Master Thesis

Smart Risk Analysis Service for Cryptocurrencies.

performed at

CAMPUS 02 GRAZ

Master degree programme in
Information Technologies & Business Informatics

From: Hans-Juergen Griesbacher
Student number: 01351980

Graz, July 5, 2020

....
Hans-Juergen Griesbacher

1 Ehrenwörtliche Erklärung / Affidavit

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.
I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

....
Hans-Juergen Griesbacher

Abstract

Cryptocurrency brokers offer price quotations to investors for a set period. The investor can decide whether to accept the offer while the quote remains open. The broker bears a risk of the price changing while the quote is open. To mitigate this risk, brokers add an individual risk premium to the price.
This thesis investigates how to improve the risk premium's precision by using an artificial neural network (ANN) instead of the classical general autoregressive conditional heteroscedasticity (GARCH) model. The thesis opens by describing financial risks, conventional traditional approaches, and the theory of machine learning, in particular neural networks. Subsequently, short-term volatility in the market is explored in terms of volatility prediction with the classical GARCH and ANN approach with a focus on an investor's purchase process. With the predicted volatility, the risk premium is calculated with the expected short-fall method.
This thesis explores BTC-USD, ETH-USD, and ETH-BTC. The risk premium is calculated for a 10-minute quotation period with data from the cryptocurrency exchange Kraken from October 2019 to April 2020. The analysis shows that the ANN approach delivers a more precise volatility prediction and risk premium calculation due to a lower mean square error deviation compared to the GARCH model. However, the ANN has a low explanation of power in the test period. Different volatility cluster phenomena in the short-term data compared to the classical daily basis are also identified.

Contents

2 Introduction

Cryptocurrencies are increasingly finding their way into society and are intended to provide an alternative to the existing financial system. Acceptance is still growing. However, the overall number of people who are actively using coins or tokens is low. This is due to different reasons in the token economy. In the following thesis, we will go into more detail about the struggles of buying and selling cryptocurrencies and how to measure the current market risk.

The current wave of cryptocurrencies was born in 2008 with Bitcoin (BTC). The last big hype about cryptocurrencies and the transformed crypto bubble was at the end of 2017 to January 2018. After the burst of the cryptocurrency bubble, prices and demand for tokens plunged. Since then, most coins and tokens in the cryptocurrency field have been facing high volatility and low trading volume.

There are different ways for people and institutions to get tokens. Either they can be purchased from an exchange or, the most popular form, from a broker. Brokers make a quote for a specific time for buying or selling tokens. A quote is an offer of buying or selling a token at a particular price. This can be seen as a buying or selling option, which remains available to the user for a specific duration. The duration is also called the lifetime and indicates how long a quote is valid. After the lifetime expires a new quote is calculated. If the quote is accepted during its lifetime, the investor receives the tokens at the fixed quote price. At the same time, a counter trade on an exchange is made by the broker. During the duration of the quote, the broker bears the price-change risk.

In the long-run, the broker has to ensure that he can deliver the token without making losses. To overcome the price risk, the broker has to add a risk premium for the duration of the quotation process. The risk premium is then added to the current market price. Therefore, it is essential to have the risk premium as high as necessary to bear the price-change risk and have a positive outcome, and as low as possible to guarantee competitive prices.

There are well-known methodologies for measuring the market risk, such as the Value at Risk (VaR) and thereof the Expected Shortfall (ES). With these measurements, the risk premium for a specific period can be calculated. These risk measurements in their standard form require that the price-change distribution of the underlying asset be distributed normally. In the literature, Neftci (2000); Brooks, Clare, Molle, and

Persand (2005), we can find much evidence that in the classical financial field, and all likelihood for cryptocurrencies, the distribution is not distributed normally.

Underlying risky assets in the return-generating process behave differently in extreme events than in normal market distributions (Neftci, 2000). What is more, risky financial assets have fat tail distribution, and this might bring an under-prediction of extreme movements and the frequency of the occurrence (Brooks et al., 2005).

Classic approaches, like the Generalized AutoRegressive Conditional Heteroscedasticity (GARCH) models from the financial field, might not be the best solution to predict the volatility of the market. This thesis will analyze whether a neural network (NN) approach can provide a more accurate solution in terms of measuring the volatility in the market, therefore leading to a more precise risk premium.

2.1 Research Question

The objective of this master thesis is to calculate a market-adjusted risk premium to overcome a short-term price-change risk. This means we are predicting the volatility of the short-term cryptocurrency market with machine learning (ML) and a traditional approach.

The research question is: To what extent is it possible to achieve a more precise risk premium with a NN approach compared to a standard GARCH model for the cryptocurrency market?

Hypotheses: NNs can predict volatility in the cryptocurrency market more precisely than standard GARCH models, and therefore a more precise risk premium can be calculated in a defined back-testing period.

Antitheses: NNs cannot predict volatility in the cryptocurrency market more precisely than standard GARCH models, and therefore a more precise risk premium cannot be calculated in a defined back-testing period.

Market risk prediction is an essential aspect of the financial industry. The market risk in financial markets is mainly driven by the potential price-change process. Therefore, a precise time-series forecast for the volatility is a relevant aspect beyond the cryptocurrency market as well.

The crucial factor for measuring market risk is the volatility (standard deviation) of the risky asset. Volatility clustering is a phenomenon which can be seen in financial markets. This means that in periods of high volatility, it is more likely that the volatility will stay high and vice versa. What is more, the returns in financial and cryptocurrency markets show a high level of kurtosis. This leads to a highly leptokurtic distribution (Phillip, Chan, & Peiris, 2017).

The volatility prediction will be made using a NN and the standard GARCH model. The predicted volatility is needed to calculate the risk measurements. A risk premium that covers the market risk for the short-term future is calculated with these risk measurements. The risk premium will then be added to the current market price. This will be compared with the price change of the next period.

The back-testing is done in a defined period for a specific cryptocurrency pair. The outcome of the risk premium for each interval will be compared with the change in the market price in the next interval. Thus, the back-testing phase will test which approach produces a better outcome.

In this thesis, we focus on the cryptocurrency pairs BTC-USD, ETH-USD and ETH-BTC with a short-term period of 10 minutes as the quotation lifetime.

There are a variety of different coins and tokens available on the market. Therefore, we will use the word token in the following text to address all kinds of assets, coins and tokens. For this thesis, a token must fulfill two constraints. It has to be a cryptographical secure digital asset, and it must be tradable on an exchange.

2.2 Structure of this thesis

Chapter two of the thesis explains the background. The current financial risk measurements with their strengths and their weaknesses will be briefly mentioned. It also gives an introduction to the GARCH models and their calculation. Furthermore, the calculation for a risk premium will also be defined. The technical theory starts with general information about different machine learning approaches, such as supervised, unsupervised and reinforcement learning. Next, the process of a NN is explained. What is more, we get an insight into the current research field of the prediction of volatility.

In the third chapter, we get insights into the data which are used in this thesis for the cryptocurrency pairs. The data fetching approach, the period and the source of data are outlined and followed by a detailed discussion about the data extraction and transformation.

The fourth chapter is about the methods which are used to measure the volatility and the risk premium. This starts with an overview of the parameter training for volatility prediction models and the prediction itself. Then the process of the risk premium calculation is defined.

Next, the result section shows the experimental setup. Then the results of the benchmark (GARCH model) and the outcome of the NN are demonstrated. After this, the corresponding findings are compared. At the end of this section, the risk premium is calculated with both volatility prediction approaches, and they are compared.

The next chapter is about the discussion of the results, the limitation of the NN model and the fast changes in the financial field. The discussion evaluates the model critically and interprets the outcome of the NN and GARCH approaches, as well as the existing risk measurements. Then a summary and potential future work are mentioned.

The last chapter is the conclusion of the thesis. Here the thesis statement is reiterated and the critical points of the work are highlighted.

3 Background

This chapter focuses on the theoretical background of the following master thesis. It starts with the types of risks, how to measure risk and calculate the risk premium. Then it focuses on the machine learning part with a short overview of the different machine learning approaches. Next, an explanation of the process and the elements of a neural network are given. In the related work chapter, the literature review gives us deeper insights into the current scientific research in this field. At the end, we highlight the point of distinction of this thesis relative to related work.

