University of Southern Denmark
Faculty of Business and Social Science
MSc Economics - Finance

A comparison between the Black and Scholes model and Artificial Neural Networks for option pricing and delta hedging strategy

Master Thesis

Author
Caterina Stifano
10/01/1993

Supervisor
Mo Zhang

November 18, 2020

Keystrokes = 85769

Sworn statement

I hereby solemnly declare that I have personally and independently prepared this paper. All quotations in the text have been marked as such, and the paper or considerable parts of it have not previously been subject to any examination or assessment.

Caterina Stifano

Abstract

Option pricing has always been a point of discussion in the world of derivatives. Since the ground-breaking paper from Black and Scholes (BSM) was published in 1973 options have been priced using their parametric model. Recently, the popularity of non-parametric models such as artificial neural networks (ANN) has increased, reaching better pricing performances than Black and Scholes as shown by Culkin and Das [9]. This thesis compares both models, first regarding pricing and second regarding delta hedging strategy performance on European-style call options in the S&P 500 index. The results show that the ANN outperforms BSM with respect to option pricing. In contrast, analyzing the delta hedging performance no statistical difference was found between the models.

Acknowledgements

I would like to dedicate this project to my mum for investing in me. Thank my sisters, my boyfriend, and J for always supporting and believing in me.

Contents

List of Tables

List of Figures

Chapter 1

Overall description

1.1 Introduction

The financial pricing theory relies on the ground-breaking articles by Black and Scholes (1973) and Merton (1973) [5]. One of the milestones for pricing assets was the BSM's formula for European-style option pricing. The Black-Scholes formula depends deeply on:

1. The assumptions:
   - The short-term interest and the variance rate of the return on the stock are known and constant.
   - There are no dividends paid out from the stock.
   - The option is European and can only be exercised at maturity time.
   - There are no transaction costs in buying or selling the option.
2. The knowledge of the stochastic process:

$$dS_t = \mu S_t d_t + \sigma S_t dW_t \tag{1.1}$$

   Which determines the price dynamics of the underlying asset S_t of the option as a continuous time diffusion with a constant drift μ constant volatility σ and W_t denoting a Wiener process.

As described in Equation 1.1 the volatility has an important role in the BSM model. Therefore, is important to compare different volatility estimation models to guarantee the lowest mispricing in the BSM pricing prediction.

Even though the dependencies of the BSM formula are typical examples of traditional parametric models, this is considered one of the best for determining the price of options. However, a wrongly specified process could lead to pricing errors. Introducing data-driven, non-parametric methods such as neural networks, the pricing error could be reduced. Since any economic variable may influence the price of the option, up to a certain degree, it

is interesting analyze a data-driven model that can to learn the dynamics behind option pricing and minimize the assumptions.

However, as Hutchinson et al (1994) [22] explain, a good pricing prediction is not sufficient to support the practical relevance of the non-parametric model. Since the BSM is a no-arbitrage-based pricing formula, is important to prove the ability of the model in replicating the option through a dynamic hedging strategy such as delta hedging.

Several previous pieces of literature have compared the performances of Black and Scholes and artificial neural network-based (ANN) methods in option pricing. The majority of the study illustrates that, generally, the artificial neural networks models can predict option prices from real market data with high accuracy. However, most of the authors performed their researches with data sets containing data up to the 2000s. Since then, changes in the economy have been registered, therefore, this thesis aims to analyze the performance of data-driven models of data set with greater overall variance. Furthermore, the increase of the level of available computational resources makes it easier to build complex data-driven models such as neural networks, allowing the researchers to benefit from a greater computational power.

1.2 Goals

This thesis provides a comparison of the pricing prediction and the hedging performance between a deep neural network model and the Black-Scholes formula. Therefore, this project aims to test if the results of the previous literature can be replicated using more recent data. Besides, a comparison of estimated volatility performances will be presented. The main hypotheses state that the deep neural network outperforms the Black-Scholes model in pricing performance and delta hedging performance.

1.3 Report structure

The structure of the thesis is divided as follows:

- The Theoretical Background is introduced for option pricing, delta hedging, volatility estimation, and neural networks.
- In the Method section, the data analysis, the numerical simulations, and tests are developed and presented.
- The chapter Results provides an analysis of the empirical results.
- The last chapter is the Conclusion and Further work discuss the results and suggests possible improvements.

1.4 Literature review

The previous literature illustrates that, generally, the artificial neural network models can learn how to predict option prices from real market data with high accuracy. Hutchinson et al (1994) [22] were the pioneer to propose the use of a neural network to pricing options in a non-parametric way and therefore to question the use of parametric models. The American-style S&P 500 index future is used from the authors to study the pricing performance and the hedging performance. Their researches are implemented in three different neural networks such as:

1. Radial Basis Function (RBF)
2. Projection Pursuit Approach (PP)
3. Multi-Layer-Perceptron (MLP) with one hidden layer of four neurons and the two input variables.

Their approach is based on using two variables as input such as the underlying asset price divided by the strike price S_t/K, and the time to maturity $T-t$ of each contract. In the paper the findings regarding the pricing predictions are remarkable because the network used outperforms the BSM model in option pricing. Since the neural network was performed directly on discrete data the authors expected it to perform better than the BSM also when a hedge strategy was implemented, because BSM has the non - realistic assumption of continuous time hedging. Concluding with the results that the neural network models outperform the Black-Scholes formula also in delta-hedging.

Subsequently, many other academic researchers followed the lead of the non-parametric approach obtaining similar results to the previous authors. On the other hand, other researchers, providing some additional insights and improvement at the original paper, found the BSM to be superior only in specific cases.

Anders et. al. (1998) [3] implement two neural networks to learn the dynamics of the pricing of the DAX index. The two implemented models are one MLP and a combination between an MLP and the BSM model. Their neural network is similar to Hutchinson's with one hidden layer and three hidden neurons. In contrast, from Hutchinson, the authors introduce more input variables such as the estimated volatility σ and the risk-free rate r. The results they obtain from the pricing prediction show that the neural network outperform the BSM with significantly smaller pricing errors. Regarding the delta hedging strategy, the authors found that the BSM performed better especially for in-the-money options which were suspected to have fewer trading volumes, therefore with less observations.

Hanke (1999) [18] concluded that after optimizing the volatility and risk-free rate parameters the Black-Scholes performance became superior.

Amilon (2003) [2] used the European style calls on the OMX index with both implicit and historical volatilities to predict the call price and perform the hedging strategy. The neural network chosen is the MLP with one hidden layer and a fluctuating number of hidden

neurons. Differently from the previous literature the author uses more input variables such as underlying asset price S_t, time-to-maturity in trading and calendar days, strike price K, risk-free interest rate r_f, lagged asset price values, volatility estimated from the underlying asset's return. In this case the author defines the output of the model to be the bid and the ask price of the index, allowing the neural network to intrinsically learn the dynamics behind the spread between the bid and the ask prices. Amilon concludes that the neural network was superior in both pricing and hedging.

Malliaris and Salchenberger (1993) [29] compared the pricing performances with the American-style S&P 100 call options, concluding that the Black-Scholes was superior for in-the-money options but contrary, for the out-the-money options, the ANN was superior.

Moreover, Bennell and Sutcliffe (2004) [4] used FTSE100 call options and compared the pricing accuracy between the dividend-adjusted Black-Scholes formula and the ANN. The neural network used is similar to Amilon's, one hidden layer and a varying number of hidden neurons. The authors use analogous inputs to the neural network as they use for the BSM, such as the Merton dividend adjustment, risk-free rate and volatility estimated from the market data, and the daily volume of each option contract. They conclude their study finding that the MLP performs significantly better than the BSM for the out-the-money options, but on the other hand the BSM outperforms the MPL for the in-the-money options.

Yao et. al. (2000) [36] choose to use in their study the same architecture of Hutchinson et al. However, they argue how the introduction of either the historical volatility or the implied volatility as input variable could restrict the ability of the neural network to learning the volatility dynamics form the data. Subsequently, they found that the ANN performs better, however, BSM predictions are better for at-the-money options. Differently from the previous literature Yao et. al. did not perform the delta hedging analysis.

Overall, it can be seen from the discussed literature that the MLP usually performs better than BSM for European style out-the-money (OTM) options. On the other hand, some results imply that the ANN is superior, while some other conclude that in the case of the in-the-money (ITM) options the results are not as clear as the OTM options. A clear similarity between the literature can be found in the neural network structure; most of the authors choose one hidden layer and a low amount of neuron per layer. However, some researchers implemented more complex networks which provides similar results.

Chapter 2

Theoretical background

This section introduces the basic features of the option contracts, followed by the widely used Black & Scholes pricing formula. Then, it covers the main theory behind volatility forecasting methods such as historical volatility and GARCH. The last section introduces the concept of artificial neural networks and their fundamentals.

2.1 Option Pricing

Option fundamentals

An option is a financial market derivative, as a contract, it gives its holder the right (but not the obligation) to buy or sell the related underlying asset for a given fixed price which is set when the option contract is purchased. As explained in Hull (2003) [21] the options can be distinguished based on their characteristics.

Based on the exercise time, options can be distinguished as:

- European options that can only be exercised at maturity time or expiration date.
- American options that can be exercised any time prior to maturity.

There are two basic types of options:

- Call option, which gives the owner the right to buy the underlying asset for a specific price known as strike price,
- Put option, which gives the owner the right to sell the underlying asset at a given strike price.

Based on the moneyness, options can be categorized as:

- In-the-money (ITM), which means that the ratio between underlying asset price and strike price is greater than 0. This implies, in the case of a call option, that the derivative has a positive intrinsic value that would be generated as a profit if the option is immediately exercised.
- At-the-money (ATM) if the underlying asset price is equivalent to the strike price. This represents the break-even point between having positive and negative intrinsic value.
- Out-the-money (OTM) when the ratio between the underlying asset price and strike price is lower than 0. This implies, in the case of a call option, that the derivative has negative intrinsic value therefore it would give a negative profit if exercised.

One of the properties that can be observed is that the price of a call option decreases as the strike price increases and the opposite is true for the put options. Moreover, the closer the expiration date the more valuable the options become. Most of the options traded on exchanges are American however, European options are generally easier to analyze. Therefore, some of the properties of American options are deduced from the European options. The largest exchange in the world for trading stock options is the Chicago Board Options Exchange (CBOE).

In the following sections, having a long position means buying options and having a short position means selling options (selling is also known as writing the option).

Black and Scholes

As stated in the introduction the most popular formula for option pricing is the Black-Scholes formula, which is the result of the following Black-Scholes equation [[5]]:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{2.1}$$

Where V is the price of the derivative, σ is the standard deviation of the stock returns, S is the price of the underlying asset, r is the continuously compounded interest rate, and t is the time. The derivative obtained when this equation is solved depends on the boundary conditions that are used, which, in the case of European call option, is

$$f = max(S - K, 0) \qquad \text{when} \qquad t = T \tag{2.2}$$

As explained in Hull (2003) [21], the portfolio used in the derivation of equation 2.1 is not permanently riskless. Therefore, is necessary to minimize the risk on the options market by implementing the delta hedging strategy which objective is to reduce the risk associated with the price movements in the underlying asset.

There are two ways of deriving the Black-Scholes-Merton formula:

1. By solving the differential equation 2.1 subject to the boundary condition 2.2.

2. By using the risk-neutral valuation approach.

The first method leads to the following Black-Scholes[[5]] formulas, obtained for the European Call and Put option respectively:

$$C(S,t) = N(d_1)S - N(d_2)Xe^{-rt} \tag{2.3}$$

$$P(S,t) = N(-d_2)Xe^{-rt} - N(-d_1)S \tag{2.4}$$

Where C is the call option price, P is the put option price, $N(\cdot)$ is the cumulative distribution function of the standard normal distribution, t is the time to maturity, X is the strike price, r is the continuously compounded interest rate, S is the price of the underlying asset and:

$$d_1 = \frac{1}{\sigma\sqrt{t}}((ln\frac{S}{X}) + (r + \frac{\sigma^2}{2}t)) \tag{2.5}$$

$$d_2 = d_1 - \sigma\sqrt{t} \tag{2.6}$$

This model, due to its simplicity is widely used, however some of the assumptions presented in the introduction are not realistic. The model underestimates the possibility of big changes in the market movements, it assumes the instantaneous and free execution of trading which leads to liquidity risk, it assumes that the process is stationary which yields a volatility risk and it assumes continuous time and continuous trading which is generally not satisfied.

2.2 Delta hedging

The delta hedging strategy aims to reduce the directional risk associated with the price movements of the underlying asset [21]. The delta Δ of an option is the rate of change of the option price with respect to the price of the underlying asset. Therefore, is formally defined as the partial derivative of the option price with respect to the underlying asset price:

$$\Delta = \frac{\partial C_t}{\partial S_t} \tag{2.7}$$

A position with $\Delta = 0$ is referred to as delta neutral. It is important to mention that, since the delta of an option does not remain constant, the position remain delta neutral for a short period of time. Therefore, the delta hedged portfolio needs to be periodically adjusted (rebalanced). The delta hedging strategy adopted in this thesis follows the method used by Hutchinson et al.(1994) [22]. The main idea of the strategy is to set up a replicating portfolio V_t that offsets the risk of the option position:

$$V_t = V_t(S) + V_t(B) + V_t(C) \tag{2.8}$$

Where $V_t(S)$ is the value of the underlying asset position, $V_t(B)$ is the value of a bond position used to finance the position in the underlying asset and $V_t(C)$ is the value of the option position held in the portfolio at time t. The replicating portfolio V_T at expiration of the option intends to have zero value, by going short in one call option $V_t(C)$, going long in the underlying asset $V_t(S)$ and refinancing for the long position in the underlying asset with the bond $V_t(B)$.

$$V_T = V_T(S) + V_T(B) + V_T(C) \tag{2.9}$$

This procedure of periodically adjusting the hedge is also known as dynamic hedging. As stated in the Introduction, Black and Scholes assume continuous time hedging, however, in reality, time can only be measured discretely. Therefore, the combined value of the hedging portfolio and the option position is unlikely to be zero due to the tracking error between the two position. This tracking error can be positive or negative, reflecting either a profit or a loss that is associated with the taken hedging position. As the final performance of the hedging strategy analysis are used, the present value of the expected absolute tracking error:

$$\epsilon = e^{-rT} E[|V(T)|] \tag{2.10}$$

And the prediction error:

$$\eta = e^{-rT} \sqrt{E^2 \left[V(T)\right] + Var \left[V(T)\right]} \tag{2.11}$$

For an European call option, with a non-dividend-paying stock, it can be shown that:

$$\Delta(call) = N(d_1) \tag{2.12}$$

Where d_1 is defined as in equation 2.5, and $N(\cdot)$ is the cumulative distribution function of the standard normal distribution. The above formula gives the delta of a long position in one call option.

2.3 Volatility estimation

2.3.1 Historical volatility

Historical volatility is used by options traders as a proxy for future volatility in evaluating options. This method relies on the usage of historical stock price data at a given time frequency. It is obtained as the standard deviation of the underlying stock's return over some period.

$$r_t = ln(\frac{S_t}{S_{t-1}}) \tag{2.13}$$

Where r_t is the return of the stocks between the period t and $t-1$. This return is often calculated in the logarithm scale from the underlying stock price S_t in the period t. The historical volatility is obtained by the standard statistical formula for the second

statistical moment, the variance, and taking its square root:

$$\sigma_{HV} = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T} r_t - \overline{r}} \tag{2.14}$$

Where HV is the historical volatility estimator, T is the number of periods over which the volatility is measured and $\overline{r}$ is the mean value of the stock returns r_t. As it can be see, classically, the historical volatility is computed as the standard deviation of daily returns within a certain period. However, it is unrealistic to assume that the volatility of asset returns remains constant during a long period; therefore the volatility estimated with the classical estimator is essentially the average volatility over the specified period [35]. In cases where the historical volatility is high, it follows from the previous unstable period with a greater magnitude in the price returns movements. On the other hand, in periods of stable trading or low activity in the market might be characterized by the low value of the historical volatility estimator. It is also possible to compare two historical volatilities over different time periods which yields the information whether the volatility recently changed positively or negatively.

