 2021.

[27] Aurélien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras Tensorflow: Concepts, Tools and Techniques to build Intelligent Systems*. O’Reilly, September 2019.

[28] Yarin Gal Zoubin Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: *University of Cambridge* (2016).

[29] Alex Graves and Jurgen Schmidhuber. *Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures*. Neural Networks, Volume 18:5-6, pp. 602-610, 2005.

[30] Françoise Beaufays Hasim Sak Andrew Senior. “Long Short-Term Memory based Recurrent Neural Networks Architectures for Large Vocabulary Speech Recognition”. In: *Google* (2014).

[31] Sepp Hochreiter and Jurgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation 9(8):1735-1780* (1997).

[32] Glyn A. Holton. “History of Value-at-Risk: 1922-1998”. In: *Contingency Analysis* (July 25, 2002).

[33] *How Wings Lift the Plane*. URL: https://www.grc.nasa.gov/www/k-12/UEET/StudentSite/dynamicsofflight.html. Accessed on August 05, 2021.

[34] Yoshua Bengio Ian Goodfellow and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[35] Corporate Finance Institute. *Corporate Finance Institute, Basel Accords*. URL: https://corporatefinanceinstitute.com/resources/knowledge/finance/basel-accords/. Accessed Mai 18, 2021.

[36] Bank for International Settlements. *History of the Basel Committee*. URL: https://www.bis.org/bcbs/history.htm. Accessed Mai 18, 2021.

[37] Chris Kirby Barbara Ostdiek Jeff Fleming. "The economic value of volatility timing using "realized" volatility". In: *Journal of Financial Economics* (2003).

[38] Andrew Freeman Kevin Buehler and Ron Hulme. "The Risk Revolution". In: *McKinsey Company* (September 2008).

[39] Ha Young Kim and Chang Hyun Won. "Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models". In: *Elsevier* (2018).

[40] Marc Lambrechts. *La facture de la crise bancaire: 465 millions d'euros*. 2018. URL: https://www.lecho.be/dossier/10ansdecrise/la-facture-de-la-crise-bancaire-465-millions-d-euros/10047323.html. Accessed July 25, 2021.

[41] Andreas C. Damianou Neil D. Lawrence. "Deep Gaussian Processes". In: *University of Sheffield* (2013).

[42] D.H. Leavens. *Diversification of Investments*. Trusts and Estates, 80, 469-473., 1945.

[43] Bernard A. Lietaer. "Financial management of foreign exchange: An operational technique to reduce risk." In: *The M.I.T. Press* (1971).

[44] Ralf Korn Magnus Wiese Robert Knobloch and Peter Kretschmer. "Quant GANs: Deep Generation of Financial Time Series". In: *Quantitative Finance* (2019).

[45] B. Mandelbrot. *The variation of certain speculative prices.* Journal of Business 36 (4), 394–419., 1963.

[46] H. Markowitz. "Portfolio Selection". In: *The Journal of Finance* (1952).

[47] Sarah Mattonen. *Do Woodpeckers Get Concussions?* URL: https://letstalkscience.ca/educational-resources/stem-in-context/do-woodpeckers-get-concussions. Accessed on August 05, 2021.

[48] Warren S. McCulloch and Walter Pitts. *A logical Calculus of the Ideas Immanent in Nervous Activity*. The Bulletin of Mathematical Biology 5, no. 4: 115-113, 1943.

[49] Donatien Hainaut Michel Denuit and Julien Trufin. *Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions*. Springer Actuarial, 2019.

[50] Emmanuel Mourlon-Druol. "'Trust is good, control is better': The 1974 Herstatt Bank Crisis and its Implications for International Regulatory Reform". In: *Business History* (19 Feb 2015).

[51] Daniel B Nelson. "Conditional Heteroskedasticity in Asset Returns: A New Approach". In: *Econometrica* (1991).

[52] Dr. David Nicolaus. *Basel IV – Banks should act now on capital and strategic planning*. 13 December 2017. URL: https://home.kpmg/xx/en/home/insights/2017/12/basel4-capital-and-strategic-planning-fs.html?a=b. Accessed on June 3, 2021.

[53] Li Erran Li Slawek Smyl Nikolay Laptev Jason Yosinski. "Time-series Extreme Event Forecasting with Neural Networks at Uber". In: *Uber Technologies* (2017).

[54] M. Bengtsson V. Olsbo. "Value at Risk Using Stochastic Volatility Models". In: (10th September 2003).

[55] Mack Ott. *International Capital Flows*. URL: `https://www.econlib.org/library/Enc/InternationalCapitalFlows.html`. Accessed on August 10, 2021.

[56] Nazreen P.M. and A.G. Ramakrishnan. "Using Monte Carlo dropout for non-stationary noise reduction from speech." In: (2018).

[57] Sonia Benito Pilar Abad and Carmen López. "A comprehensive review of Value at Risk methodologies". In: *The Spanish Review of Financial Economics* (2014).

[58] Jerry Coakley John C. Nankervis Rasoul Sajjad. "Markov-switching GARCH modeling of Value-at-Risk". In: *Studies in Nonlinear Dynamics Econometrics* (2008).

[59] A.D. Roy. "Safety First and the Holding of Assets." In: *Econometrica* (1952).

[60] William Sharpe. "Capital Asset Prices: A Theory of Market Equilibrium Under Conditions of Risk". In: *The Journal of Finance* (September 1964).

[61] Alfred Steinherrs. *Derivatives The Wild Beast of Finance: A Path to Effective Globalisation?* Wiley, 1998.

[62] S. Taylor. "Financial Returns Modelled by the Product of Two Stochastic Processes". In: *Economics* (1982).

[63] Turan Bali Panayiotis Theodossiou. "A conditional-SGT-VaR approach with alternative GARCH models". In: *Annals of Operations Research* (2007).

[64] M. Balıbey S. Türkyilmaz. "Value-at-Risk Analysis in the Presence of Asymmetry and Long Memory: The Case of Turkish Stock Market". In: *International Journal of Economics and Financial Issues* (2014).

[65] Ilya Sutskever Wojciech Zaremba and Oriol Vinyals. "Recurrent Neural Network Regularization". In: *Conference Paper at ICLR 2015* (2015).

[66] Jeffrey M. Wooldridge. *Introduction à l'économétrie*. 2nd Edition, DeBoeck Supérieur, 2018.

[67] Kin Keung Lai Jerome Yen Xiaoliang Chen. "A Statistical Neural Network Approach for Value-at-Risk Analysis". In: *International Joint Conference on Computational Sciences and Optimization* (2009).

[68] Menghan Wang Zhichao Du and Zhewen Xu. "On Estimation of Value-at-Risk with Recurrent Neural Network". In: *Second International Conference on Artificial Intelligence for Industries* (2019).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Faculté des sciences
Place des sciences, 2 bte L6.06.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/sc**