3.1 Risk

In many business fields, risk and measurement of risk are essential topics. Therefore, a precise definition of risk is critical. The meaning of risk has been loaded with controversy. Hence the description can influence the outcome of policy debates, power distribution and resource allocation (Fischhoff, Watson, & Hope, 1984).

The word "risk" is a broad term and has many different definitions. Most of the definitions have in common the idea that risk results from the uncertainty of future outcomes. The description which fits best for this approach is the following definition. Risk is the possibility of deviating from goals as a result of the unpredictability of the future (Jonen, 2007).

3.2 Different Types of Risks

We distinguish between four different types of risk in the financial field. There are credit risk, operational risk, liquidity risk and market risk.

Credit Risk:
Credit risk compares the default risk and migration risk. Default risk is the default of the debtor or any contracting party. With migration risk, we mean the risk management if the credit deteriorates (Hull, 2018, p. 429-434).

Operational Risk:
Operational risk has an internal, external and legal perspective. Internal risks are personnel risks, process and structural risks, system and technology risks. External risks include natural disasters, terrorist attacks, blackout, etc. Legal risks include any controvertible or invalid contract design, incorrect disclaimer, etc. (Hull, 2018, p. 515-536).

Liquidity Risk:
Liquidity risk addresses short-term liquidity risk, funding liquidity and market liquidity. Short-term liquidity risk is the risk of the bankruptcy of a party in a contract. Funding liquidity risk is structural liquidity risk. This means that the refinancing costs are increasing and this harms liquidity. Market liquidity bears the risk that the asset cannot be traded on a market because the volume which can be traded is less than the requested trading volume (Hull, 2018, p. 537-565).

Market Risk:
Market risk describes the change in market prices for an asset. Examples here are interest rate risk, credit spread risk, equity risk, foreign exchange risk, commodity risk and volatility risk (Hull, 2018, p. 159-230).

In the following thesis, we focus on the market risk for equity risk in the token market. This is due to the research question, which focuses on the price fluctuation during a specific period.

3.3 Measuring Market Risk

In the following section, we will focus on how to measure market risk. We start with defining the assumptions for the market, then focus on volatility, GARCH models, and then go more deeply into the risk measurement of a market with the VaR and ES approach. In the following text, the return is defined as the price change between t_0 and t_1.

3.3.1 Volatility

For every participant in the financial field, it is important to monitor and measure the volatility in a market. The volatility denoted as σ, is the standard deviation of the return. In this thesis, the unit of time is measured in minutes. The volatility is the standard deviation of the proportional change in the return of a specified period (Hull, 2018, p. 213-220).

Volatility can be interpreted as the temperature of the market, and it changes rapidly. If the risk management does not take these fluctuations into account, it will often under- or overestimate the risk of the current market situation (Eberlein, Kallsen, & Kristen, 2002).

The volatility can also be shown in the variance σ_t^2 of the return which is the volatility squared. It measures the dispersion of the actual return from the average return. Calculations are made to determine that positive deviations are not neutralizing negative deviations, and to find the expected value of the deviation squared from the expected return. A high mean value indicates a high dispersion of the outcomes (Bodie, Kane, & Marcus, 2018, p. 127).

Volatility Clustering:
Volatility clustering refers to the latency of the return of small or large changes back to the mean level. This means that large changes in the return tend to be followed by large changes in the future. Minor changes also tend to be followed by small changes. This leads over time to volatility clustering (Mandelbrot, 1963).

What is more, absolute returns in the financial field show that the autocorrelation slowly declines for all positive returns over time (Granger & Ding, 1995).

3.3.2 Methods of Volatility Forecasting

In the following section, the different methods for volatility forecasting with their models are explained.

3.3.2.1 Historical Volatility Models

The historical volatility models are calculated with historical volatility. These models are an easy and robust volatility forecasting method.

Random Walk:
The random walk model is the simplest type of historical volatility model. The model implies that the difference between the current volatility t_0 and the following volatility t_1 is random noise. This indicates that the next period's volatility is the current volatility. This model includes the changes in near-term levels and the persistence of volatility (Marra, Stephen, 2015).

Historical Mean:
This model takes the historical volatility data and calculates average volatility with

equal weights to all data. Historical mean captures the mean-reverting aspect. However, the changing character of volatility and the short-term persistence are not obtained. This model underestimates future volatility due to its assumption that volatility reverts to its long-run average (Marra, Stephen, 2015).

Moving Average:
This model calculates the volatility with a fixed length of time and weighting scheme. It does not include the whole history and is weighted towards the current fluctuation. The moving average approach can be modified by adjusting the weighting scheme (Marra, Stephen, 2015).

Exponentially Weighted Moving Average (EWMA):
The EWMA model decreases the weights exponentially when moving back through time. With this approach, the data storage requirements are modest. It only needs the current estimation of the variance and the recent observation of the market value. The formula is

$$\sigma_n^2 = \lambda\sigma_{n-1}^2 + (1-\lambda) * \mu_{n-1}^2 \tag{3.1}$$

The σ_n for period n is calculated from the σ_{n-1} of the last period and μ_{n-1} is the current percent change. λ is the weight of the current period. Therefore, it is a relative number between zero and one (Hull, 2018, p. 225-227).

3.3.2.2 Discrete Historical Models

The discrete historical models use regular and historical periods. These periods are assigned weights, but these weights stay the same in the period (Marra, Stephen, 2015).

Autoregressive Moving Average Models (ARMA):
The ARMA model adds an autoregression to the moving average from the historical models, which are mentioned above. With this adjustment of the weights, the predictive power increases. This autoregressive model uses a linear function of a time series of its previous values. With the adjustment of the weights, the model becomes more dynamic. This added variable leads to a better fit of the last observation to the next one (Marra, Stephen, 2015).

Autoregressive Conditional Heteroscedasticity (ARCH):
The ARCH model was described by Engle (1982). In 2003, Robert Engle received the Nobel Prize in economics for his work on ARCH models. The ARCH model estimates the variance within a long-term perspective average variance and n observations. The younger an inspection, the more weight it is given (Hull, 2018, p. 223-225).

Generalized Autoregressive Conditional Heteroscedasticity (GARCH):
The GARCH model was first suggested by Bollerslev (1986). The ARCH model was extended to the GARCH. In the ARCH model, the next period's volatility is limited by the last period's volatility. The GARCH model also takes into account the changes in the error term and follows the volatility about its long-term mean (Marra, Stephen, 2015).

The ARCH and GARCH models show in studies from Danielsson and Moritomo (2000) and Danielsson and de Vries (2000) that their approach is not perfect for analyzing significant risks because the outcome shows volatile estimations.

The performance of these models gets worse when it comes to analyzing the risk in the tail of the losses (Gencay, Selcukb, & Ulugülyagci, 2003).

3.3.3 GARCH models

To be able to predict the standard deviation of financial time series, GARCH models are used most frequently in practice. The standard GARCH model estimates the variance of the future period based on characteristics of the squared returns and the variances of past periods. The GARCH equation is:

$$\sigma_t^2 = \omega + \alpha_1 * r_{t-1}^2 + \alpha_2 * r_{t-2}^2 + ... + \alpha_q * r_{t-q}^2 + \beta * \sigma_{t-1}^2 + \beta_2 * \sigma_{t-2}^2 + ... + \beta_q * \sigma_{t-q}^2 \tag{3.2}$$

where σ_t^2 is the variance for the current period, ω is an unknown parameter (the expected value calculated using the model), the alpha parameters α_q indicate the extent to which the current variance is dependent on past squared returns, and the beta parameters β_p suggest the area to which the current variance depends on the variances of past periods. The two key figures, p and $q(p, q)$ show how many prior periods are taken into account and are referred to as the order of the GARCH model.

For example, a standard GARCH model of order (1, 1) only takes into account the squared return of the last past period and the conditional variance of the recent period. In practice it is often limited to a GARCH model of order (1, 1), since this already produces good results, which can usually only be improved insignificantly by taking other past periods into account (Cryer & Chan, 2008, p. 289-296).