2.3.2 GARCH model

As stated in the previous paragraph, assuming that the volatility of assets returns remains constant during a long period is not realistic, since prices do not follow exactly the constant volatility log-normal diffusion model. Therefore, problems are caused due to the serial correlation in returns, time varying volatility, a noisy estimate of the mean, and also the fact that returns are not normally distributed. [13]

On one hand, the basic model which assumes time-invariant volatility could be specified as follows

$$r_t = E(r_t) + \varepsilon_t \tag{2.15}$$

s.t.

$$E(\varepsilon) = 0 \tag{2.16}$$

$$Var(\varepsilon) = \sigma_t^2 \tag{2.17}$$

where in the simplest model the variance would be assumed to be constant

$$\sigma_t^2 = C \tag{2.18}$$

On the other hand, the time varying volatility can be handled by the models which involve the volatility prediction over time such as AutoRegressive Conditional Heteroskedasticity (ARCH) used by Engle (1982) [12]. In this case, the variance in period t is given by:

$$\sigma_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 \tag{2.19}$$

Where the conditional variance σ_t^2 is dependent on the realized value ε_{t-1}^2. If the realized value of ε_{t-1}^2 is large, the conditional variance in t will be large as well. In Equation 2.19, the conditional variance is a first-order AutoRegressive Conditionally Heteroskedastic

process denoted by ARCH(1). This model reflects one lag of the errors in the estimation of the variance [11]. Adding more lags we would obtain ARCH(q) model with $q \in \mathbb{N}$ lagged residual terms. Therefore, the model is given by:

$$\sigma_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + ... + \alpha_q \varepsilon_{t-q}^2 \tag{2.20}$$

Bollerslev (1986) extended Engle's original work by developing a technique that allows the conditional variance to be an ARMA process. As explained by Enders (2008) [11], the Generalized ARCH(p, q), also know as GARCH(p, q), allows for both autoregressive and moving average components in the heteroskedastic variance. This generalization of the GARCH model is given by:

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + ... + \alpha_q \varepsilon_{t-q}^2 + \beta_1 \sigma_{t-1}^2 + ... + \beta_p \sigma_{t-p}^2 \tag{2.21}$$

Where ω, β_i and α_j for all i and j are regression coefficients.

The most used GARCH model is the GARCH(1,1) and is given by

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \tag{2.22}$$

where ω, β_i and α_j are the estimation parameter and ε_{t-1} is the error term. On one hand, the GARCH model with its simplicity captures the kind of time variation that it seems plausible for variances. On the other hand, the model is not easily adjustable and restricts the impact of a shock to be independent of its sign [13].

2.4 Artificial Neural Network

Artificial Neural Networks or ANN is an information processing paradigm that is inspired by the way the human brain processes information. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve a specific problem.

Neurons

The brain is made of approximately 100 billion nerve cells, called neurons. These cells are responsible for the collection, processing, and broadcasting of electrical signals. Brain's information processing capacity comes from networks of such neurons.

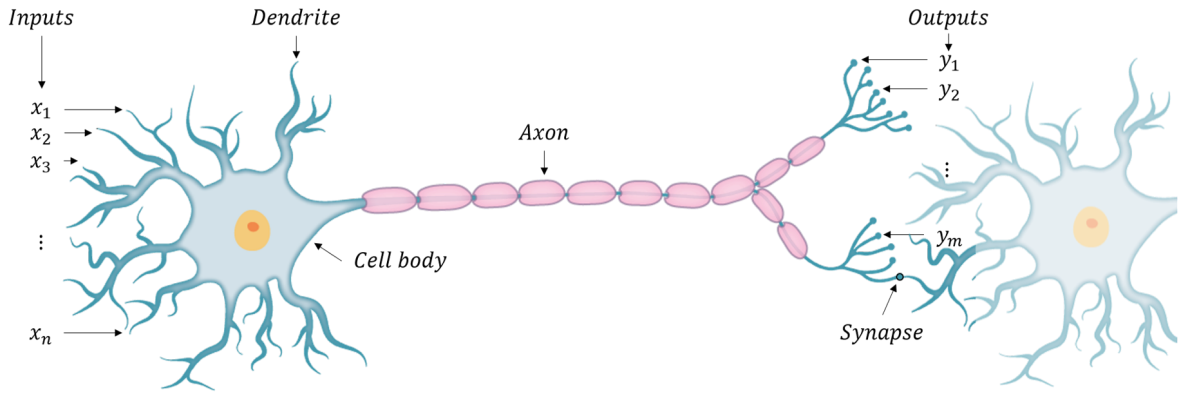


Figure 2.1: Biological neuron

A neuron consist of the following main parts [Figure 2.1]:

- Cell body: It contains the nucleus of the cell and it carries out the biochemical transformations necessary for the neuron's life.
- Dendrites: This root-like structure branches out from the cell body forming a tree structure. Each dendrite connects to other neurons in order to provide an input for the cell.
- Axon: It is a long, thin, tubular structure in charge of transmitting the signal generated by the neuron.
- Synapse: The connection between neurons forms a complex spatial arrangement. When the axon has reached its final destination, it starts branching into highly complex and specialized structures called synapses. These structures are responsible for the connection between cells in order to transmit the signal from one neuron to the following.

To sum up, the neuron activity starts when the dendrites receive an incoming signal through the synapses of other neurons. The cell body processes the input signals over time and converts the processed value in order to generate an output. Finally, this signal is sent to the other neurons via the axon and the synapses.

The first Neural network architectures were developed by McCulloch and Pitts (1943) [30], where the authors presented a model of how real biological neurons are linked and work together in the brains of mammals. Moreover, Hebb (1949) [20] proposed a theory on how biological neurons in the brain adapt during the learning process.

The simplest form of a neural network is composed of a single layer with one neuron and it is called Perceptron, presented by Rosenblatt (1958) [32] which is inspired by a biological neuron. The Perceptron structure is represented in Figure 2.2.

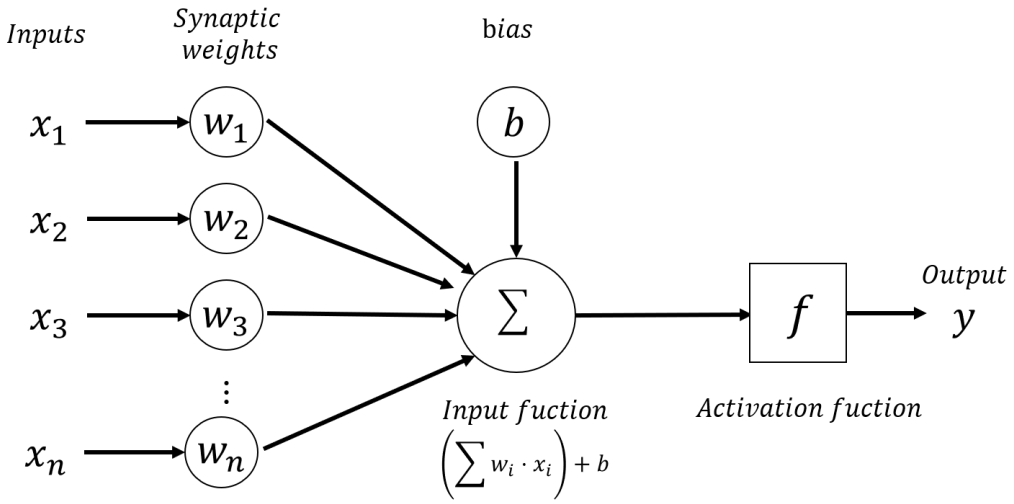


Figure 2.2: Perceptron

Where

- The inputs x_1, x_2, x_3, x_n represent independent variables from one single observation.
- The synaptic weights w_1, w_2, w_3, w_n, represent the strength of each input. These weights are multiplied by their corresponding input.
- The input function aggregates the information of the inputs, the most common is the summation. The last parameter is the bias, b, which allows to add an adjustable offset.
- Activation function generates the output y based on the results provided by the input function.

Based on the perceptron, Ivakhnenko (1970) [23] and Fukushima (1980)[14] developed a more complex model called the Multilayer Perceptron (MLP).

Multilayer Perceptron

The multilayer perceptron consists of multiple layers of neurons and is also known as feedforward network and is said to be fully-connected if all the neurons are connected to each other.

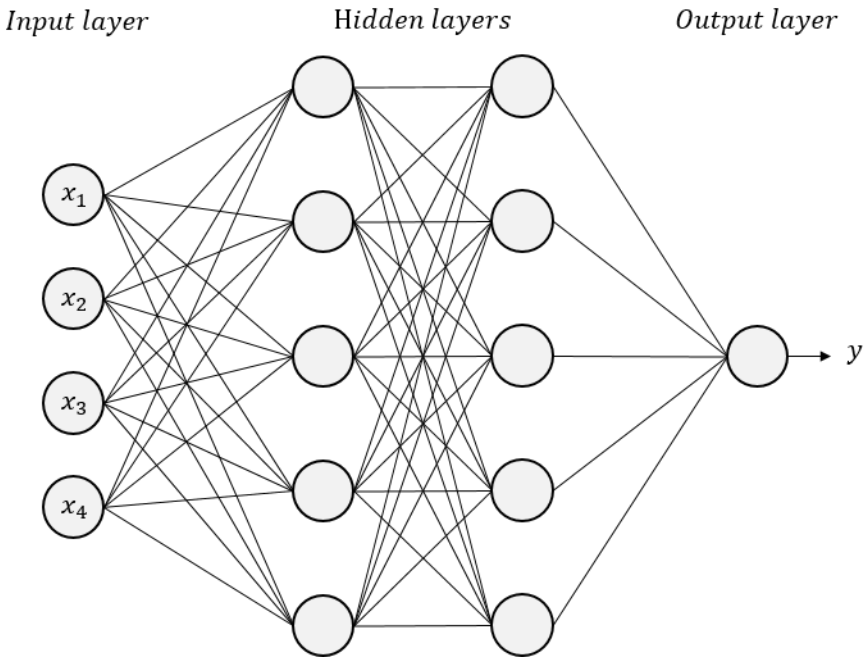


Figure 2.3: Multilayer perceptron diagram

The layers are divided in:

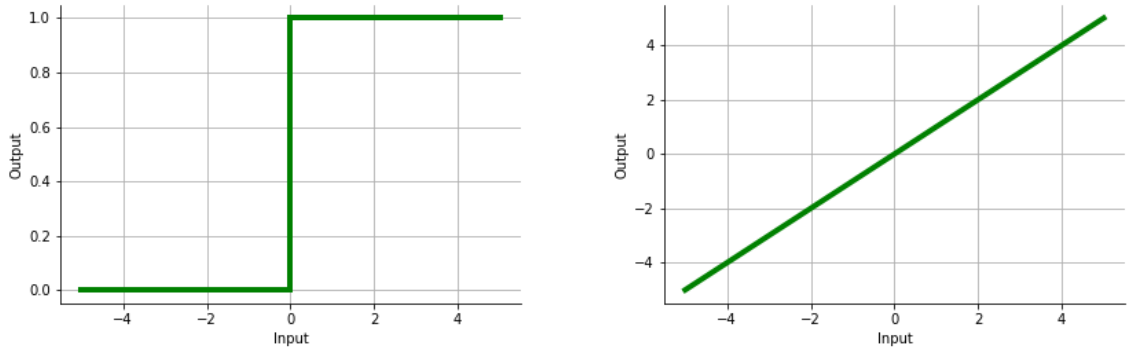
- First layer which is also known as the Input layer. This consists of the input vector. All the elements of the input layer are connected to each neuron in the first hidden layer. Moreover, the input of each neuron in the first hidden layer is a weighted sum of all the inputs plus the bias term.
- Last layer which consist of the Output layer. The outputs of all the neurons in the last hidden layer are connected to the neurons in the output layer.
- Layers in between which are the hidden layers. As presented by LeCun et al. (2015) [28] an arbitrarily number of hidden layers and neurons can exist between the input and output layers in the Multilayer Perceptron.

Activation Functions

The activation function is a crucial part of the neurons because it is in charge of controlling the output of the neuron. The activation function takes as input the results of the input function [2.2] and transforms it as the output of the neuron.

The original activation function [Figure 2.4a] used in the perceptron was the binary step function which is threshold-based. This means that the neuron sends the signal to the

next layer depending on if the input value is above or below a certain threshold [25]. The disadvantage of this activation function is that the output is constraint only to two possible values, therefore it cannot be used to solve classification problems with multiple categories.



(a) Binary Step Function (b) Linear Function

Figure 2.4: Activation functions

The simplest activation function [28] [34] is the linear function [Figure 2.4b], where the inputs are linearly transformed in order to compute the output. An MLP which uses linear activation functions is equivalent to a linear regression model. Therefore, this model would have limited power and reduced ability to handle complex data. On one hand, the linear activation function is better then the step function since allows multiple outputs.

On the other hand, the linear activation function has two problems:

1. The impossibility to use backpropagation algorithm to train the model. In fact, the derivative of a linear function is constant therefore, is not possible to understand which weights in the input neurons can provide a better prediction.
2. A linear combination of linear functions is still a linear function[27] . Building a network with multiple layers, which uses the linear activation function, the output of the network will be a linear function of the input. This means that the multilayer network would be equivalent to a network with a single layer.

More recent neural network models tend to use non-linear activation functions since they allow the ANN to generate better and more complex mappings between inputs and outputs of the network. This is due to the fact that the non-linear function solves the problem of the linear activation function. Therefore,

1. It is possible to apply backpropagation since their derivative function is related to the inputs.
2. Multiple hidden layers of neurons are allowed and this leads to the model to learn complex data sets with high levels of accuracy.

The non-linear activation function used in this project are:

ReLU (Rectified Linear Unit)

This function, first introduced by Hahnloser et al. (2000) [17], is linear for values greater than zero and for the negative value the output is always zero. This means that the ReLu activation function has some of a linear activation function when training a neural network using backpropagation, but is not exactly the like the linear activation function since the derivative is not constant.

$$ReLU(x) = \begin{cases} x \text{ if } x > 0 \\ 0 \text{ if } x \leq 0 \end{cases} \tag{2.23}$$

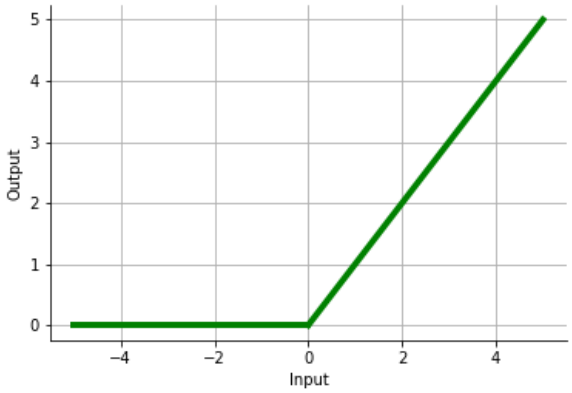


Figure 2.5: Activation function: ReLu

On the other hand, the disadvantage of the ReLu is that when the input approaches zero the network cannot perform backpropagation, and therefore cannot learn. This problem is also known as a dead ReLu problem [1].