This standard GARCH model has been specified for many different purposes in the past. One of these specifications is the threshold GARCH (TGARCH) model, which will be briefly introduced here since TGARCH models have already shown good results in empirical studies in modelling the volatility of the Bitcoin price (Katsiampa, 2017). One reason why TGARCH models are particularly suitable for analyzing financial time series is that a distinction can be made between the effects of positive and negative price shocks. Empirical analyses of economic data have shown that negative price shocks lead to a more significant increase in volatility than positive price shocks. The TGARCH model of order (1, 1) can be stated as (Stier, 2001, p. 355-358)

$$\sigma_t^2 = \omega + \alpha_1 * r_{t-1}^2 + \beta * \sigma_{t-1}^2 + \gamma * r_{t-1}^2 * I_{t-1} \tag{3.3}$$

where I_{t-1} would assume a value of 1 in the case of a negative return and 0 in the case of a positive return. As a result of a negative return, the value of the squared return of the previous period has a more significant influence on the conditional variance of the current period than in the case of a positive return ($\alpha + \gamma$ instead of α) (Stier, 2001, p. 355-358).

3.4 Risk Measurement

In this section, we will gain a brief insight into the risk measurements of the VaR and the ES. Next, we will define the risk premium calculation for the price quotation process.

3.4.1 Value at Risk

In general, value at risk (VaR) is the highest loss, expressed in monetary units, which will not be exceeded with a certain probability over a certain forecast period (Jorion, 2006, Chapter 1-2).

The VaR is a risk measurement which gives us the following statement: Let us assume the VaR is -7% for a confidential interval of 95%. This means that with a certainty of 95%, the loss will not be greater than 7% or with a certainty of 5%, the loss will be greater than or equal to 7%.

Here it is calculated as the probability of losses during the period. The VaR is equal to the loss at the nth percentile of the distribution. VaR is often used to measure the risk because it is easy to understand. What is more, VaR answers the question of "How bad can the price get during this period?". However, the VaR is not saying what the probable loss will be, if we come below the nth percentile (Hull, 2018, p. 213-220).

There are different equations for the calculation of the VaR as a risk measurement. Most widely used is:

$$VaR_\alpha = -\mu + \sigma * Z_\alpha \tag{3.4}$$

where the expected value of the return is denoted by μ, σ stands for the standard deviation and Z_α for the Z value of the standard normal distribution for a given confidence level α. In practice, confidence levels of 95% or 99% are most widely used (Dowd, 2005, p. 27-32, 154-159).

The criticism of the VaR is that it only provides information about which loss will not be exceeded with a certain probability. However, it gives no information about what loss is expected if the VaR is surpassed. The concept of the VaR is valuable, but it is by no means infallible (Hochegger, 2010).

3.4.2 Expected Shortfall

The expected shortfall (ES) is a measure that can produce better incentives for traders than VaR. ES is also referred to as conditional tail expectation, expected tail loss, or conditional value at risk. As mentioned above, VaR asks the question: "How bad can things get?", whereas ES asks: "If things do get bad, what is the expected loss?" (Hull, 2018, p. 269-281).

The ES of 5% corresponds to the expected loss in the event that the 5% of the worst cases occur at a confidence level of 95% (or the expected loss in the event that the 1% of the worst cases occur at a confidence level of 99%). The ES is usually calculated as (Dowd, 2005, p. 35-37, 154-159):

$$ES_\alpha = -\mu + \sigma * \frac{\varphi(Z_\alpha)}{1-\alpha} \tag{3.5}$$

where the value of the density function at the point of the Z-value for the given confidence level α is denoted by $\varphi(Z_\alpha)$ (Dowd, 2005, p. 154-159).

3.4.3 Calculation of the Risk Premium for the ES with the Confidence Level

For the ES, we have to set two parameters: the time horizon and the confidence level (α). The time horizon depends on the application. When the market is very liquid and actively traded, a short time horizon makes sense. For an illiquid market, a longer time horizon should be chosen (Hull, 2018, p. 269-281).

The risk premium for each trade is calculated as

$$Rp_\alpha = E(R) + \frac{\alpha}{1-\alpha} ES \tag{3.6}$$

where Rp_α is the risk premium for the confidence level of alpha. $E(R)$ is the expected return of the market and defines the trend of the market. This risk premium calculation is a classical approach to achieve risk-neutral price calculations.

3.4.4 Black-Scholes Option Pricing

The Black-Scholes (BS) model is a model which was developed by Black and Scholes (1973). The model was initially viewed as a mathematical description of the financial market and derivative investment instruments with constant volatility where the trading option actually poses a risk due to random components such as the volatility. A few years later, Sheraz and Preda (2014) were working on the calculation of the implied volatility for the Black-Scholes model using GARCH models.

To simplify the analysis, the Black-Scholes model assumes that all investors are risk-neutral, that the expected return on the underlying asset corresponds to the risk-free interest rate, and that the expected payment from the option is calculated at the end of the period and was discounted with the risk-free interest rate. The basis for the BS model was the Black-Scholes-Merton differential equation. The equation of the Black-Scholes-Merton formula is

$$c = S_0 N(d_1) - Ke^{-rT} N(d_2) \tag{3.7}$$

and

$$p = Ke^{-rT} N(-d_2) - S_0 N(-d_1) \tag{3.8}$$

where

$$d_1 = \frac{ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}} \tag{3.9}$$

$$d_2 = \frac{ln(S_0/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T} \tag{3.10}$$

where c stands for call and is the European buy option. p stands for put and is a European sell option. A so-called European option has the characteristic that the option can only be executed at the end of the duration. S_0 is the price of the underlying (Asset) at time zero, K is the strike price, r is the risk-free rate, σ is the volatility of the market and T is the term to maturity. The function $N(x)$ is the cumulative distribution function of a standard distribution. This means it is the probability that the variable of a standard normal distribution $\phi(0, 1)$ will be smaller or equal to x (Hull, 2018, p. 571-582, 673-675).

3.4.5 Black Scholes vs Expected Shortfall

ES is calculated from the variance-covariance approach. To use this approach, we need the standard deviation σ of the distribution. The volatility is well estimated for the benchmark. We use a GARCH model in the calculation to be able to predict the standard deviation. With the predicted standard deviation, one can deal more precisely with present fluctuations in volatility to obtain the risk premium and thereby cover the x% of the worst cases.

The BS approach views the quotation as an option, calculates the option value from it and uses it as a risk premium. To calculate the BS with a call option, the current market price, the identical exercise price, an interest rate of zero and the predicted volatility could be used. The exercise price and the market price are assumed to be identical since this calculates the risk premium based on the change in volatility. The risk premium is, therefore, the value of the option.

As mentioned in Mitra (2015), the relationship between ES and option prices, like BS, is overlapping. However, there is very little literature which relates to the two approaches. Because the ES approach was developed to measure precisely the risk in the market, this thesis will take the ES into account for the calculation of the risk premium.

3.4.6 Market Assumptions

As mentioned above from Neftci (2000); Brooks et al. (2005) in section 2, the return distribution of the market is most likely not distributed normally. The distribution is leptokurtic and has, therefore, fat tails and high peaks. Consequently, we are focusing on the Student's t distribution for the calculation. For the sake of simplicity, the explanation above about the equations for the risk measurements and the ES were for the normal distribution.

The Student's t-probability density function is given by

$$f(t) = \frac{\Gamma(\frac{\mu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})}(1+\frac{t^2}{\nu})^{-\frac{\nu+1}{2}} \tag{3.11}$$

where Γ is the gamma function and ν are the degrees of freedom (Student, 1908).

The ES equation with a Student's t distribution is

$$ES_\alpha = -\mu + \sigma * \sqrt{\frac{n-2}{n}} * \frac{\phi_t(t_n^{-1}(\alpha))}{1-\alpha} * \Big(\frac{n + (t_n^{-1}(\alpha))^2}{n-1}\Big) \tag{3.12}$$

where n is the amount of degrees of freedom of the distribution and $\phi_{t,n}$ is the density function of the student-t distribution (McNeil & Embrechts, 2005, p. 45-46).

3.5 Machine Learning

The term machine learning is broadly defined and omnipresent nowadays. It stands for multiple applications and methods which are at work in the field of generating knowledge from experience (Gentsch, 2019, p. 37-39).