Leaky ReLU

The Leaky ReLu is similar to the ReLu function with the difference that in the negative part has a small positive slope which helps the learning process because it solves the deal ReLu problem.

$$ReLU(x) = \begin{cases} x \text{ if } x > 0 \\ \alpha x \text{ if } x \leq 0 \end{cases} \tag{2.24}$$

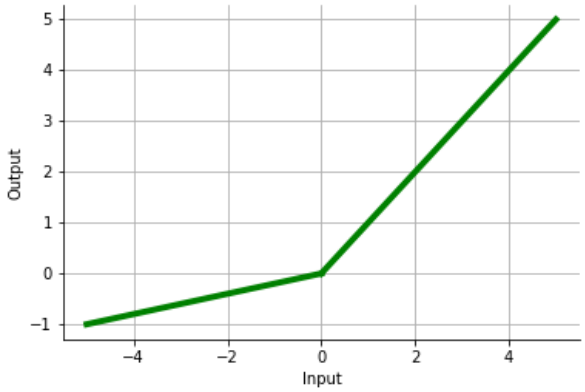


Figure 2.6: Activation function: Leaky ReLu

As explained from the equation, we see that any x-value maps to the same y-value, if the x-value is greater than 0. But if the x-value is less than 0, we have a coefficient of alpha, which is 0.2. That means, if the input value x is 5, the output value it maps to is 1.

ELU (Exponential Linear Unit)

The ELU activation function is similar to the Leaky ReLu with the difference that the negative part, instead of being a linear function with a positive slope, is an exponential function.

$$ReLU(x) = \{ \begin{matrix} x \text{ if } x > 0 \\ \alpha(e^x - 1) \text{ if } x \leq 0 \end{matrix} \tag{2.25}$$

Therefore, in this case if the input value is less than 0 the derivative is not constant so the model has a better learning. The ELU is more computationally expensive than the ReLu.

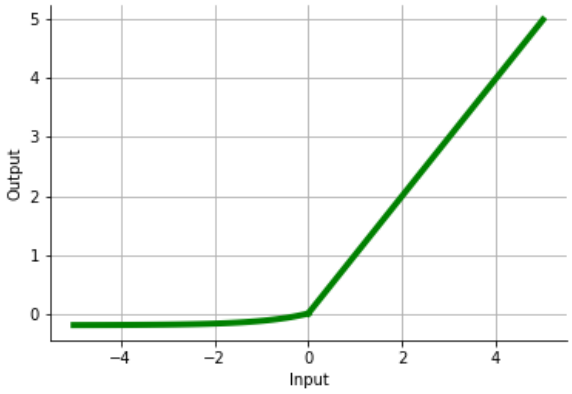


Figure 2.7: Activation function: ELu

Exponential

One characteristic of the previous activation functions is that the output has negative values when the input is smaller than 0. In some cases, it is interesting to have an output with range $[0, \infty]$.

$$EXP(x) = e^x \tag{2.26}$$

Although any arbitrary function could be used in this situation, the exponential function provides an output that matches the desired range while having an easy to compute derivation.

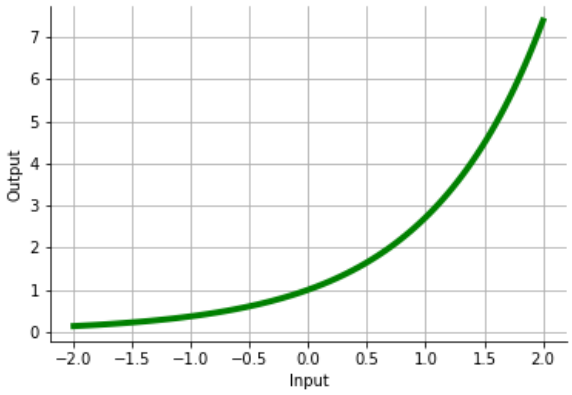


Figure 2.8: Activation function: EXP

Backpropagation

The Backpropagation algorithm is proposed by Rumelhart et al. (1986) [33]. The learning of the neural network depends on adjusting all the parameters in the model in order to extract valuable information from the input data. The simplest method would be using a brute force method to find the best weights for the network. However, for more complex neural network models this approach would be computationally expensive due to a large number of parameters to be adjusted. Currently, most of the ANN's use the Backpropagation algorithm for the learning phase because of its efficiency.

The workflow of the backpropagation algorithm can be described as follow:

1. The first step consist of building the neural network. This means, defining the number of layers, the number of neurons, the activation function for each of the layers as well as the input and output parameters.

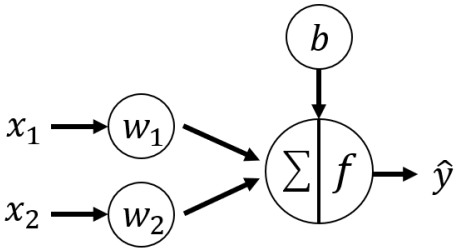


Figure 2.9: Backpropagation algorithm: building ANN

2. Once all the parameters of the ANN are defined it is necessary to initialize the weights and the biases. This initialization is crucial since a wrong initialization could affect negatively the performance of the model. In order to avoid this the values of the weights and biases are randomly assigned. These values are usually in a range between 0 and 1.

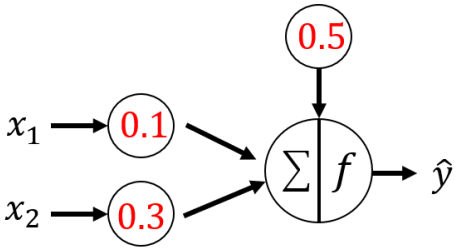


Figure 2.10: Backpropagation algorithm: weight initialization

3. After initializing the weight the learning process can begin. This process starts with selecting the input data which is then fed to the network and forwarded through all the neurons and their connections.

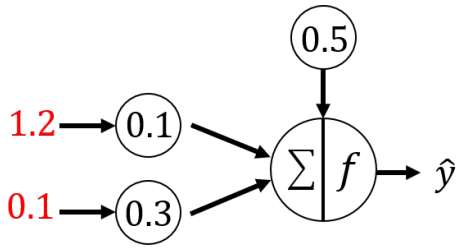


Figure 2.11: Backpropagation algorithm: forward pass

4. As a result of the forward pass, the model will output the predicted value $\hat{y}$.

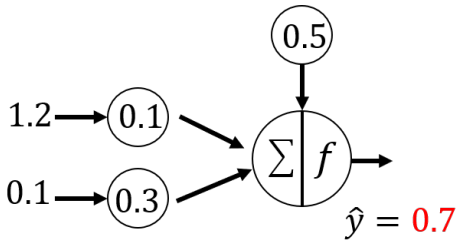


Figure 2.12: Backpropagation algorithm: initial prediction

5. During the learning phase for each set of inputs, the expected output y is available. This makes it possible to use an error function to compute the difference between the predicted and actual value. This error is also known as "loss" and it is used to adjust the model parameters.

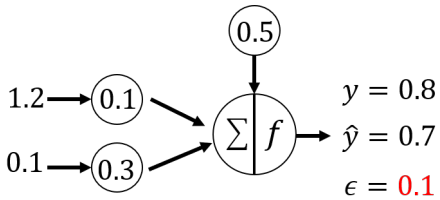


Figure 2.13: Backpropagation algorithm

6. In order to adjust the weights and biases of the network first, it is necessary to identify the contribution of each parameter regarding the error. To do so, an optimizer is used and it computes the partial derivative of the output with respect to each neuron's parameter. The derivative is then used to calculate the necessary adjustment. Among the common optimizers, there are Stochastic Gradient Descent and ADAM.

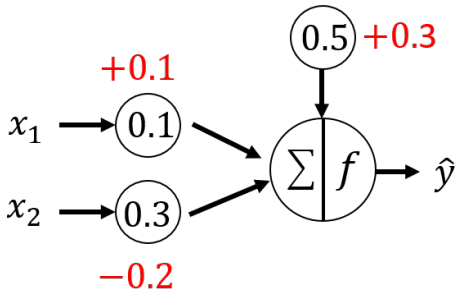


Figure 2.14: Backpropagation algorithm: Backpropagation

7. In the final step, the weights and biases are updated and the next forward pass takes place. Although the parameters could be updated for every new input this is usually not practical because with large data set this would be computationally expensive. A common practice is to group the samples in batches and only update the weights once a batch had been processed. The size of the batches, together with the number of epochs (number of times that the data set is processed) are important hyperparameters which need to be tuned to ensure the best performance of the model.

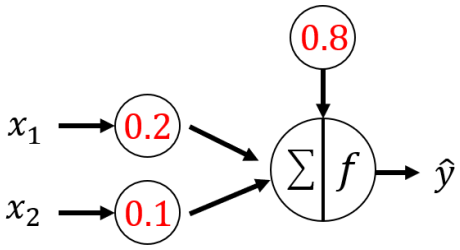


Figure 2.15: Backpropagation algorithm: Weights and Biases updated

Chapter 3

Method

In this chapter, the methods used to obtain the results of the project are presented and explained. The first section is the research environment, here the different programming languages and libraries are introduced. The data pre-processing is the second part of this section, in which the important process of data-cleaning is explained and subsequently, the data set is analyzed and presented. The third section describes the Neural Network architecture implemented. The most important section is the last one where the performance metrics are described. The first performance metric consists of the volatility estimation, in this section the historical volatility is compared to the GARCH. The second performance metric consists of the pricing performance, here the pricing evaluating method is presented. The last performance metric describes the method implemented for the hedging strategy, the evaluating methods, and the ANN delta.

3.1 Research environment

In this section, the different programming languages and libraries used for the implementation of the project are introduced and described.

The data pre-processing is implemented in R which is a language and environment for statistical computing and graphics. This language provides a wide range of statistical and graphical techniques and is highly extensible. R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form.

The implementation of the GARCH model, the Neural Network, and the Hedging Strategy are written in Python. This language provides a wide variety of useful libraries, among them TensorFlow and Keras, for working with deep learning models.

TensorFlow & Keras

TensorFlow is an end-to-end open-source platform for machine learning, the documentation is available online and besides, it can be configured to use the computer GPU to make the training and testing processes faster. For this project, TensorFlow-GPU is configured together with CUDA and cuDNN using an NVIDIA GeForce GTX 1050 Ti.

The deep learning models presented in this project are build using Keras, which provides a high-level API for TensorFlow. Keras is widely used for fast prototyping, production, and research. It has three key advantages which make it a popular library:

1. It has a user-friendly interface, which facilitates usage.
2. Keras models are constructed by connecting configurable building blocks together, therefore its modularity.
3. Keras is easy to extend by creating new elements such as layers, loss functions, or metrics.

3.2 Data pre-processing

This section presents the most important phase of a machine learning project: data cleaning. This process aims to detect and correct (or remove) corrupt or inaccurate observations from the data set. Analyzing data that has not been carefully screened for such problems can produce misleading results. Afterward, the nature and the features of the data are presented.

In the current section, the description of the data includes European style call and put options. However, from section 3.4 only the European style call options are used, due to the computational cost that processing the whole data set would generate. Nevertheless, the procedure described in the project can be applied also for the European style put options (considering that, in the case of put option, the Black and Scholes pricing formula would be Equation 2.4).

The data used in the empirical analysis is collected from OptionMetrics, and they consist of the daily closing quotes of the S&P 500 stock market index and daily closing quotes of the S&P 500 stock index options. The S&P 500 or Standard & Poor's 500 Index is a market-capitalization-weighted index of the 500 largest U.S. publicly traded companies. The observations start from January 1996 and end in August 2014. The continuously compounded return of the 90-day US treasury bill is collected from FRED (fred.stlouisfed.org) and is used to approximate the risk-free rate.

The full data set consists of 4 352 445 observations. The data cleaning process follows the approach taken by Anders et al. (1996) [3].

Therefore, the observations excluded are:

1. The call option traded below 10 basis points:

$$C_t \leq 0.1 \tag{3.1}$$

   Where C_t is the call price. This criterion aims to exclude options with very low prices. According to Anders et al.(1996) [3] options tend to be traded at integer values, leading to relatively high deviations between observed and theoretical option prices when the option values are very low.

2. The option with maturity $(T-t)$ less than 15 days and more than 2 years:

$$T - t \geq 2 \quad or \quad T - t \leq \frac{15}{252} \tag{3.2}$$

   Where 252 is the number of trading days. This criterion excludes options with both very low or very high time to maturity since the small/high time-value, together with the integer pricing behavior, would lead to deviations between theoretical and real options.

3. The options which are either deep in-the-money (ITM) or deep out-the-money (OTM):

$$\frac{S_t}{K} < 0.5 \quad or \quad \frac{S_t}{K} > 1.55 \tag{3.3}$$

   Where $\frac{S_t}{K}$ is the moneyness of the option, S_t is the underlying asset price and K is the strike price. This criterion excludes both deep ITM and deep OTM options since these are traded at their intrinsic value and have very low trading volume.

4. The options that violate the lower boundary condition:

$$C_t \geq S_t - Ke^{-r(T-t)} \tag{3.4}$$

   According to Hull (2003) [21] the no-arbitrage condition is unbreakable. Therefore, this last criterion excludes options whose prices are not consistent with the no-arbitrage condition.

After removing all the non-representative observations, the data set consists of 1 015 063 samples which is a reasonably large number to train the neural network examined in this thesis. The following figures summarize the characteristic of the data set.

In Figure 3.1 is shown the closing price of the S&P 500 index in the period between 1996 and 2014. The range of the underlying price per one share of the S&P500 index is between 598,48$ and 2003,37$. Two main valleys can be observed. The first one can be associated with the dot-com bubble (1999-2003) and the second valley can be associated with the financial crisis (2007-2009).

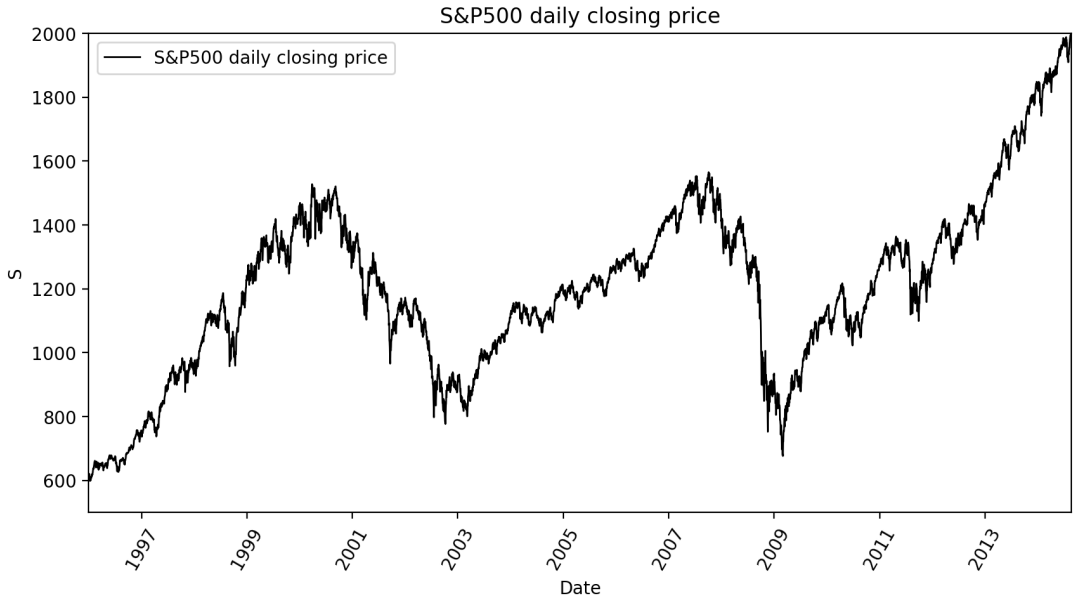


Figure 3.1: S&P500 daily closing prices between 1996-2014

From Figure 3.2 it can be seen that the returns are generally oscillating around the mean which is close to 0. The high dispersion of returns, which is represented by the spikes.