The American computer scientist Tom Mitchell defined the term machine learning as follows: "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.* " (Mitchell, 1997, p. 2)

Machine learning is a research field in statistics, artificial intelligence and computer science. It deals with extracting knowledge from data. In the past, applications used hardcoded rules, like "if" and "else" decisions to adjust to the users' input or process with data. However, if there was a slight change in the task, the whole system might need to be rewritten. It was not flexible and not easily maintainable. What's more, a perfect understanding of the domain and how a decision should be made is required. This approach would not work for tasks like detecting faces in images In a program with a collection of faces that must be learned, an algorithm is required. Using machine learning, the program can determine what characteristics are needed to identify a face. Machine learning finds pattern where humans cannot find one, or doing so is too complicated. The goal of machine learning is to enable computers to learn on their own (Müller & Guido, 2016, p. 1-5).

3.5.1 Types of Machine Learning

Three different types of machine learning are distinguished. These are supervised learning, unsupervised learning and reinforcement learning. In the following section, we will gain a little insight into the basics of each type.

3.5.1.1 Supervised Learning

In supervised learning, the correct outcome is already known. This means the algorithm gets input data and also accurate output data. With this data set, the algorithm can learn. We will go more deeply into two types of supervised learning: the regression and the classification. Both types of supervised learning need input, output, and the task to be learned so as to get from the input to the output (Alpaydin, 2004, p. 1-4).

Classification:
The goal of classification is to identify a unique class. An example would be a risk classification for bank credits. There are the classes low-risk and high-risk. The input data would be the information on a customer. Then the algorithm has to decide whether it is a low-risk or high-risk credit. The output of the data is grouped. The data set needs to be categorical or discrete, e.g., map one discrete value to a categorical instance. (Alpaydin, 2004, p. 4-6).

Regression:
Regression is about estimating a continuous value. The regression model is about predicting a value based on independent features. The regression model allows predictions about values to be made. An example would be the predictions of the price of a car. Input variables would be attributes of the vehicle, like brand, year, mileage, etc. With the regression, the output would be a price prediction (Alpaydin, 2004, p. 8-9).

3.5.1.2 Unsupervised Learning

In contrast to supervised learning, unsupervised learning does not require any prelabeled target values. Instead, it is intended to independently identify commonalities in the data records and then form clusters or compress the data. Thus, it is about identifying hidden, unconscious patterns in data. Unsupervised learning algorithms can be used, for example, for a customer and market segmentation or clustering genes in genome research to reduce the number of characteristics (Gentsch, 2019, p. 38).

The learning algorithm in unsupervised learning only gets the input data and is asked to extract knowledge from this data (Müller & Guido, 2016, p. 131-134). We will look into the clustering algorithm in unsupervised learning.

Clustering:
Clustering tries to find a structure in data. Therefore, the clustering process forms groups in data. The data which belong to a group are more similar to data in that group (cluster) than to data in other clusters. Decreasing storage costs combined with the increase of the collection of data have resulted in a large mass of data (big data). With this big data, clustering has received more attention in the analysis of big data, so as to obtain better decisions and a safer decision-making process (Masulli, Petrosino, & Rovetta, 2012, p. 1-5).

3.5.1.3 Reinforcement Learning

An alternative to unsupervised learning are the models of reinforcement learning, in which learning patterns from nature are conceptually modeled. The combination of dynamic programming and supervised learning can solve problems that previously seemed unsolvable. The optimal solution must be found iteratively through trial and error. Useful approaches are promoted with rewards, and wrong steps are punished. The system can incorporate and react to a multitude of environmental influences in the decision-making process. Reinforcement learning belongs to the area of exploration learning, in which a system independently finds its solutions, apart from the trend-setting rewards and punishments that can differ significantly from the human-made solutions. Reinforcement learning received much attention after the victory of Google DeepMinds AlphaGo over Lee Sedol. Through reinforcement learning, artificial intelligence gains the ability to find new solutions independently and, at least seemingly, to act intuitively (Gentsch, 2019, p. 38).

3.5.1.4 Algorithms

For the sake of completeness, two well-known machine learning algorithms are briefly mentioned below.

Support Vector Machine:
The support vector machine (SVM) is a classification algorithm and was introduced to solving pattern recognition problems. It involves classifying with separation of the classes using a so-called margin. The separation line is chosen so that it touches the nearest examples on each site. If the margin is more significant, the classifier has higher chances to work well on future data. The nearest examples to the classifier are

called support vectors. This algorithm tries to identify the optimum of the support vectors to maximize the margin (Vapnik, 1998, p. 401-416).

Random Forest:
Random forests are a method for classification using ensemble learning methods. The decision trees are trained with random values sampled independently. All decision trees combine their results in the random forest. With the random forest approach, the generalized error is minimized. The minimization of the error depends on the strength of each tree. Random forest is a robust method for classifying and can also be used for regression (Breiman, 2001).

3.6 Artificial Neural Network

An artificial neural network (ANN) tries to create structures where intelligent behavior, is created based on the inspiration provided by the biological working of a brain. This information processing follows a bottom-up paradigm. Most of the incentives for this machine learning approach come from the fields of psychology or neurobiology (Gentsch, 2019, p. 35).

3.6.1 Structure and Functionality of a Biological Neuron

There are different types of neurons, but all of them transfer an electrical signal. The electrical signal goes from the dendrites through the axon to the terminals. The signal then goes from one neuron to another neuron. On this way, creatures process sound, light, heat, etc. Signals from sensory neurons are transmitted through the nerve system to the brain. The brain itself also consists of neurons. In Figure 3.1, we can see a sketch of a neuron from a pigeon (Rashid, 2017, p. 30).

The human brain has about 100 trillion neurons. A fruit fly has about 100,000 neurons, and a roundworm has 302 neurons. The fruit fly can accomplish intricate work like flying, eating, recognizing dangers and much more. The quantity of 100,000 neurons is in the capacitance range of a modern computer. A neuron works as follows: it takes an electrical signal as an incoming signal and releases another electrical signal. But the neuron does not send the signal immediately. Instead, it suppresses the release until the input signal is strong enough to force an outgoing signal (Rashid, 2017, p. 30-32).

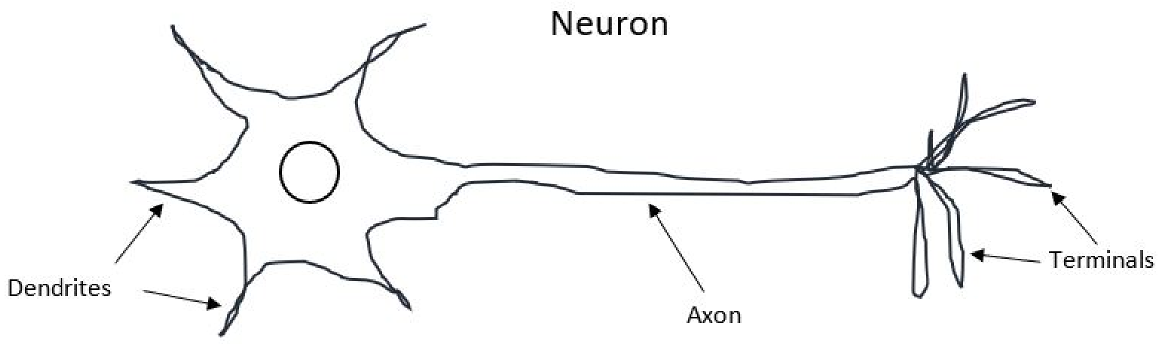


Figure 3.1: A neuron from a pigeon (Rashid, 2017, Cf. p. 30)

3.6.2 Layers and Forward Propagation

A type of a neural network is the so-called perceptron, which was developed by Rosenblatt (1958) and was inspired by McCulloch and Pitts (1943). A perceptron takes several binary inputs and produces one output. Each input is given a weight, which expresses the importance of the input relative to the output. The output is also binary. Thus, if the input is more significant than a certain threshold, the output is 1, otherwise it is 0. This shows how a perceptron can weigh up evidence to make a decision. Consequently, a network of perceptrons can make sophisticated decisions. Each neuron mentioned above can be replicated through a perceptron.

In Figure 3.2, we can see a multilayer perceptron (MLP) with the different layers of a neural network. The neurons can be seen in the circles, which are arranged in the hidden and output layers. For simplicity, the hidden layer consists of a single layer. However, the hidden layer can also include multiple hidden layers. Neurons at the same level do not communicate, but adjacent layers are fully interconnected (but not for every type of neural network, e.g., convolutional neural networks are not fully connected). Every neuron-to-neuron link represented as a line in Figure 3.2, has a weight. The weight of the connection from a neuron from the m-th neuron layer to the n-th neuron layer is denoted as w_{mn}. The first digit refers to the neuron in the beginning layer and the second digit to the neuron in the next layer (Kubat, 2017, p. 91-95).

The weight is present between every input and hidden layer, between hidden layers, and between the hidden and the output layer. The series of weighted incoming signals is summed up, a bias is added and the result is put into a function, like the sigmoid function. The outcome of this function is then used for the next neuron as an input (Müller & Guido, 2016, p. 105-107).

In the forward propagation, the input value goes from the input layer to the hidden layer, where the attribute values are multiplied with the corresponding weights.

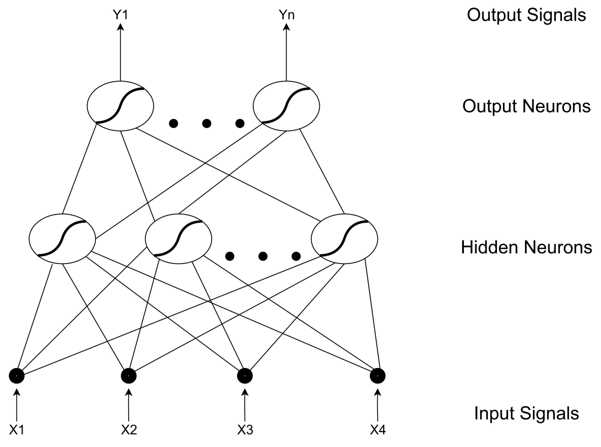


Figure 3.2: Neural network consisting of two interconnected layers (Kubat, 2017, Cf. p. 92)

Then the results are reproduced with the next weights to the next layer. This goes on until the output neurons are subjected to the activation function. And this is where the output values are obtained. With the right choice of weights and the number of hidden layers, classification problems can be addressed (Kubat, 2017, p. 93-95). The goal of the learning algorithm is to find the right weights and biases.

3.6.3 Activation Function:

A real biological neuron does not receive only a single incoming signal. It receives multiple signals. Therefore, the neuron aggregates all incoming signals. If the aggregated signal is not big enough, the neuron will not fire. However, if the aggregated incoming signal is big enough, the neuron will fire. The electrical signals are gathered from a neuron, and these signals are combined to fire a stronger signal. The fired neuron sends the signal along the axon to the terminals and the next neurons (Rashid, 2017, p. 33-36). Figure 3.5 shows the combination of multiple neurons.

The activation function takes the incoming signal, and after a certain threshold, it will generate an outgoing signal. Different activation functions are explained below.

Step Function:
The easiest way to show an activation function would be via a step function. In Figure 3.3, we see that the outgoing signal is null when the threshold is not reached. Above the threshold, the outgoing signal is generated. The generation of an outgoing signal is called " the neuron fires" in the literature. The characteristic of the step function is very close to a biological neuron.

Sigmoid Function:
An S-shaped function like the sigmoid function can represent a more graduated curve

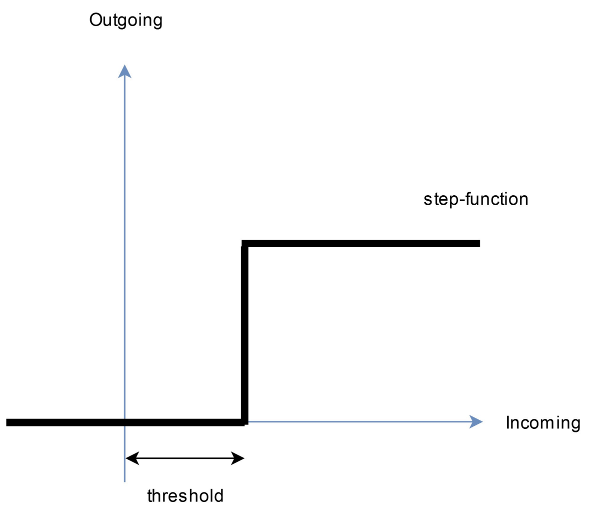


Figure 3.3: A simple step-function (Rashid, 2017, Cf. p. 33)

than the step function. In Figure 3.4, we see the sigmoid function, which is a more realistic representation of the natural firing process of a neuron - natura non facit saltus (nature does not make jumps). The sigmoid function is often used for binary classification in logistic regression models. The sigmoid function, also referred to as a logistic function, is defined as

$$y = \frac{1}{1 + \exp(-x)} \tag{3.13}$$

where y is the outgoing signal and x is the sum of the incoming signals (Rashid, 2017, p. 33-35).

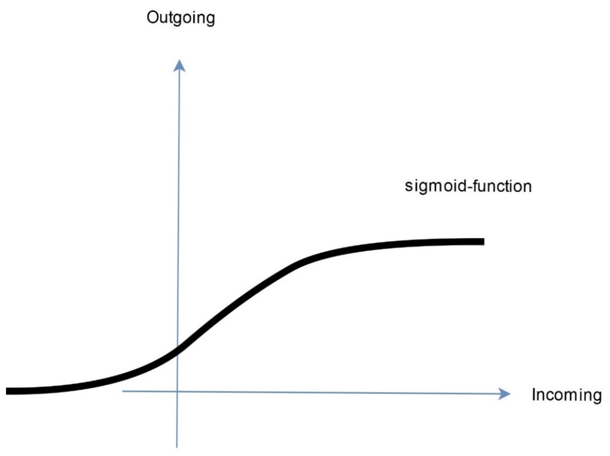


Figure 3.4: Sigmoid function as an activation function (Rashid, 2017, Cf. p. 34)

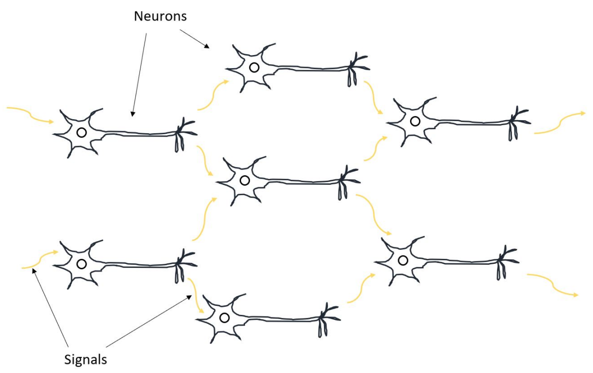


Figure 3.5: Several neurons connected to each other (Rashid, 2017, Cf. p. 36)

Rectified Linear Unit:
Another important activation function is the non-saturating nonlinearity rectified linear unit (ReLU). An ReLu is defined by Nair and Hinton (2010) as

$$f(x) = max(0, x) \tag{3.14}$$

A neuron only learns with an ReLU if there is at least one positive input. It has also been shown that networks train much faster with the ReLU function as compared to other functions. Thus, when the ReLU is compared to a sigmoid function, it is revealed that with a sigmoid function, the network learns much more slowly and the training is more difficult. The function gives 0 as an output when x < 0. If the input is x > 0, the result is a linear output with a slope of 1(Agarap, 2018). Figure 3.6 shows the ReLU function.

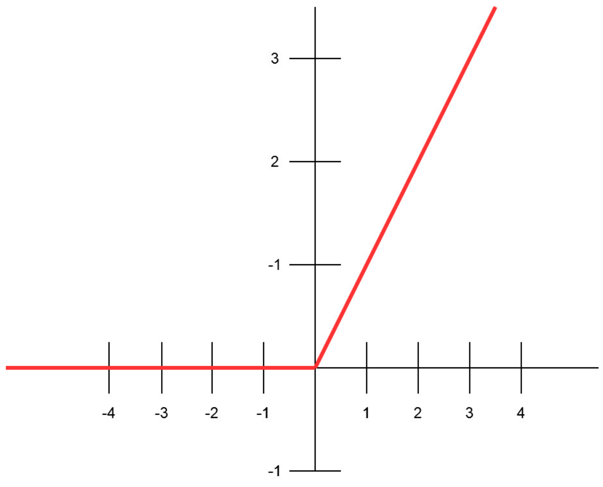


Figure 3.6: The ReLU activation function (Agarap, 2018)

Softmax:
This function was proposed by Bridle (1990). Softmax maps the output in the range

of 0-1. An important feature is that the total sum is 1. Therefore this function acts as a probability distribution. The equation for softmax is

$$\sigma(z)_j = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)} \tag{3.15}$$

where z is a vector of inputs to the output layer and j indexes the output units. This function is often used for multi-classification in logistic regression models and for the last layer.