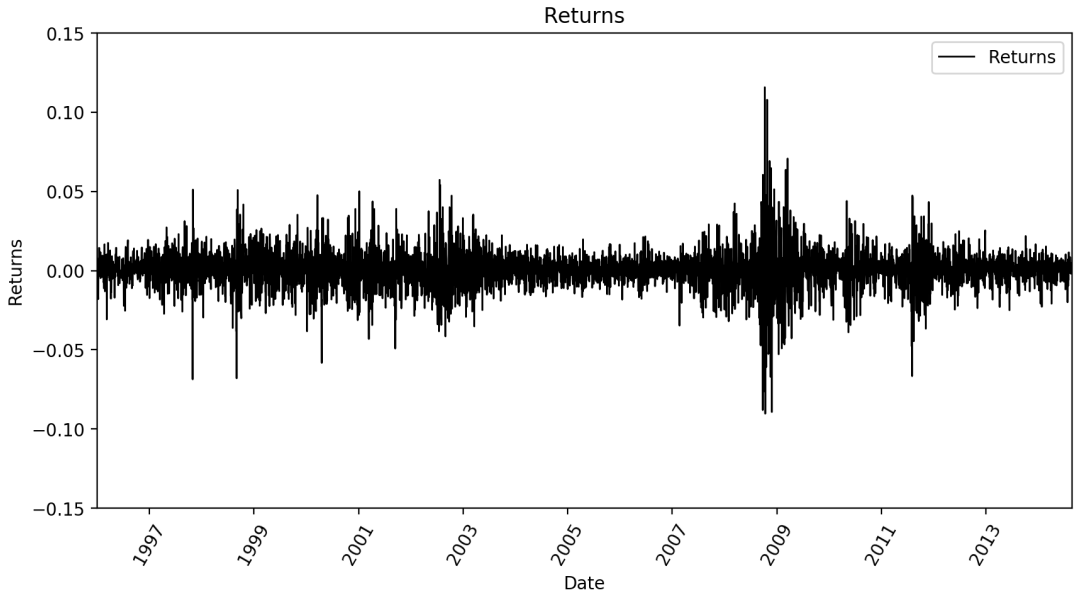


Figure 3.2: Returns on S&P500 index between 1996-2014

The historical volatility is computed from the value of moving averages over five distinct periods and they are estimated according to Formula 2.14. For visualization purposes Figure 3.3 displays only the 5 (σ_{MA5}) , 60 (σ_{MA60}), and 120 (σ_{MA120}) window moving averages over the returns.

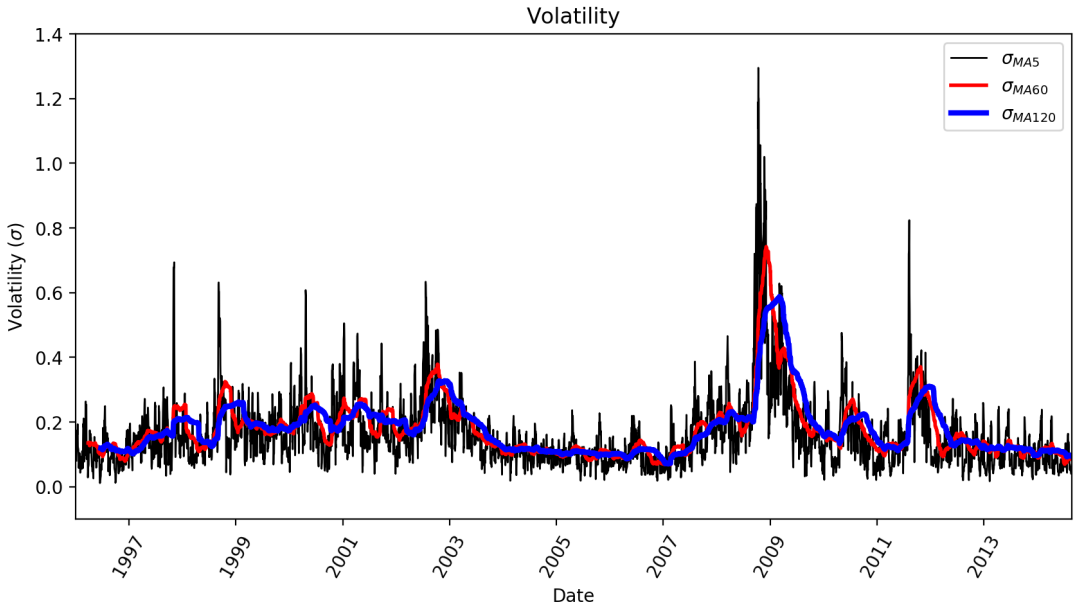


Figure 3.3: Historical Volatility of S&P500 index between 1996-2014

The risk-free rate is estimated with the 90-day US treasury bond yield. The Treasury Bill or T-Bill is a short-term U.S. government debt obligation. This security is known as low-risk and secure investments since they are guaranteed by the U.S. government. The risk-free interest rate in this project is used in the Black and Scholes pricing formula.

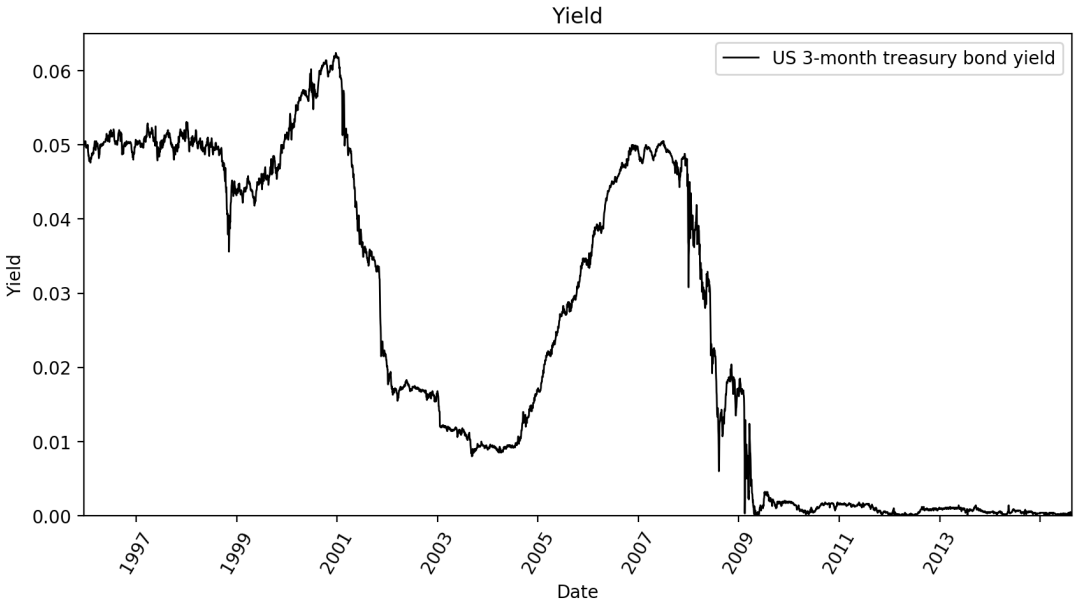
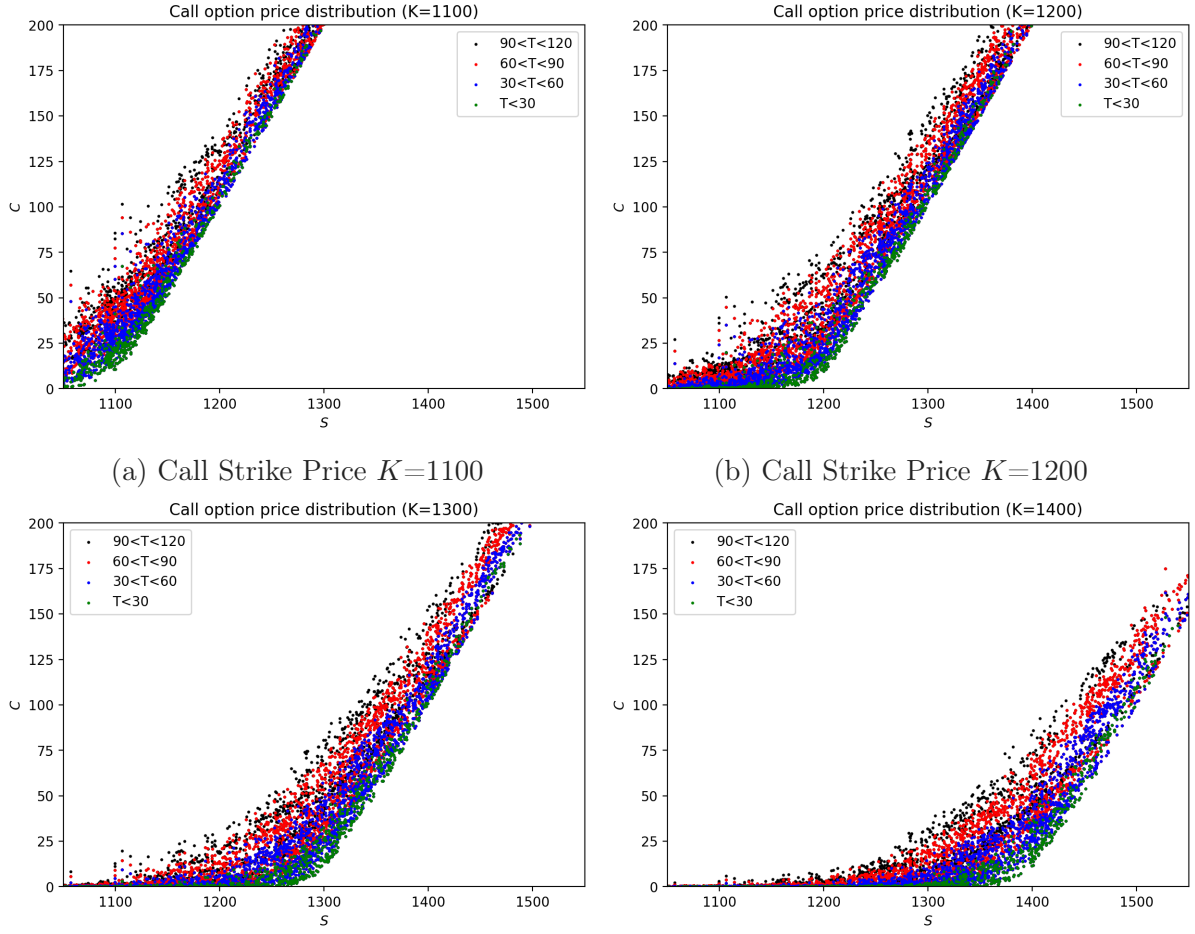


Figure 3.4: US 3-month treasury bond yield

The block of images in Figure 3.5 shows the call option prices with the most common strike prices present in the data set, which are 1100$,1200$,1300$,1400$, respectively. The strike price of the call options in the data set is spread among the smallest strike price 440$, and the bigger strike price 3000$.



(a) Call Strike Price K=1100

(b) Call Strike Price K=1200

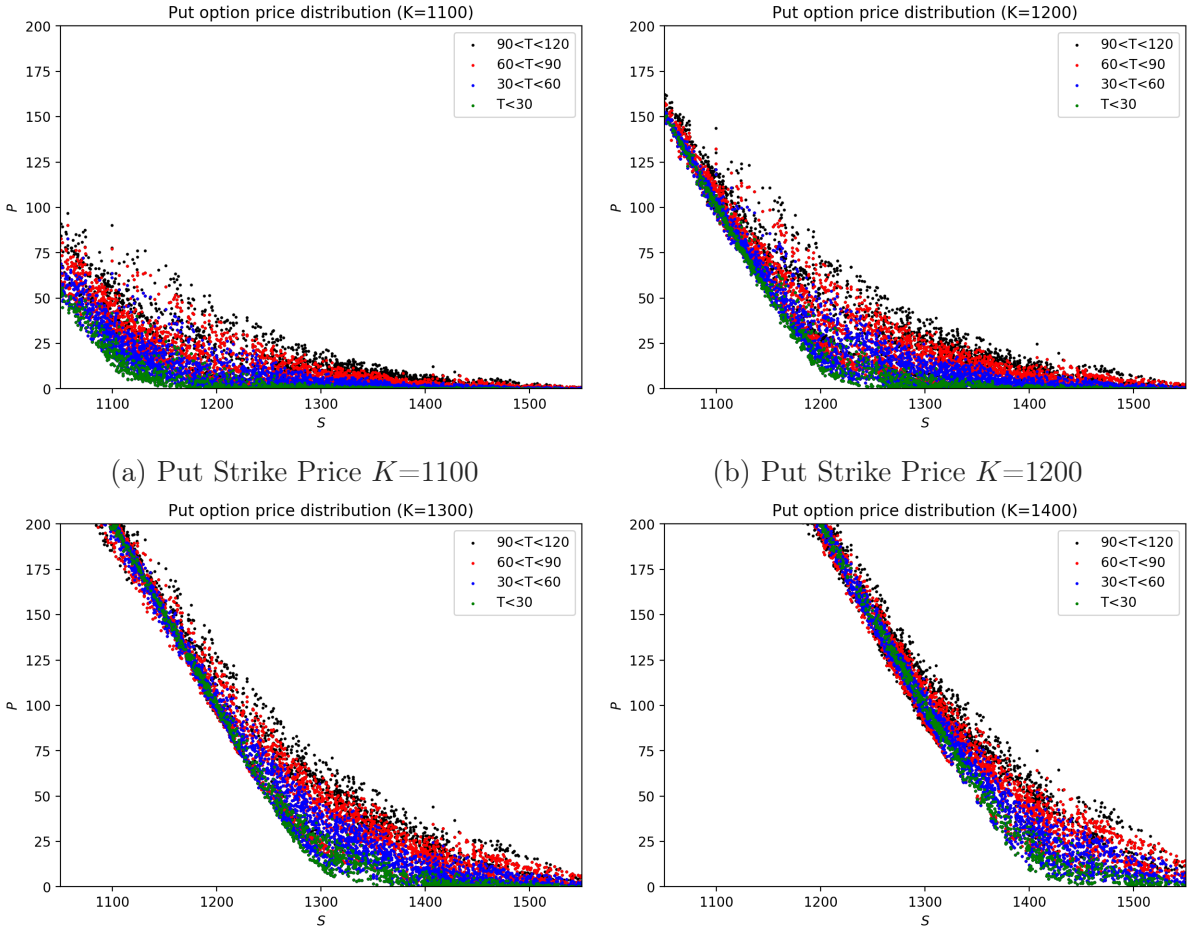
(c) Call Strike Price K=1300

(d) Call Strike Price K=1400

Figure 3.5: Call option price distribution

Similarly to the call option, Figure 3.6 shows the put option prices with the most common strike prices present in the data set. The strike price of the Put options in the data set is spread among the smallest strike price 425$, and the bigger strike price 3000$.

Overall, in the blocks of Figures 3.5 and 3.6 it can be seen the ramp functional shape typical of the options. Moreover, the figures also delineate (color coded) the time value of the option given by the separation of the maturity time. Therefore, it can be observed that the closer the option is to expiration the more similar is to its theoretical shape, convex for the call option, and concave for the put.