3.6.4 Neural Network Errors

The error rate tells us about how often the network has failed its classification. The training data is sent to an MLP, and the forward propagation establishes the label. If the outcome of the MLP differs from the known class of the training example, this is considered an error. By dividing the wrong predicted results by the amount of training data, an error rate can be calculated. But with an outcome from the sigmoid function, we can measure the size of each error. The outcome with the highest value is chosen. This shows us the values for each class. Thus, we can differentiate according to how wrong a decision was (Kubat, 2017, p. 95-97).

Loss function:
The loss function (cost function, objective function) measures the estimation outcome of the network. This function puts all aspects of the model into a single number, which shows how well or poorly the network performs. An improvement of this number, in our case, the minimization of the error, is an indication of a better model.

As Reed and Marksll (1999, p. 155) stated, *"The cost function reduces all the various good and bad aspects of a possibly complex system down to a single number, a scalar value, which allows candidate solutions to be ranked and compared."*

Mean Squared Error:
The mean squared error (MSE) is often used for regression problems. The MSE is selected under the inference framework of maximum likelihood and a standard normal distribution of the target value.

The MSE is defined as

$$MSE = \frac{1}{n} \sum_{i}^{n} (y_i - \hat{y}_i)^2 \tag{3.16}$$

where y_i is the predicted outcome of the network, $\hat{y}_i$ is the desired outcome, and n is the number of data points. The result of the MSE is always positive. The perfect value would be 0. It adds up the squares of the differences, and this gives the total cost of the network.

Cross Entropy:
The cross entropy between the training data and the models is a well-used cost function for binary classification problems. The cross entropy is the difference between two probability distributions p and q. The equation is defined as

$$H(x) = \sum_{i}^{n} p(x) log q(x) \tag{3.17}$$

where p is the predicted outcome and q is the desired outcome. The best use of the cost function is if the data is normalized between 0 and 1 (Goodfellow, Bengio, & Courville, 2016, p. 173-176).

3.6.5 Backpropagation of Error

The task of artificial intelligence is to solve complex business problems with big data by finding the right weights for the network to optimize the classification performance. This is due to that the fact that the weights affect the network's behavior. The scenario in principle is that the initial weights are small random numbers. Then the training data is forward propagated. The discrepancy between the output and the target vector modifies the weights. After the modification, the next example will be propagated forward. An epoch is completed after the last training example is reached. Thes training can require many epochs to be successful. (Kubat, 2017, p. 97-101).

The backpropagation algorithm from Rumelhart and McClelland (1987) is a computation method that lets a NN use an optimization algorithm, like the gradient descent. It allows feedback of information to compute the gradient.

Backpropagation attempts to minimize the cost function. The question is, what effect does a training example have on the adjustment of the weights and biases. The output of a neuron, as mentioned above, consists of the weighted sum of all activations in the previous layer, plus a bias, and the activation function. Thus, there are three different parameters which influence the output: the bias, the weight and the activations from the previous layer (Nielsen, 2018).

The Hebbian theory, from Hebb (1949), says that neurons that fire together, wire together. This means that neurons with the most prominent connections and weights between neurons are the most active and are also the ones that should be the most active. The activations from the previous layer are by proportional changes the size of the corresponding weights. There is no direct influence of the activation, but it can be controlled via the previous weights and biases.

The backward propagation idea is about adding together all desired effects and getting a list with the change rate in the second-to-last layer. With this list, it is possible to recursively apply the same process to the weights and biases that determine those values. The same process is repeated through moving backwards of the layers in the network. This backpropagation routine has to go through the training data. Every change in the weights and biases is recorded and they are averaged. This average change is the negative gradient of the cost function (Nielsen, 2018).

Gradient Descent:
The goal is to minimize the cost function to get a more precise outcome. Gradient descent tries to decrease the loss of the function most quickly. The process of repeatedly nudging an input of a function is called gradient descent. Each component of the negative gradient provides information about whether the corresponding input vector (weight) should be increased or decreased and also to what extent it should be changed. The learning of a network deals with which weights and biases minimize the cost function (Nielsen, 2018).

A hyperparameter which should be mentioned here is the so-called learning rate. The gradient descent calculates the error gradient of the model. Then the model is updated. The updated amount during training represents the learning rate. The learning rate is in the range between 0 and 1. Thus, the learning rate tells how much the weights in the network are adjusted. As the value approaches zero, the process of the downward slope becomes slower.

3.6.6 Generalization Methods

The concept of generalization is about learning from data and correctly applying the "knowledge" on other data. For the NN, this means learning as effectively as possible with the training data.

Underfitting:
This occurs when the model has not learned well enough to get a good prediction on the test data. Underfitting occurs when the complexity of the model is too limited. This phenomenon indicates a large error in the training data.

Overfitting:
Overfitting is a problem in the machine learning field. This refers to the model when it learns the training data too precisely and is not able to generalize data for the prediction. This means the network can only perfectly predict the training data but is

not able to do it correctly for the test data. This implies a low error rate on the training data and a high error rate on the test data. This problem often occurs when the complexity of the model is too great. Preventing overfitting in neural networks can be done with more training data, weight regularization, less capacity of the network, early stopping, dropout, batch normalization and less capacity of the network.

Weight Regularization:
This method adds extra cost to the cost function when large weights are resulting. Thus, the model is penalized for having large weights. This leads to a regular order of weight distribution.

Dropout:
This method drops out single neurons with a certain probability. Thus, there are neurons which will not be connected to other neurons. This basically involves training many smaller subset networks. But they change continuously because of the randomness of the dropout neurons. In the end, there will be an average of the overall network (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

Occam's Razor Principle:
This principle has two explanations. The first one is, "Given two models with the same generalization error, the simpler one should be preferred because simplicity is desirable in itself." The second razor principle says, "Given two models with the same training-set error, the simpler one should be preferred because it is likely to have lower generalization error." (Domingos, 1999)

Batch Normalization:
This is an optimization strategy and was developed by Ioffe and Szegedy (2015).

If every training example and gradient descent step is added up, it takes a long time to train the network. In this case, the training data is shuffled and divided into minibatches. Thus, the gradient descent is not calculated for every single training example. Instead, the gradient descent of a minibatch is calculated. After each minibatch, the weights and biases are adjusted. This technique is called stochastic gradient descent. It also speeds up the training time of the network (Nielsen, 2018).

Holdout Selection:
The data is split into a training set, validation set and holdout set. The training set is provided so that the machine learning algorithm can learn, and these models are applied to the validation set. With the validation set, a check is made to see how well the model has been trained and how accurately model properties are estimated. With the holdout set, also called the test data, the generalization error is measured. The generalization error measures the predictive power of a model.

Generalization Error Measure:
The model inaccuracy, sampling error and noise are some of the reasons for a discrep-

ancy between the prediction and the observed data. The bias-variance decomposition is defined as follows

$$Error(f_{\hat{\theta}}) = Bias(f_{\hat{\theta}}) + Variance(f_{\hat{\theta}}) \tag{3.18}$$

where $\hat{\theta}$ is a random variable. In this equation, it can be seen that the variance and the bias are dependet- on the complexity and are part of the error (Spilak, 2018, p. 23).

Early Stopping:
This is a very efficient hyper-parameter selection algorithm. It is a simple method for reducing training time. Early stopping means stopping the training when a monitored quantity is no longer improving. If the loss is increasing or the accuracy is decreasing, the training will be stopped (Yao, Rosasco, & Caponnetto, 2007).

3.7 Related Work

This chapter provides a deeper insight into the research in the field of cryptocurrencies, volatility prediction, the use of NNs for forecasting prices and the forecasting of volatility. There is already a good deal of research in the field of volatility forecasting and predictions with different ML techniques.

3.7.1 Research Specific to the Cryptocurrency Market

Different studies examine the market efficiency of cryptocurrencies, the volatility characteristics, liquidity, microstructure, the risk and speculative aspects.

Since the publication of Nakamoto, Satoshi (2008) and the birth of Bitcoin, cryptocurrencies have gained significant attention in the financial and research fields.