(a) Put Strike Price K=1100

(b) Put Strike Price K=1200

(c) Put Strike Price K=1300

(d) Put Strike Price K=1400

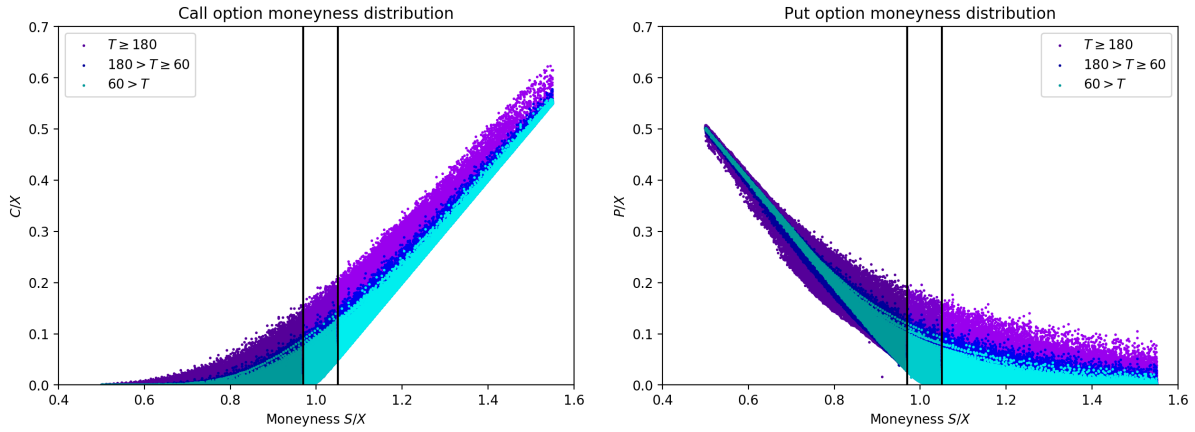
Figure 3.6: Put option price distribution

The moneyness is obtained by normalizing the underlying asset price S_t with the strike price K. Figure 3.7 describes all the characteristics of the option moneyness distribution, creating 9 different subsets. This division process follows Gradojevic et al. (2009) [16] method, which chooses the boundaries of the moneyness and time to maturity in order to have sufficient observations for each subset.

The boundaries are therefore set up at:

1. Time to maturity
   - $T \geq 180$ which represents long-term options
   - $180 > T \geq 60$
   - $T < 60$ which represents the short-term options
2. Moneyness for call options, for the put options the moneyness is inverted.
   - $\frac{S}{K} < 0.95$ which represents the out-the-money (OTM) options
   - $0.95 < \frac{S}{K} < 1.05$ which represents the near-the-money (NTM) options
   - $\frac{S}{K} > 1.05$ which represents the in-the-money (ITM) options

From Figure 3.7 it can be observed that some of both call and put options were extremely ITM, as the graph shows a clear cut, which derives from the exclusion criteria for moneyness levels used by Anders et al.(1996) [3] and explained previosly in this section. It can also be observed how options with maturity below 60 days are mainly concentrated when the option is NTM. Moreover, Table 3.1 allow analyzing further the theoretical phenomena of the options.



(a) Call Option (b) Put option

Figure 3.7: Option moneyness distribution

In the "Average option price" part of Table 3.1 it can be seen that the further the options are from maturity time the higher the premium. The second observable phenomenon is that the price of the options increases as the moneyness increase. This is realistic since higher moneyness could mean that the option has a higher probability of expiring ITM and therefore the price is higher.

Call options

Time to maturity	T < 60	60 < T < 180	T > 180	Total
Moneyness (S/K)	Average option price			
OTM (< 0.95)	2.02693 $	5.8405 $	20.7029 $	10.4957 $
NTM (0.95-1.05)	29.329 $	47.866 $	91.8142 $	49.2401 $
ITM (> 1.05)	146.819 $	164.755 $	221.186 $	177.541 $
	Number or observation			
OTM (< 0.95)	103289	147500	155618	411028
NTM (0.95-1.05)	132254	116280	66464	319226
ITM (> 1.05)	84831	104437	92040	284777

Table 3.1: S&P 500 Data set summary

In the second part of the table, we can see how the data is distributed. The smallest subset is the one with the options with a long-time to maturity and NTM, which account for 66 464 call observations. Overall, the in-the-money (ITM) options make the smallest subset. However, the difference between the different groups of moneyness is not cumbersome.

3.3 Network architecture

The Artificial Neural Network implemented in this thesis is a Multilayer Perceptron (MLP), and its architecture follows the one adopted by Culkin et al. (2017) [9]. The input parameters used consist of: the underlying asset's price series S_t, the strike price K, the time to maturity $T-t$, the risk-free interest rate r_t and the volatility estimates σ_5, σ_{30}, σ_{60}, σ_{90}, σ_{120}, σ_{garch}. The network's goal is to approximate the market's option pricing function $f(\cdot)$ such that:

$$\widehat{C}_t = f\left\{S_t, K, T-t, r_t, \sigma_5, \sigma_{30}, \sigma_{60}, \sigma_{90}, \sigma_{120}, \sigma_{garch}\right\} \tag{3.5}$$

The theorem of Merton (1990) [31] states that the return of the underlying asset is independent of the level of the price of the underlying asset S_t, therefore, also independent of the pricing function $f(\cdot)$ used to determine the price of an option C_t. The theorem demonstrates that this price is homogeneous of degree one in both, the underlying asset S_t and the strike price K. Following the lead of Merton, Huchinson et al. (1994) [22] normalize their data by dividing their inputs with the strike price and the equation above becomes:

$$\frac{\widehat{C_t}}{K} = f\left\{\frac{S_t}{K}, 1, T-t, r_t, \sigma_5, \sigma_{30}, \sigma_{60}, \sigma_{90}, \sigma_{120}, \sigma_{garch}\right\} \tag{3.6}$$

Therefore, this normalized data is fed into the deep learning network as input variables, with the output being the prices $\frac{C_t}{K}$. Regarding the network structure [Figure 3.8], the input layer takes the ten input parameters which are then forwarded through four hidden layers with 120 neurons each. The output layer consists of a single neuron outputting the forecast price. The activation function for the nodes depends on each layer. For the first hidden layer the **Leaky Relu** is selected. In the second and fourth layers, the selected activation function is **elu**. The third layer uses the **relu** function and the output, since the price of an option cannot be negative, is set to be the standard exponential function $exp(\cdot)$. At each hidden layer, during the learning phase a dropout rate of 25% is applied to reduce the probability of overfitting.

To get more robust results the resampling method used in this project bootstrapping. This is commonly used to provide a measure of the accuracy of a given statistical learning method [24]. The data set is divided into two sets 25% for testing and 75% for training. In total 100 iterations of bootstrapping are run. This approach draws observations from the data set one at a time and returning them to the data set after they have been chosen. This allows a given observation to be included in the training set more than once. This sampling method is called sampling with replacement. The performance's evaluation of the neural network model is done by training the model on the train test and evaluating on the test set, where the test set is not included in the train test. The samples not included in the test set are called the out-of-bag samples.

The loss function used for optimization is the mean-squared error (MSE) and the batch size implemented is 64, with 10 epochs.

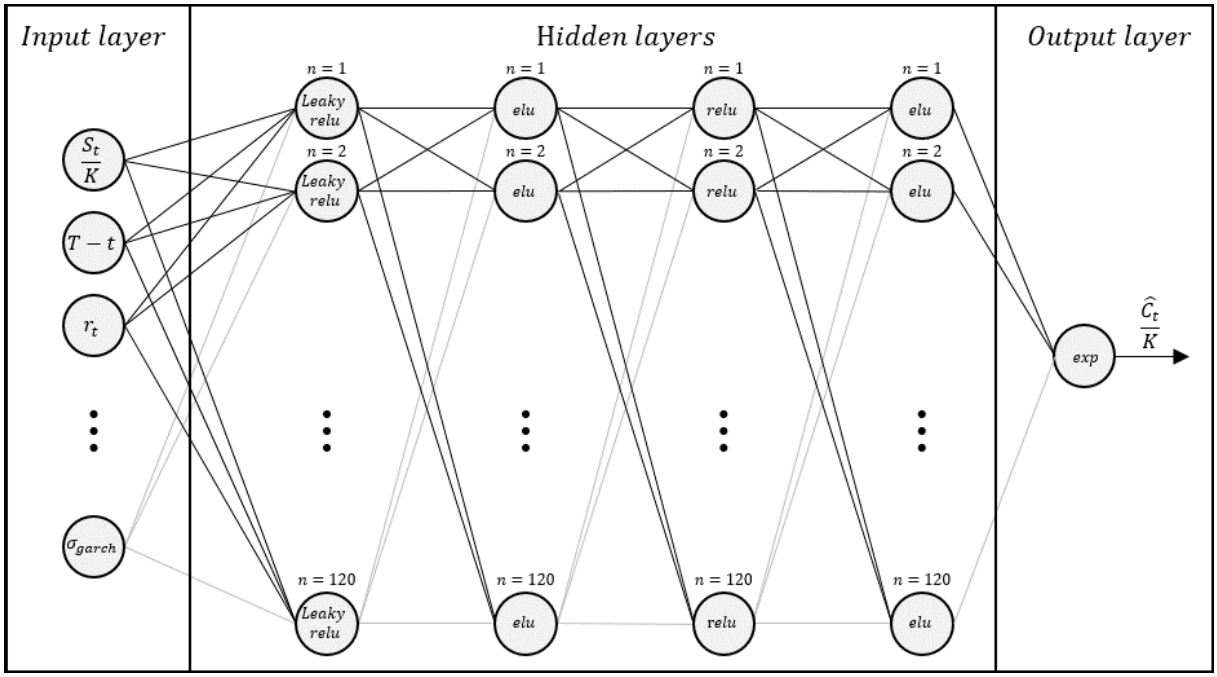


Figure 3.8: Architecture of the ANN

3.4 Performance metrics

3.4.1 Volatility performance

The goal of this section is to examine if the use of GARCH forecasts, compared to the historical volatility estimation method, will reduce the mispricing of the option model. Therefore, in this thesis, two different methods are used to estimate volatility:

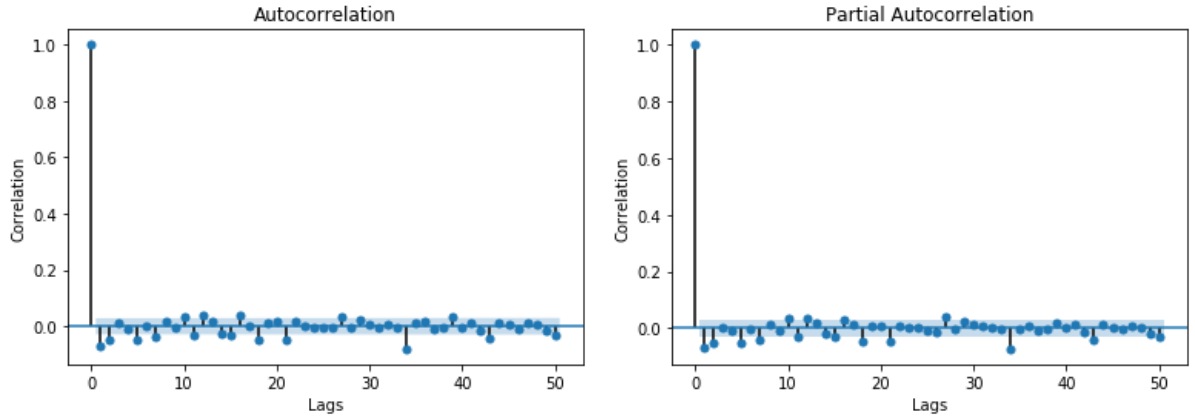
1. Historical volatility, which is estimated by measuring the standard deviation from the moving average over the historical window as described in Section 2.3. The volatility is estimated from the historical price series of the underlying asset, with 5-day, 30-day, 60-day, 90-day and 120-day moving averages. This method is also used by Hutchinson et al. (1994) [22], whose use the standard deviation of the continuously compounded daily returns of the S&P 500 stock market index denoted as s, and the volatility estimate $\widehat{\sigma}_t$ is quoted in annual terms with 252 trading days per year.

   $$\widehat{\sigma}_t = s_t\sqrt{252} \tag{3.7}$$

   The neural network that learns this volatility parameter uses all five volatility measures as input.

2. GARCH(1,1) process, developed by Engle (1982) [12] and Bollerslev (1896) [6] whose believe this model to be appropriate for the daily returns.

The methods such as historical volatility and GARCH use historical returns data which are susceptible to estimation error, due to non-stationarity in volatility. Therefore, is important to ensure that the return volatilities follow a GARCH process before proceeding to estimate option prices with a time-varying variance option model.



(a) Autocorrelation (b) Partial autocorrelation

Figure 3.9: Autocorrelation and Partial Autocorrelation returns

Bollerslev (1986) [6] shows that autocorrelation and partial autocorrelation function for the squared residuals are useful to identify and check the time series behavior in the conditional variance equation of the GARCH form. In figure 3.9 the autocorrelation and the partial autocorrelation of the return of the S&P 500 suggest a GARCH(1,1) model, which would be consistent with the empirical evidence that most financial time series can be represented by the GARCH(1,1).

The resulting parameters of the GARCH(1,1) are shown in Table 3.2, the parameters are denoted in equation 2.22 and they are estimated on a sample of 4699 observations.

Number of observations		4699	
$\sigma_{GARCH(1,1)}$	Coefficients	Standard Error	t
μ	0.0672	1.286e-02	5.223
ω	0.0166	4.770-e03	3.487
α_1	0.0887	1.200-e02	7.396
β_1	0.9002	1.252-e02	71.885

Table 3.2: Estimate of the GARCH(1,1) model

The table shows the mean returns μ, around which the volatility is measured, to be 0.06. However, the most important coefficients are α_1 and β_1. Moreover, since β_1 is an order of magnitude bigger than α_1, the model is rather determined by the residual of the mean equation than by the past volatility. The volatility forecasted with the GARCH(1,1) method is compared, in the next graph, with the short-term historical volatility of five days moving average σ_{MA5}, where it can be seen that they are similar.

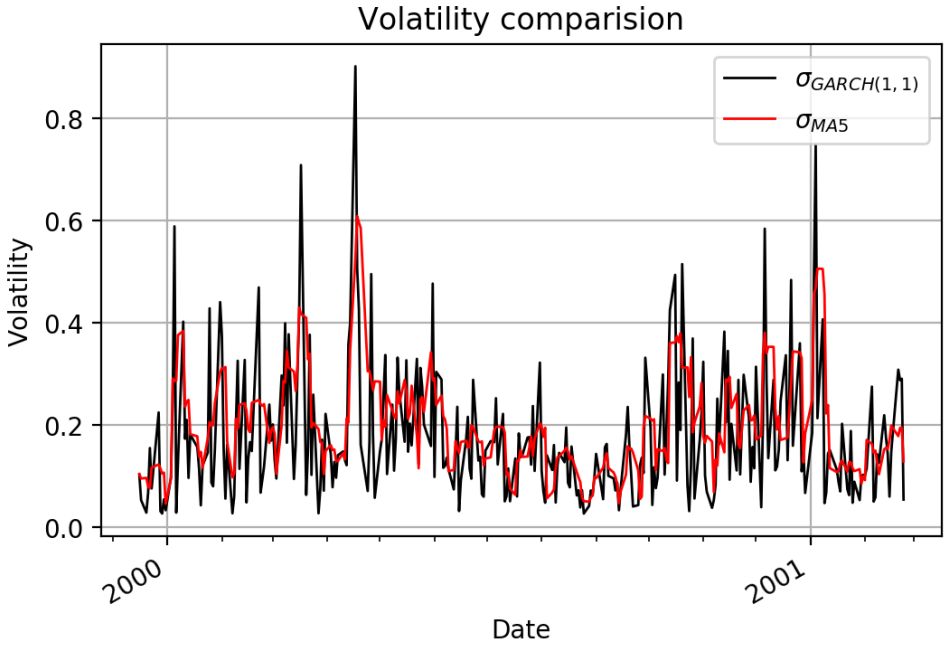


Figure 3.10: Comparision of Garch(1,1) and 5-day moving average rolling volatility

To determine which volatility has the best performance the following metrics are computed: the mean absolute error (MAE), the root mean square error (RMSE), and the mean absolute percentage error (MAPE) are computed. This comparison will lead to choosing the best BSM model which will be afterward compared with the neural network model. Therefore, the best BSM model will be the one with the lowest Root Mean Squared Pricing Error (RMSE) shown in Equation 3.9.

	Pricing error			
Volatility	MAE	MPE	MSE	RMSE
BS5	16.059	0.361774	633.008	25.1597
BS30	14.4626	0.438356	929.371	30.4856
BS60	11.1281	0.318652	491.099	22.1608
BS90	10.4524	0.28125	382.579	19.5596
BS120	10.2476	0.261718	331.286	18.2013
BSgarch	10.3808	0.255054	314.629	17.7378

Table 3.3: Volatility pricing errors

Focusing on the pricing error results [Table 3.3] it can be concluded that the GARCH(1,1) produces lower mispricing compared to the historical volatilities. Therefore, the best model is $BS_{\sigma_{garch}}$. This result is consistent with the one obtained from Chu et al. (1996) [7] in their study about the comparison of the mispricing of option valuation models when different volatility estimation techniques are applied.

3.4.2 Pricing performance

Following the method of Anders et al. (1996)[3], Bennell and Sutcliffe (2004) [4], and Culkin and Das (2017) [9] the pricing performance will be evaluated with tree different error function:

1. Mean Squared Error (MSE)

$$MSE = \frac{1}{N}\sum_{i}^{n}(\widehat{C}_l - C_i)^2 \tag{3.8}$$

2. Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{n}(\widehat{C}_l - C_i)^2} \tag{3.9}$$

3. Mean Absolute Error (MAE)

$$MAE = \frac{1}{N}\sum_{i}^{n}\left|\widehat{C}_l - C_i\right| \tag{3.10}$$

Where, $C_i{=}\frac{C_{i,t}}{K}$, N is the number of observations in the test set, $\widehat{C}_l$ is the call price estimated from the neural network and C_i is the real call price.

Some general results about the Black and Scholes model's ability on pricing options correctly, collected from Yao et al. (2000) [36] and Gencay and Salih (2003) [15], can be presented in four main points:

1. Black and Scholes is a good model for pricing ATM and NTM options, especially when the time-to-maturity is above two months.

2. Black and Scholes model's pricing accuracy decreases significantly for deep ITM and OTM options. The pricing deviation is large, therefore there is a big difference between the model's price and the actual price.

3. Black and Scholes model's often misprices options with maturity less than one month.

4. Black and Scholes model's often misprices options with large volatility.

In Chapter 4 these biases will be measured graphically. The neural network will be quantitatively tested against them, by comparing the pricing prediction of the neural network model on different moneyness, time-to-maturity, and volatilities.

3.4.3 Hedging performance

This subsection is developed following the Hutchinson et al. (1994)[22] method. The authors argue that approximating prices with good accuracy is not synonymous with how well the neural networks learn the dynamics of option pricing. Moreover, Amilon (2003) [2] argues that "the market option prices reflect the anticipation of the future distribution of the underlying asset. If these expectations are not fulfilled on average, then the market option prices are wrong and, consequently, measures-of-fit based on price accuracy are of little use". Therefore, since the option pricing theory is based on the replication of an option with a portfolio composed of other assets, is important to determine whether the neural network model is of practical use.

As described in Section 2.2 of Chapter 2, the delta-hedging strategy can be structured as follows:

$$V_t = V_t(S) + V_t(B) + V_t(C) \tag{3.11}$$

Where $V_t(S)$ is the value of the underlying asset position, $V_t(B)$ is the value of a bond position used to finance the position in the underlying asset and $V_t(C)$ is the value of the option position held in the portfolio at time t.

The composition of the portfolio at time $t = 0$ is assumed to be:

$$V_0(C) = -C_{BSM,0} \tag{3.12}$$

$$V_0(S) = S_0 \Delta_{NN,0} \tag{3.13}$$

$$V_0(B) = -(V_0(S) + V_0(C)) \tag{3.14}$$

$$\Delta_{NN,0} = \frac{\partial C_{NN,0}}{\partial S_0} \tag{3.15}$$

Where $C_{NN,0}$ is the price of the call option predicted by the neural network from which the delta of the call option $\Delta_{NN,0}$ can be computed. The strategy consists of writing one call option (3.12), going long for the underlying asset for $\Delta_{NN,0}$ number of shares at price S_0 (3.13), and going short for a bond (3.14) to finance the rest of the long position in the underlying asset that is not financed with the sale of the call option. The initial value of the replicating portfolio is zero since the long position is financed entirely with riskless borrowing and the sale of the call option.

$$V_0 = V_0(S) + V_0(C) + V_0(B) = 0 \tag{3.16}$$

Between the initialization of V at time $t = 0$ and the expiration at T, at all time $t - T$ the spot and bond positions are rebalanced daily to satisfy the followings:

$$V_t(S) = S_t \Delta_{NN,t} \quad where \quad \Delta_{NN,t} = \frac{\partial C_{NN,t}}{\partial S_t} \tag{3.17}$$

$$V_t(B) = e^{r\tau} V_{t-\tau}(B) - S_t(\Delta_{NN,t} - \Delta_{NN,t-\tau}) \tag{3.18}$$

Where r is the risk-free rate and τ is defined to be one day in this case. The tracking

error is defined by the value of the replicating portfolio at expiration date:

$$V_T = V_T(S) + V_T(C) + V_T(B) \tag{3.19}$$

In this project the same performance measures adopted by Hutchinson et al. (1994)[22] are applied in analyzing the hedging strategy performance: The present value of the expected absolute tracking error:

$$\xi = e^{-rT} E[|V(T)|] \tag{3.20}$$

And the prediction error:

$$\eta = e^{-rT} \sqrt{E^2 \left[V(T)\right] + Var \left[V(T)\right]} \tag{3.21}$$

Where ξ provides information regarding the accuracy of the option pricing formula and η in which the information contained in the expected tracking error is combined with the information contained in the variance of the tracking error.

The delta of the neural network for the call option $\Delta_{NN,0}$ in Equation 3.15 is computed using the Jacobian matrix. Assuming that y is the output vector of f, the main idea is that the Jacobian matrix of f contains the partial derivatives of each element of the output y with respect to each element of the input x.

$$\mathrm{J}_f = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_m} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix} \tag{3.22}$$

$$\mathrm{J}_{ANN} = \begin{bmatrix} \frac{\partial C_t}{\partial S_t} \\ \vdots \\ \frac{\partial C_t}{\partial \sigma_{GARCH}} \end{bmatrix} \tag{3.23}$$

The built-in function provided by the TensorFlow package was used to compute the Jacobian of the ANN.

The hedging performance, in this thesis, consists of a comparison between the delta-hedge analysis for the Black and Scholes model and the artificial neural network model. This comparison will be developed on the test set considering only the options contracts that have over 50 days of observations in the test set.

To conclude a two-sided test is performed to test for the null hypothesis that two independent samples have identical expected values.

Chapter 4

Results

In this chapter, the results obtained from the empirical study are presented. There are two sub-chapters:

1. Pricing results in which the Black and Scholes pricing formula and the Artificial Neural Network model are compared in their pricing prediction.
2. Hedging results which present the outcome of the delta hedging strategy.

4.1 Pricing results

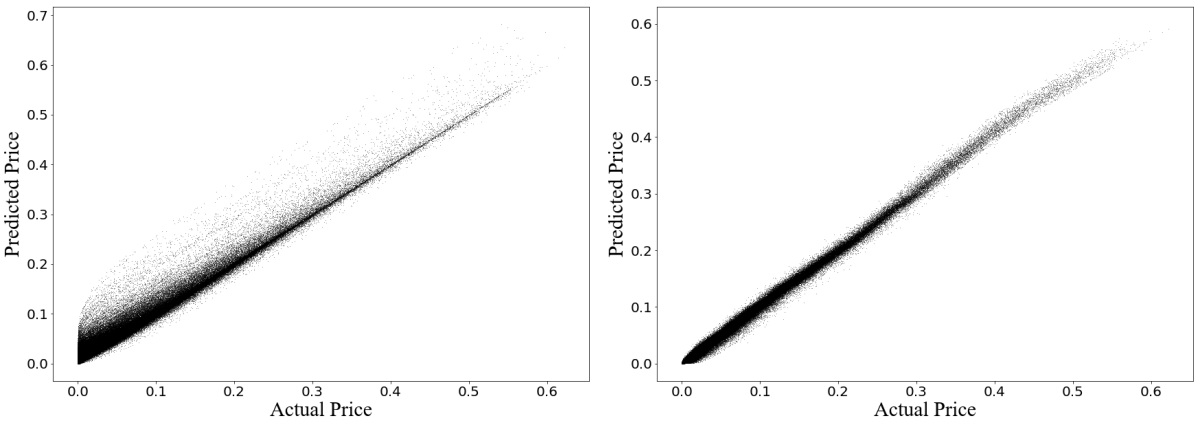
The results consist of three columns with the pricing errors MSE, RMSE, and MAE, for both the neural network and the Black and Scholes. The 95% confidence interval are shown in parentheses.

Pricing performance - entire test set				
No. Observations	Model	MSE	RMSE	MAE
1015063	Neural Network	8e-05	0.00853	0.00572
	CI	(3e-05,0.0002)	(0.00562,0.01402)	(0.00359,0.01039)
	Black - Scholes	0.00026	0.01626	0.00882
	CI	(0.00026,0.00027)	(0.0162,0.01632)	(0.00879,0.00885)

Table 4.1: Pricing performance - Entire test set

Table 4.1 presents the pricing performance results of the entire test set, from which it can be seen that the neural network model outperforms Black and Scholes in pricing for all the error measures. This outcome is consistent with the previous literature [Section 1.4], several authors agreed on the superiority of the Neural network model when the data set was not divided by any measure.

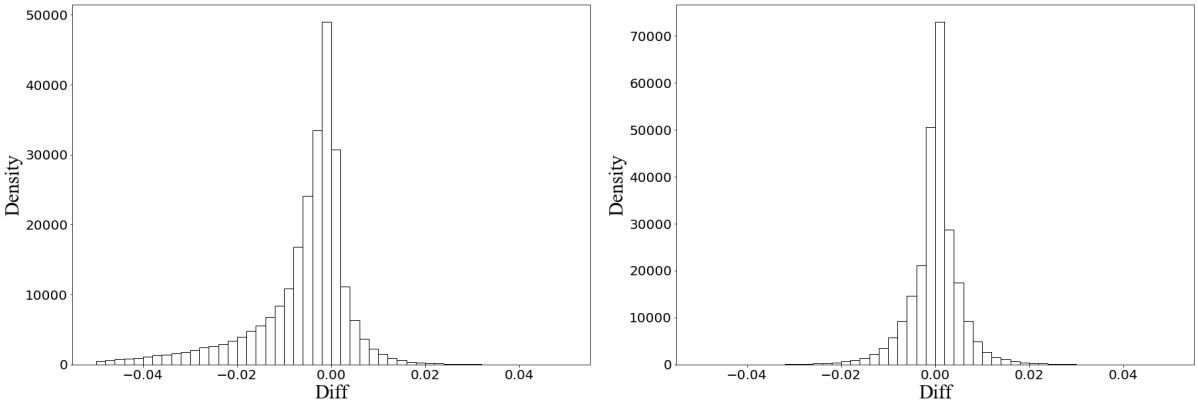
A graphical representation of the entire test set results can be seen in Figure 4.1 where the predicted price is plotted as a function of the actual price.



(a) Black & Scholes (b) Artificial Neural Network

Figure 4.1: Predicted price vs Actual price

From the two graphs, on one hand, it can be seen that, when the prices are small, the Black and Scholes model's predictions tend to deviate. On the other hand, there is almost a straight line for the neural network model. Moreover, Black and Scholes shows a greater dispersion in the left corner which means that the predicted price does not match with the actual price. This analysis can be confirmed by looking at the histogram of the pricing error of the two models in Figure 4.2.



(a) Black & Scholes (b) Artificial Neural Network

Figure 4.2: Pricing error density

The pricing error density of the neural network is centered and with most of the observations around zero, it appears also to have an equal amount of errors' density in both sides of the mean. However, the pricing error density of the BSM is clearly skewed on the left side of the distribution in fact the interval is between -0,05 and 0,02. The negative skew of the density distribution means that the BSM model tends to underprice the options, which is consistent with the previous literature such as Gencay & Salih (2003)[15].

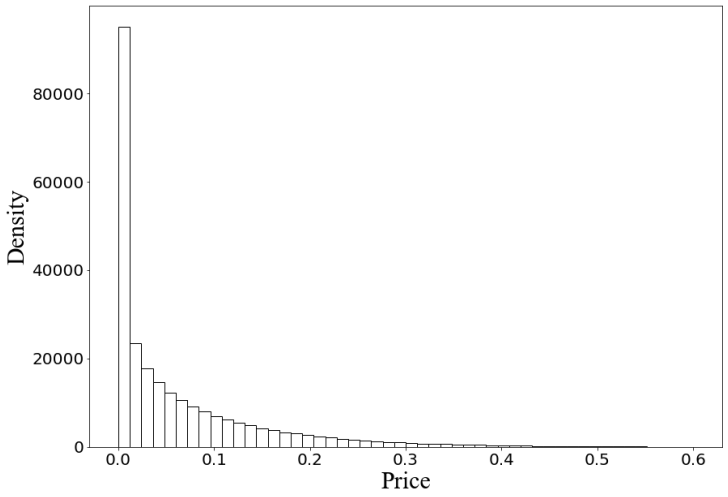


Figure 4.3: Price Density

In the price density figure, it can be seen that the majority of the observation consist of small prices.

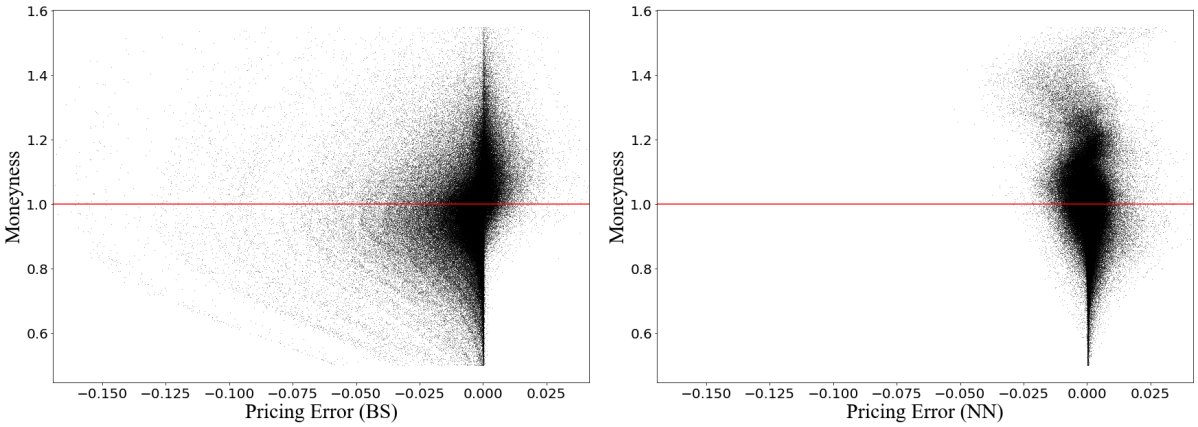
Pricing performance - grouped by different moneyness					
Model	Moneyness	No. Observations	MSE	RMSE	MAE
Neural network			0.00017	0.01237	0.00983
CI	ITM	284747	(6e-05,0.0005)	(0.00764,0.02239)	(0.00542,0.01854)
Black-Scholes			0.00026	0.0161	0.00815
CI			(0.00026,0.00026)	(0.01598,0.01622)	(0.00811,0.0082)
Neural network			2e-05	0.00456	0.00246
CI	OTM	411028	(1e-05,4e-05)	(0.00359,0.00667)	(0.00194,0.00373)
Black-Scholes			0.00031	0.0177	0.00968
CI			(0.00031,0.00032)	(0.01758,0.01778)	(0.00963,0.00972)
Neural network			7e-05	0.008	0.00626
CI	NTM	319226	(3e-05,0.00022)	(0.00521,0.01478)	(0.00372,0.01255)
Black-Scholes			0.00021	0.01436	0.0083
CI			(0.0002,0.00021)	(0.01425,0.01448)	(0.00826,0.00834)

Table 4.2: Pricing performance - Grouped by moneyness

As it can be seen from Table 4.2 comes to the attention that the greatest mispricing is for the OTM options for the BSM. This is consistent with the above analysis. Moreover, the previous literature Yao et al. (2000)[36] and Gencay & Salih (2003)[15] agreed on the mispricing tendency of the BSM model especially for options deeply ITM and OTM, where the misprice consists of underpricing OTM options and overpricing ITM options.