There are already many papers focusing on the specifications of the cryptocurrency market. The broad financial research focuses on market efficiency, volatility, pricing bubbles and the impact of news announcements or name-dropping in social media networks. An interesting question is if cryptocurrencies, and specifically Bitcoin, behave in a manner that is similar to any official currency.

The paper of Corbet, McHugh, and Meegan (2017) reports finding that the volatility of the Bitcoin is influenced by the interest rate and quantitative easing announcements. This is surprising because the Bitcoin is decentralized and independent of government influence.

Gajardo, Kristjanpoller, and Minutolo (2018) were researching the cross-correlations between the classical financial fields, like currency rates, indices, commodities, and cryptocurrencies. The Bitcoin shows a different relationship to indices and commodities.

The statistical properties and the multifractality of the Bitcoin were examined by Takaishi (2018). Based on 1-min return, they found that the distribution is leptokurtic, and the kurtosis approaches the Gaussian expectation very slowly, which means a sampling period of about a few weeks is needed. The skewness is negative daily and becomes zero when the period is extended. In their research, the volatility of the Bitcoin shows the typical volatility clustering but no volatility asymmetry.

Begusic, Kostanjcar, Stanley, and Podobnik (2018) were examining that idea that the price fluctuation of the Bitcoin price distribution is heavy-tailed and follows a power-law distribution. These findings support the covariance-based risk measurements approaches.

The volatility connectedness between eight cryptocurrencies was considered by Yi, Xu, and Wang (2018). They found that the connectedness in the volatility fluctuates periodically. The volatility connectedness increases when there are unstable market conditions or shocks. What is more, the connectedness of volatility is not necessarily linked to the market capitalization.

Lahmiri, Bekiros, and Salvi (2018) investigated the nonlinear patterns of volatility in Bitcoin markets. The empirical outcome shows that a significant underlying long-range memory can be found in all distributional assumptions and volatility series.

The liquidity uncertainty of the Bitcoin was examined by Koutmos (2018b) using the bid-ask spread. The findings were that rising volatility is related to liquidity uncertainty. If the trade volume and the market capitalization increase, this is associated with less liquidity uncertainty.

Osterrieder and Lorenz (2017) were examining the statistical risk assessment and the tail behavior of the Bitcoin. This research showed that the Bitcoin has more volatility and heavier tails than FIAT money. The volatility of the Bitcoin is about 6-7 times greater than in traditional currencies. The tails of the Bitcoin also show a higher level of extreme events compared to FIAT money.

Corbet, Lucey, Urquhart, and Yarovaya (2019) were analyzing the research on cryptocurrencies from 2009 to August 2018. This paper explains the trilemma of cryptocurrencies and their resulting excessive volatility. The three interrelated issues are regulatory alignment, the potential for inherent bubbles and cyber criminality. Furthermore, they have identified several gaps in the literature which have not been fully acknowledged regarding cryptocurrencies. These are environmental challenges, asymmetrical information issues, alternative uses of the blockchain, and cryptocurrencies as part of a diversified portfolio.

The microstructure of cryptocurrencies and in particular that of the Bitcoin was examined in an empirical study by Koutmos (2018a). The Bitcoin return and the transaction activity are linked. This also leads to the assumption that the volatility of such assets is hardly affected by the volume.

Balcilar, Bouri, Gupta, and Roubaud (2017) were checking if the Bitcoin trading volume can be used to predict the return and the volatility. They found that with volume as input, it is possible to predict returns in a functioning market around the "normal" mode without any bearish or bullish movements. In their opinion, prediction of the volatility using the trading volume does not work.

The paper of Pichl and Kaizoji (2017) investigates volatility using empirical studies of Bitcoin prices. They found that the Bitcoin price to US Dollars (USD) is more volatile than that of the Euro to the USD. What is more, the prediction of the logarithmic returns of the Bitcoin with the use of a standard NN approach is capable of reproducing the clustering feature and the shape of the distribution.

3.7.2 Volatility Forecasting without ML

In the financial field, the forecasting of volatility has been the focus of a good deal of empirical and theoretical research. As mentioned in section 3.3.1, the ARCH model from Engle (1982) and the extension to the GARCH model from Bollerslev (1986) has been critical in analyzing volatility. The popularity of these approaches is their flexible structure which allows accommodation of the facts from financial time-series, e.g., volatility clusters. Due to extensive research on forecasting volatility with the GARCH model, several different extensions have resulted.

The exponential GARCH (EGARCH) by Nelson (1991) deals with keeping the non-negativity constraints based on the volatility and alleviating the positivity constraints.

Another famous model is the Runkle, Glosten, and Jagannathan GARCH (GJR-GARCH) approach proposed by Runkle, Glosten, and Jagannathan (1993). The GJR-GARCH model has the addition of a leverage effect on the vanilla GARCH. Therefore, it uses an unconstrained maximization as the maximum likelihood. The GJR-GARCH is a richer model compared to the vanilla GARCH. Both approaches show a more precise volatility forecast than the standard GARCH model empirically.

However, there are papers which support the EGARCH model such as Lee (1991), Cao and Tsay (1992) and Heynen and Kat (1994). On the other side, there are Brailsford and Faff (1995) and Taylor (2004) who support the GJR-GARCH as a better approach to volatility prediction.

Granger and Poon (2004) examined 72 working papers about volatility forecasting and searched for comprehensive insights. They split the literature into historical price information only and the implied volatility in option prices. In research papers, scientists use different intervals, which makes it hard to compare the results. What is more, there are many studies which are done by workers or students, trying to make a specific technique or viewpoint look good. Thus, the academic review also includes unequal results. There is also a lack in the literature concerning the identification of the most accurate model for predicting volatility.

3.7.3 Price and Interest Rate Prediction with Neural Networks

In the following, we will gain deeper insights into the use of NNs for predicting stock and cryptocurrency prices.

Sin and Wang (2017) were predicting daily Bitcoin prices for the next day using an ANN ensemble approach. For the NN ensemble, they created 5 MLP with different numbers of nodes in the layer, but with the same specific actions. For the next day forecast, they used the data from 780 days, where 730 days were for training and the remaining 50 days for backtesting. The output of each MLP was either zero, which indicated a price drop, or one, which meant a price increase. In the backtesting, this method achieved an accuracy of 64% (34 correct predictions out of 50).

Lahmiri and Bekiros (2019) were using a chaotic NN for the price prediction of Bitcoin, Digital Cash and Ripple. They built a NN to extract the hidden information from the linear or nonlinear pattern within the data. The NN is a Long-Short Term Memory (LSTM) deep learning NN. This empirical study showed that cryptocurrencies demonstrate chaotic characteristics and a strong self-similarity in the sub-samples. The LSTM NN outperformed the generalized regression NN. Deep learning neural systems can memorize long and short-term information and learn fractal patterns. The overall outcome was that deep learning approaches can be more efficient in learning and forecasting patterns for cryptocurrencies.

Forecasting of the exchange rate from Bitcoin to USD using four ANNs was examined by Radityo, Munajat, and Budi (2017). In their study, they compared different training methods for NNs. They used a backpropagation NN (BPNN), genetic algorithm NN (GANN), genetic algorithm backpropagation NN (GABPNN), and neuroevolution of augmenting topologies (NEAT) as models. The experiments showed that the GABPNN approach achieves the best forecast accuracy with a mean absolute percentage error (MAPE) of 1.883%. However, the BPNN approach is three times faster, and the difference in the MAPE to GABPNN is about 0.115%. This shows that both methods work well for the price prediction of Bitcoin to USD with the given data in the study.

Lei (2018) used the Wavelet NN (WNN) and the Rough Set (RS) to predict stock prices. The RS can reduce input dimensions and therefore optimizes the structure of the data for the WNN approach. The WNN improves the performance of the prediction through the combination of the time-frequency localization characteristics and the learning function of the NN.

Adhikari and Agrawal (2014) follows a combination of the random walk (RW) and the ANN approach for the modeling and the use of forecasting. The RW is used to process the linear part of the data set. For the nonlinear residuals, they used an ensemble of feed-forward ANN (FANN) and Elman ANN (EANN). They tested each model alone and with their developed hybrid model. The exogenous variables given the hybrid model were the exchange rate for USD-INR and GBP-USD, the S&P500 daily closing prices and the IBM stock price. The hybrid model achieved the best outcome overall for every series.