Figure 4.4 represents the pricing error as a function of moneyness. The red horizontal line divides ITM options above from OTM options below. This division brings to attention the mispricing problems discussed above:

- In Figure 4.4a below the red line, the dispersion on the left side of the mean is bigger, than the ANN in Figure 4.4b, in both magnitude and density. Therefore,



(a) Black & Scholes (b) Artificial Neural Network

Figure 4.4: Moneyness vs Pricing error

the OTM options tend to be more underpriced by the Black and Scholes model than by the ANN.

- In Figure 4.4a above the red line the dispersion on the right side is bigger, not in magnitude, but in density than the dispersion in Figure 4.4b. This means that the BSM overprice ITM options more frequently than ANN.

Is also interesting to observe how the more the option goes deep ITM the more the ANN has more variance in the error, where overprices the options. Moreover, in Table 4.2 the ITM error of the ANN does not differ much from the BSM. In the previous literature (Yao et al. 2000 [36], Gencay & Salih 2003 [15]) the ITM options are usually overpriced by the BSM, therefore these results are not consistent with them.

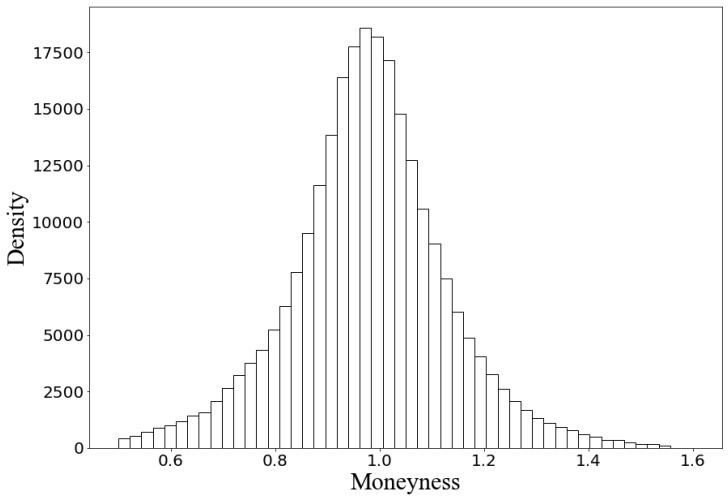


Figure 4.5: Moneyness Density

Figure 4.5 could lead to one of the reasons why the overpricing behavior of the ANN for deep ITM options. It can be argued that the mispricing is due to a lack of data. In fact from the graph that describes the moneyness distribution in the data set it can be seen that the deeper the option goes ITM the smaller the density of observation in the data

set. Since the ANN is a data driven-model, the model would learn the features associated with the most common type of data in the data set. This could also be the reason why the previous literature has different outcomes. In fact, Yao et al. (2000) [36] found that there was not much difference in the degree of mispricing between the two models and Bennell and Sutcliffe (2004) [4] found the BSM model to outperform the neural network model. However, overall, when the data set is partitioned by moneyness the ANN remains superior to the BSM model for all moneyness types.

In Table 4.3 the data is grouped by maturity where:

- Short represents $T - t < \frac{1}{12}$
- Medium represents $\frac{1}{12} \geq T - t \geq \frac{1}{2}$
- Long represents $T - t > \frac{1}{2}$

Pricing performance - grouped by different maturity					
Model	Maturity	No. Observations	MSE	RMSE	MAE
Neural network			0.00017	0.01237	0.00983
CI	Short	104111	(6e-05,0.0005)	(0.00764,0.02239)	(0.00542,0.01854)
Black-Scholes			0.00026	0.0161	0.00815
CI			(0.00026,0.00026)	(0.01598,0.01622)	(0.00811,0.0082)
Neural network			6e-05	0.00736	0.00501
CI	Medium	596822	(2e-05,0.00018)	(0.00459,0.01338)	(0.00289,0.00988)
Black-Scholes			0.0001	0.01016	0.00576
CI			(0.0001,0.0001)	(0.01012,0.0102)	(0.00574,0.00578)
Neural network			0.00011	0.00998	0.0068
CI	Long	314130	(5e-05,0.00026)	(0.00688,0.01606)	(0.00459,0.01122)
Black-Scholes			0.00052	0.02278	0.01378
CI			(0.00051,0.00052)	(0.02268,0.02289)	(0.01372,0.01384)

Table 4.3: Pricing performance - Grouped by maturity

On one hand, it can be seen that the artificial neural network model outperforms the Black and Scholes model for Medium and Long maturity options. On the other hand, in the Short maturity case, the difference in mispricing between the two models is small. The ANN outperforms BSM when the error is measured with MSE and RMSE, contrary the BSM performs better when the error is measured with MAE. Since the difference in error is infinitesimal it can be concluded that there does not seem to be a significant difference in mispricing between the two models for short maturity options.

The results presented in this table are in disagreement with the previous literature for:

- Short-term options where Gencay & Salih (2003) [15] and Bennell & Sutcliffe (2004) [4] found the ANN to be clearly superior to BSM in the short-term options. One of the reasons why in this model the ANN is just slightly better than the BSM in the short-term options could be the pre-processing method adopted in Chapter 3 Section 3.2. In this section Equation(3.2), the options with less the 15 days are excluded from the data set, however, these options tend to have high trading volumes, and

therefore, by excluding them fewer observations are available for where the ANN could be learning from.

- Long-term options where Yao et al. (2000) [36] and Bennell & Sutcliffe (2004) [4] found that the BSM model tends to outperform the ANN model.

Moreover, using a model as the artificial neural network is important to emphasize that the results obtained depend on both the composition of the data set and the importance that the ANN model gives to each of the inputs. If one input is not important for the ANN in determining the output, its density distribution will not affect much the output. This behavior can be seen when relating Table 4.3 with Figure 4.6.

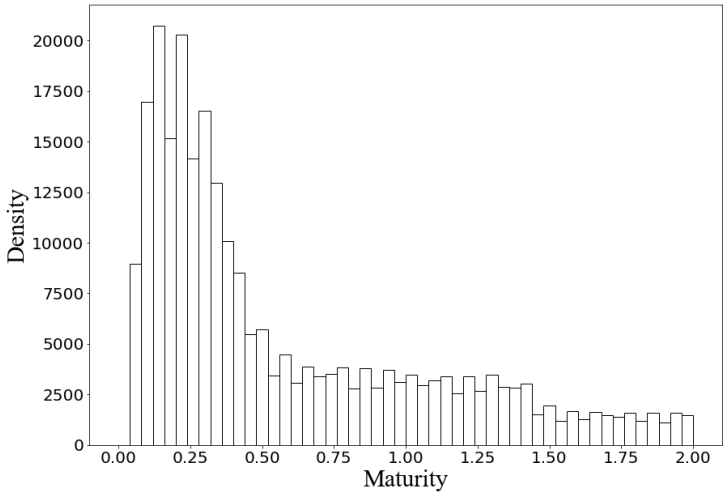
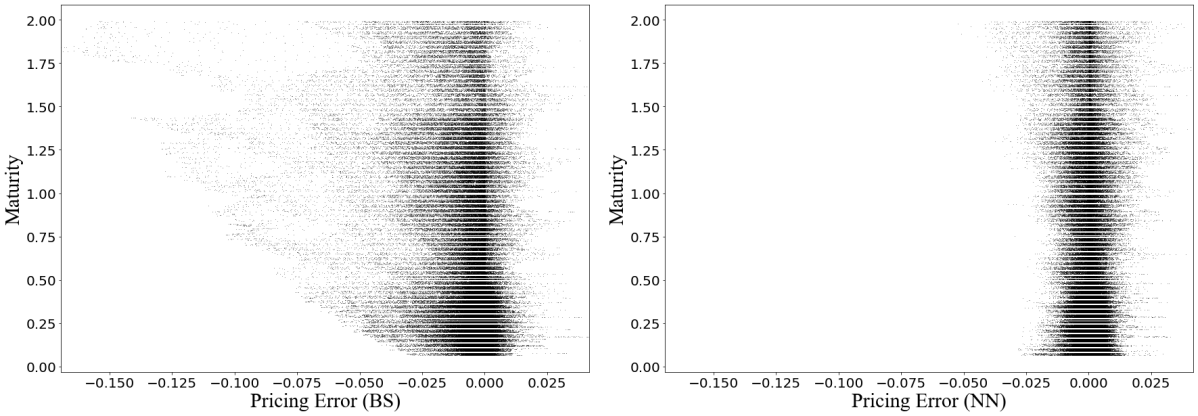


Figure 4.6: Maturity Density

The data set mostly consists of short-term maturity options, and yet in Table 4.3, the difference between ANN and BSM is not significantly big. This could mean that the ANN model gives more importance to other input variables in predicting the price of the options.



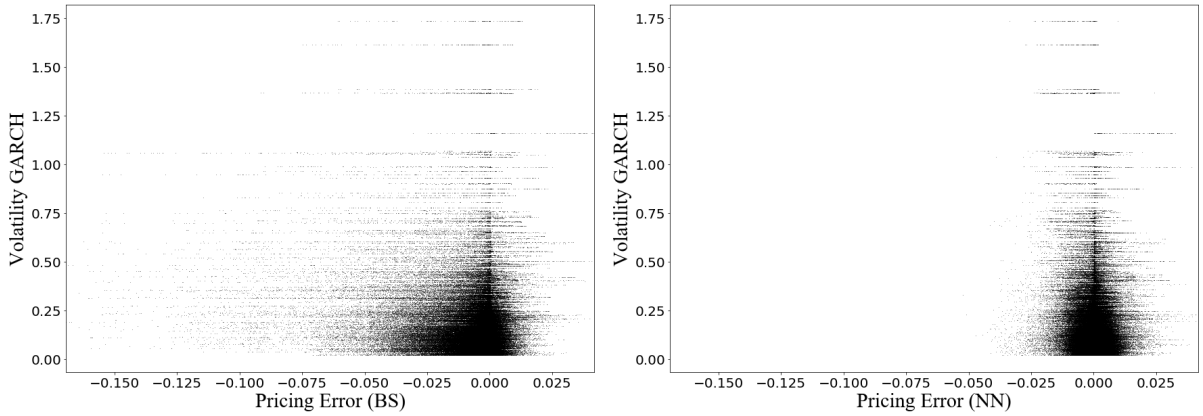
(a) Black & Scholes (b) Artificial Neural Network

Figure 4.7: Maturity vs Pricing error

Figure 4.7 represents the pricing errors of the Artificial Neural Network as a function of time-to-maturity for both models. It can be observed that for both the ANN and BSM the pricing errors are centered on zero. On the other hand, BSM's pricing errors have more variance and it clearly shows some underpricing as the maturity increases, however, the density of this variance is minimal. Therefore, this graph is consistent with the fact that there is not much difference in the pricing error of the two models when grouped by time-to-maturity.

Gencay and Salih (2003) [15] in their study found that as the maturity increases the BSM model exhibits an increase in the pricing error. This appears to be consistent with Figure 4.7 .

Figure 4.8 represents the pricing errors of the two models as a function of the Garch volatility (which is selected based on the study in Chapter 3 Section 2.3).



(a) Black & Scholes (b) ANN

Figure 4.8: Volatility vs Pricing error

It can be observed that again the neural network is more centered on zero and has less deviation in both directions. The BSM model seems to systematically underprice options with an increase in the pricing error as the volatility increases. Yao et al. (2000) [36], Gencay & Salih (2003) [15] call this results as bias of the BSM model. These results, lead to the conclusion that ANN models can learn the influence of the different levels of volatility on the price of an option. Moreover, ANN models will not deviate as much as the BSM's when the volatility increases.

To summarize the pricing results, the artificial neural network model is superior to the Black and Scholes model regarding the pricing performance overall. The BSM has negatively skewed error distribution, which means that the model tends to underprice options. On the other hand, the ANN's pricing errors are more centered on zero with less deviation than the BSM model.

4.2 Hedging results

The hedging performance results for the neural network model and the Black and Scholes model are measured with the mean and standard deviation of both, the discounted absolute tacking error ξ and the prediction error η. In parenthesis are shown the 95% confidence intervals.

Table 4.4 represents the hedging performance results of the entire tests sets obtained from the bootstrapping.

Hedging performance				
Model	ξ		η	
	Mean	Standard Deviation	Mean	Standard Deviation
Neural network	3177.27	4609.52	4546.48	7373.94
CI	(2358.4,4040.19)	(2753.56,6572.05)	(3196.95,5975.31)	(4215.22,10633.79)
Black-Scholes	3149.18	4573.93	4505.79	7313.58
CI	(2335.87,4006.0)	(2728.18,6526.57)	(3165.89,5924.05)	(4173.22,10556.03)

Table 4.4: Hedging performance

The hedging performance results for the entire test set show that for both model's measurement errors, the mean and the standard deviation are very similar. This result is conflicting with the pricing results presented in the previous section, since the ANN outperform in pricing BSM it would be expected to do the same for the delta hedging strategy.

Hedging Performance - Grouped by Moneyness					
Model	Moneyness	ξ		η	
		Mean	Standard Deviation	Mean	Standard Deviation
Neural network		3308.77	4802.04	4624.33	7413.77
CI	ITM	(1429.78,5461.73)	(1638.78,7910.27)	(1587.0,8153.61)	(1891.85,12862.3)
Black-Scholes		3282.75	4770.87	4587.34	7362.0
CI		(1419.28,5418.02)	(1631.43,7856.02)	(1575.62,8086.72)	(1882.93,12768.96)
Neural network		3098.28	3182.31	4607.95	5176.4
CI	NTM	(2198.8,4253.37)	(2703.06,3846.39)	(3150.77,6422.02)	(4336.47,6335.38)
Black-Scholes		3066.18	3150.99	4560.34	5124.96
CI		(2176.85,4213.99)	(2681.02,3805.07)	(3119.85,6362.68)	(4300.0,6267.53)
Neural network		1289.82	1430.61	1773.3	2267.05
CI	OTM	(1208.22,1409.59)	(1305.45,1613.09)	(1642.39,1961.07)	(2058.72,2567.68)
Black-Scholes		1274.16	1417.4	1752.15	2245.52
CI		(1193.22,1392.89)	(1293.15,1598.22)	(1622.33,1938.24)	(2038.76,2543.4)

Table 4.5: Hedging Performance - Grouped by Moneyness

In Table 4.5 the performance measures are grouped by moneyness where

- ITM refers to $S/K > 1.05$
- OTM refers to $S/K < 0.95$
- NTM refers to $0.95 \geq S/K \geq 1.05$

Similarly, when grouped by the moneyness the BSM model is slightly better than the neural network. However, the difference is not cumbersome.

Hedging Performance - Grouped by Maturity					
Model	Maturity	ξ		η	
		Mean	Standard Deviation	Mean	Standard Deviation
Neural network		3074.82	4519.9	4399.11	7291.33
CI	Short	(2610.85,3575.63)	(2864.48,6326.97)	(3569.37,5286.54)	(4375.44,10378.11)
Black-Scholes		3046.53	4483.24	4358.0	7228.98
CI		(2586.25,3543.05)	(2838.99,6278.64)	(3536.54,5237.55)	(4333.46,10296.06)
Neural network		3416.59	4933.2	4924.5	7765.88
CI	Medium	(758.2,5083.79)	(1122.76,7511.12)	(831.47,7542.44)	(1281.37,12447.57)
Black-Scholes		3388.88	4900.59	4884.45	7710.76
CI		(749.76,5045.22)	(1117.69,7451.61)	(822.61,7484.69)	(1275.25,12348.86)
Neural network		284.13	30.16	282.54	35.95
CI	Long	(264.41,294.3)	(11.11,38.24)	(259.01,294.71)	(13.03,45.87)
Black-Scholes		279.26	29.64	277.68	35.34
CI		(259.87,289.25)	(10.92,37.59)	(254.55,289.65)	(12.81,45.1)

Table 4.6: Hedging Performance - Grouped by Maturity

In Table 4.6 the performance measures are grouped by maturity where:

- Short represents $T - t < \frac{1}{12}$
- Medium represents $\frac{1}{12} \geq T - t \geq \frac{1}{2}$
- Long represents $T - t > \frac{1}{2}$

Similar results are observed when the data is partitioned by maturity. The BSM is slightly better than the ANN model in hedging, but the difference is still not substantial.

In order to investigate if the two models are significantly different, a two-sided t-test is performed. The null hypothesis is:

$$D = \mu_{BS} - \mu_{ANN}$$

- $H_0 : D = 0$
- $H_1 : D \neq 0$

Where D is the difference between the means of the discounted absolute tracking error. For the ANN and BS.

From Table 4.7 it can be observed that the null hypothesis is not rejected, this means that the results are not statistically significant different from each other.

The results obtained in the hedging strategy performance are in disagreement with the one obtained in the previous literature.

Paired t-tests		
Sample	t-statistic	p-value
Moneyness		
ITM	0.586	0.565
OTM	0.187	0.853
NTM	0.241	0.813
Maturity		
Short	0.427	0.675
Medium	0.888	0.386
Long	0.901	0.419
All	0.888	0.386

Table 4.7: Paired t-test

On one hand, in terms of discounted absolute mean tracking error, Hutchinson et al. (1994) [22] found the ANN model to outperform the BSM model, in terms of discounted mean tracking error Amilon (2003) [2] found also the ANN model to be superior to the BSM model.

On the other hand, Anders et al. (1998) [3] exploring the hedging parameters found that the neural network is not capable of learning the dynamics of the options' partial derivatives. Their model overestimated the delta parameter and did not follow the monotonically increasing function. Following the lead of Anders et al. (1998) [3], the delta surfaces are plotted to enquire why the ANN does not outperform in the hedging strategy, even though is superior in predicting option prices. Since the hedging strategy is performed on a great number of call options, for visualization reasons only 5000 observations are plotted.

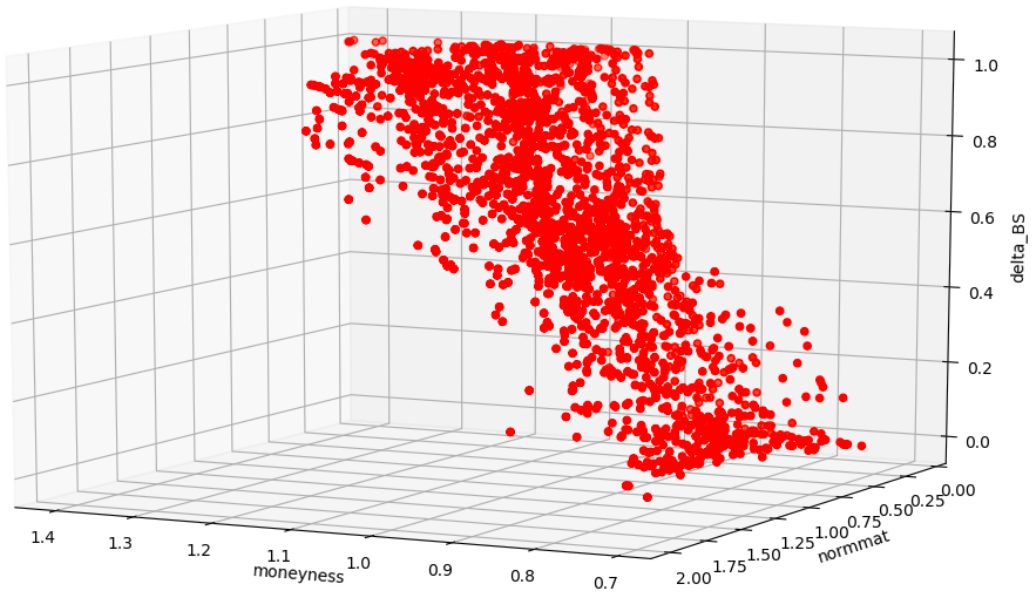


Figure 4.9: Δ_{BS}

The surface of the Δ_{BS} [Figure 4.9] is a monotonically increasing function and it follows the bounding condition for a call option where the $\Delta \in [0, 1]$. It can be seen that there is a delta decay which means that the time-value of the call options increases for ITM options and decreases for OTM options as saw in Cox & Rubinstein 1985, pp. 46-47 [8].

Contrary, The surface of the Δ_{ANN} [Figure 4.10] it is not a monotonically increasing function.

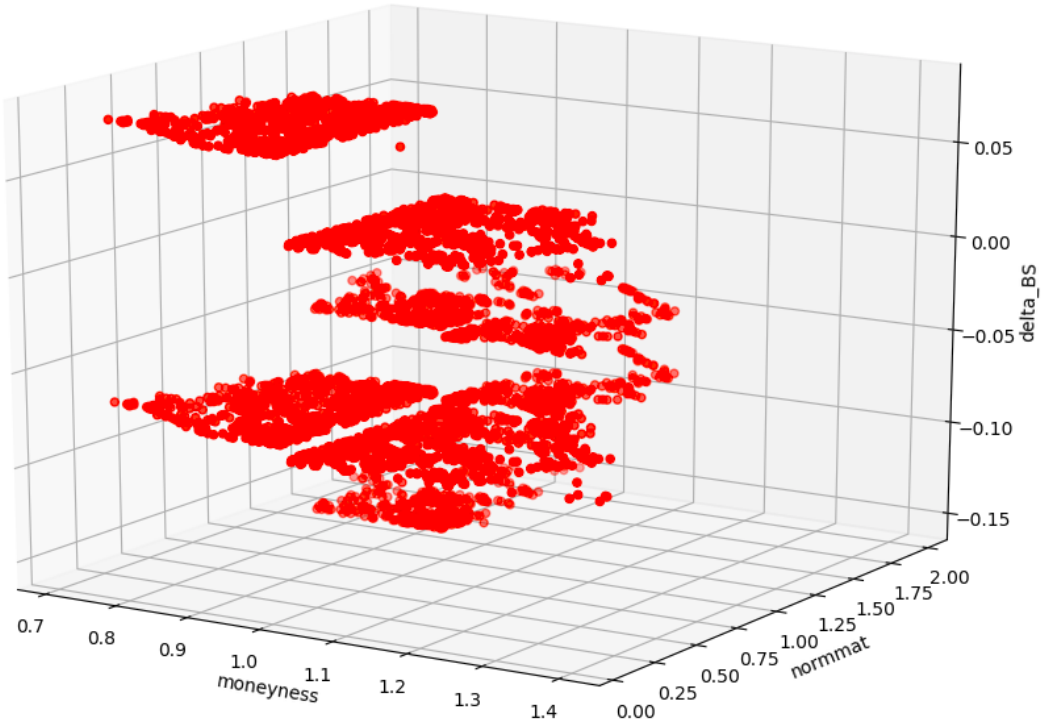


Figure 4.10: Δ_{ANN}

This could explain why the ANN does not outperform the BSM model. However, since the two models are not significantly different from each other, the ANN is still capable of performing as good as the BSM formula. Moreover, since the ANN is a data-driven model, the nature of the data is always important to take into consideration in case of outperforming or not. In this case is clear that the ANN is not able to learn the dynamics of the Delta in the same fashion as the BSM.

To conclude, with this data set and this model, the BSM model is slightly better than the ANN model regarding delta hedging, although, the two models are not significantly different from each other.

Chapter 5

Conclusions and further work

5.1 Conclusions

This thesis analyses and gives an overview of the researches conducted regarding pricing options using Artificial Neural Networks. The two models, Artificial Neural Network and Black and Scholes are studied and compared in order to analyze whether one outperforms the other in terms of pricing and hedging the European-Style call options on the S&P 500 stock index. As in the previous literature, this project also focused on the basic MPL neural network since there would always be some room for improving the model.

The pricing performances were measured with several error functions, such are Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). Subsequently, the hedging performances were measured in terms of Mean and Standard Deviation of the discounted absolute tracking error ξ and the prediction error η. After the data pre-processing 1.015.063 observations were left to conduct the study. The data set is divided into a train set with 75% of the observations, and a test set with 25% of the observations, afterward to have a better understanding of the pricing performance 100 iterations of bootstrapping are run. The Multilayer Perceptron was trained on the train test and the performance measures were evaluated on the test set. Subsequently, the results were compared with the results obtained from the well known Black and Scholes pricing formula. The best Black and Scholes model was chosen based on which volatility estimation that produces the lowest mispricing. From the study emerges that between several moving average historical volatility and GARCH the method that produces less pricing error is the GARCH, therefore the Garch volatility estimation is used to compute Black and Scholes formula. In terms of pricing performance, the results show that the ANN model outperforms the BSM model in pricing the European-style call options. These results are consistent with the previous literature. However, in terms of hedging performance, the results show that there is no statistical significance between the two models. This is in contrast with some of the previous literature.

Overall, it seems that the Neural Network is able to learn and outperform the dynamics of European-styled options pricing and at least perform as good as the BSM the hedging strategy.

5.2 Further work

Regarding the hedging performance, it is suggested for further research to introduce new input variables in the MLP, even though they could not improve the pricing performance they could be important for the ANN to improve the approximation of the delta hedging.

In order to improve the hedging performance a new neural network that learns the dynamics of the Delta could be implemented.

Experimenting with different models such as Convolutional Neural Network or Recurrent Neural Network could lead to an improvement of the results.

Moreover, including all the observations contained before the data-exclusion process could lead to a more generic model.

Bibliography

[1] Fazeel Abid et al. "Sentiment analysis through recurrent variants latterly on convolutional neural network of Twitter". In: *Future Generation Computer Systems* 95 (2019), pp. 292–308.

[2] Henrik Amilon. "A neural network versus Black–Scholes: a comparison of pricing and hedging performances". In: *Journal of Forecasting* 22.4 (2003), pp. 317–335.

[3] Ulrich Anders, Olaf Korn, and Christian Schmitt. "Improving the pricing of options: A neural network approach". In: *Journal of forecasting* 17.5-6 (1998), pp. 369–388.

[4] Julia Bennell and Charles Sutcliffe. "Black–Scholes versus artificial neural networks in pricing FTSE 100 options". In: *Intelligent Systems in Accounting, Finance & Management: International Journal* 12.4 (2004), pp. 243–260.

[5] Fischer Black and Myron Scholes. "The pricing of options and corporate liabilities". In: *Journal of political economy* 81.3 (1973), pp. 637–654.

[6] Tim Bollerslev. "Generalized autoregressive conditional heteroskedasticity". In: *Journal of econometrics* 31.3 (1986), pp. 307–327.

[7] Shin-Herng Chu and Steven Freund. "Volatility estimation for stock index options: a GARCH approach". In: *The Quarterly Review of Economics and Finance* 36.4 (1996), pp. 431–450.

[8] John C Cox and Mark Rubinstein. *Options markets*. Prentice Hall, 1985.

[9] Robert Culkin and Sanjiv R Das. "Machine learning in finance: The case of deep learning for option pricing". In: *Journal of Investment Management* 15.4 (2017), pp. 92–100.

[10] B Efron. "Bootstrap methods: another look at the jackknife. e Annals of Statistics, 7 (1): 1–26". In: *URL http://www. jstor. org/stable/2958830* (1979).

[11] Walter Enders. *Applied econometric time series*. John Wiley & Sons, 2008.

[12] Robert F Engle. "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation". In: *Econometrica: Journal of the Econometric Society* (1982), pp. 987–1007.

[13] Stephen Figlewski. "Forecasting volatility using historical data". In: (1994).

[14] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[15] Ramazan Gençay and Aslihan Salih. "Degree of mispricing with the black-scholes model and nonparametric cures". In: *Economics and Finance. Annals* 4 (2003), pp. 73–101.

[16] Nikola Gradojevic, Ramazan Gençay, and Dragan Kukolj. "Option pricing with modular neural networks". In: *IEEE transactions on neural networks* 20.4 (2009), pp. 626–637.

[17] Richard HR Hahnloser et al. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit". In: *Nature* 405.6789 (2000), pp. 947–951.

[18] Michael Hanke. "Neural networks versus Black-Scholes: An empirical comparison of the pricing accuracy of two fundamentally different option pricing methods". In: *Journal of Computational Intelligence in Finance* 7.1 (1999), pp. 26–34.

[19] Espen Gaarder Haug. *The complete guide to option pricing formulas*. McGraw-Hill Companies, 2007.

[20] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.

[21] John C Hull. *Options futures and other derivatives*. Pearson Education India, 2003.

[22] James M Hutchinson, Andrew W Lo, and Tomaso Poggio. "A nonparametric approach to pricing and hedging derivative securities via learning networks". In: *The Journal of Finance* 49.3 (1994), pp. 851–889.

[23] AG Ivakhnenko. "Heuristic self-organization in problems of engineering cybernetics". In: *Automatica* 6.2 (1970), pp. 207–219.

[24] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[25] Stamatios V Kartalopoulos and Stamatios V Kartakapoulos. *Understanding neural networks and fuzzy logic: basic concepts and applications*. Wiley-IEEE Press, 1997.

[26] Hans R Künsch. "The jackknife and the bootstrap for general stationary observations". In: *Annals of Statistics* 17.3 (1989), pp. 1217–1241.

[27] David C Lay and Thomas Polaski. *Instructor's Solutions Manual [to Accompany] Linear Algebra and Its Applications, Update [by] David C. Lay*. Addison-Wesley, 2006.

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[29] Mary Malliaris and Linda Salchenberger. "A neural network model for estimating option prices". In: *Applied Intelligence* 3.3 (1993), pp. 193–206.

[30] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[31] Robert C Merton and Paul Anthony Samuelson. "Continuous-time finance". In: (1990).

[32] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[33] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[34] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[35] Jinghong Shu and Jin E Zhang. "Testing range estimators of historical volatility". In: *Journal of Futures Markets: Futures, Options, and Other Derivative Products* 26.3 (2006), pp. 297–313.

[36] Jingtao Yao, Yili Li, and Chew Lim Tan. "Option price forecasting using neural networks". In: *Omega* 28.4 (2000), pp. 455–466.