To predict the stock exchange rate, Guresen, Kayakutlu, and Daim (2011) used MLP, dynamic ANN and a hybrid GARCH-ANN model. The outcome of these models was measured with the mean squared error and the mean absolute deviate. In this research, Guresen et al. (2011), focused on the NASDAQ stock exchange with data from October 2008 to June 2009. In total, they used data from 182 days, where about 80% of the data was for training and cross-validation, and the rest for testing. In this study, the MLP outperformed the dynamic ANN and the GARCH-ANN.

Roh (2007) focuses on the volatility forecast of the stock price index. Roh used hybrid models with NN and time series models to predict the deviation and direction of the stock price. For their study, they used KOSPI 200 time-series data and KSE KOSPI 200 option daily trading data. The models used were NN, NN-EWMA, NN-GARCH and NN-EGARCH, where the hybrid models were compared with the classical NN. The NN-EWA model gives more weight to recent data and is, therefore, more suitable for short-term volatility predictions. The comparison between the models is made using mean average error and the direction (hit ratio) of the realized volatility. The NN-EGARCH model was the most successful with respect to the mean average error. This is due to its ability to use leverage and leverage effect as input variables. The comparison of the hit ratios shows that the NN-GARCH and the NN-EGARCH models show the best outcome as compared to the NN. Roh also showed that 20-25% of the learning of NNs came from the GARCH or EGARCH variable. The rest of the learning comes from other correlated variables. The maturity of the learning rate (75-80%) comes from fundamental analysis, like bond prices, bond yields, volume, premiums, etc.

3.7.4 Volatility Forecast with Neural Networks in the Classical Financial Markets

Donaldson and Kamstra (1997) found that GARCH models cannot model nonlinear effects of volatility and therefore used an ANN approach, which leads to better forecasting. This is because of their ability to capture asymmetric volatility effects. There are now several studies which seek to a hybrid model for forecasting the volatility in the stock market.

Bildirici and Ersin (2009) used different GARCH models and extended them with ANN models (MLP) for stock exchange volatility forecasts.

Another combination of NN and GARCH models was made by Monfared and Enke (2014). In their study, they combined a NN with a GJR-GARCH model to predict the volatility. The testing of the NN with the GJR-GARCH model included a feed-forward with backpropagation, a generalized regression and a radial basis function.

Kristjanpoller and Hernandez (2017) focused on the prediction of the volatility for metals with an ANN-GARCH model with regressors. They used the GARCH model prediction as input for the ANN with additional explanatory variables to improve the prediction power. These explanatory variables were different stock market indices, currency rates and the oil price. In their study, they used an ANN with two hidden layers and five neurons. The input variables given the NN were the explanatory variables as mentioned above, the forecast of the GARCH model and the squared metal price return. This study showed that a hybrid NN improves the prediction of volatility in the given market and period as compared to the GARCH model.

Kristjanpoller and Minutolo (2015) were forecasting the volatility for the spot and future markets of the gold price using an ANN-GARCH model. The benchmark for this approach is the GARCH model. The explanatory variables given the model were exchange rates, stock market indices and the oil price. With the ANN-GARCH approach, the study showed a reduction of 25% of the MAPE in the spot market and 38% for the future price. The ANN makes it is possible to improve the prediction accuracy of the volatility. What is more, with this approach, it is possible to detect the variables which influence the volatility. However, it should be noticed that the increase in the number of variables does not necessarily improve the performance of the model.

Volatility forecasting with the use of a hybrid NN-GARCH model for Latin-American stock exchange indices was researched by Kristjanpoller, Fadic, and Minutolo (2014). The input variables given the NN were the realized volatility, the output of the GARCH model, the squared index log return and the index return. The input variables were removed from the NN stepwise. The NN was a MLP with ten neurons and one hidden layer. The result was tested using different error measures, like the mean square error, root mean square error, mean absolute deviation and the MAPE. The best outcome was shown by the NN-GARCH model using all four input variables.

Lu, Que, and Cao (2016) also worked with a hybrid NN-GARCH model for predicting energy price volatility. In this study, they were testing ANN-EGARCH and ANN-GJR-GARCH models, as well as EGARCH-ANN and GJR-GARCH-ANN models. For the second type of models, the output of the ANN acts as an input variable for the GARCH model. They were working with Chinese energy data. The result showed that the EGARCH-ANN model had the best performance in the given market. This indicates that the model is better because of the fit of the heteroscedasticity condition of log returns.

Hajizadeh, A., Fazel Zarandi, and Turksen (2012) were predicting the volatility of the S&P500 index return. For the forecast of 10 and 15 days, they used two hybrid models of the EGARCH model and an ANN. As input variables for the first model, they chose the index price, squared price, return, squared return, volatility from the GARCH model, one-day lagged volatility and the traded volume of the index. As exogenous variables, they included the prices of other indices, treasury yields and exchange rates. The second model used multiple simulated volatility series. The input variables were the same as in the first model. In this study, the second model provided better forecast performance than the first model with the best EGARCH model.

Hamid and Iqbal (2004) were forecasting the volatility of S&P 500 index futures prices with the help of a NN. As data for the prediction, they used futures contracts and indices. The futures contracts were a mixture of commodities, treasury obligations and foreign currencies, and the indices were DJIA, NYSE Composite and S&P 500 index. What is more, the explanatory variable they used was the one-day lagged S&P 500 future price. From this raw data, they calculated the 20-day rolling historical standard deviation (HSDs). The forecast accuracy was measured using the mean of absolute errors and the root mean of squared errors of the volatility prediction to the realized volatility. They used three different forecast horizons in their study: a 55-day forecast, 35-day forecast and a 15-day forecast. As an input variable, they took every variable that was either correlated by greater than 5% or less than -5%, or which had a high relative contribution (greater than 0.07). This made a total of 13 explanatory variables for the NN. The NN outperformed the benchmark in all three forecast classes, except the 15-day forecast with the root mean of squared error.

3.7.5 Volatility Forecasts with Neural Networks in the Cryptocurrency Market

The research by Kristjanpoller and Minutolo (2018) focused on the use of a hybrid volatility forecast model integrating GARCH, ANN, technical analysis and principal components analysis in the field of cryptocurrencies. The challenge in this study was the highly volatile Bitcoin series compared to FIAT money. The analyzed data for this study was the Bitcoin price in USD from September 2011 to August 2017. The return was calculated as the log return. The BTC return in the given period shows a leptokurtic distribution and negative skewness. What is more, heteroscedasticity exists in the data, which confirms the nonlinearity of the volatility in BTC.

As input variables, they used seven exogenous variables derived from the BTC closing prices series. These are momentum 12 (M12), moving average convergence divergence (MACD), MACD line, relative strength index (RSI), stochastic RSI (S-RSI), Williams percentage 14 days and Bollinger upper bound (BUB). Furthermore, a percentage variation transformation (DTA) was performed on these seven technical indices. A PCA algorithm was conducted using the technical indices and the transformation. Using these 14 indicators, better prediction performance with the MLP was expected.

For the forecasting process, they used twelve models with different combinations of models and inputs. For the comparison, they used the mean square error as the loss function. For the analysis of the volatility results, they used three different GARCH models. These are the GARCH base, EGARCH and APGARCH models. The best outcome of the GARCH models was shown by the EGARCH model. Only three technical indicators were chosen as the input for the EGARCH model. This was due to the high correlation between some of the input variables, so only the M12, S-RSI and BUB were taken as indicators.

For the hybrid models, networks with 1-4 hidden layers, 5-20 (step size 5) neurons for a period of 10, 22 and 44 days were tested. Models 1-3 were MLP hybrid models with only econometric inputs, models 4-8 were the MLP with econometric and technical indices as input and models 9-12 used the incorporated preprocessed technical indicators with the PCA algorithm. All the hybrid models outperformed the mean square error as compared to the best GARCH model (EGARCH). The model confidence set (MCS) was used for the comparison of which models had the lowest mean square error and were statistically significant.

Thirteen models were statistically significant for the 10-day volatility forecast. Out of these 13, there were 11 models which included the technical indicators, or the PCA and the technical indicators. For the 22-day volatility forecast, there were 20 models which were statistically significant, and none of them were without technical indicators. For the 44-day volatility forecasting, eight models were significant. Only one model out of the eight significant models was without technical indicators or PCA.

The research paper by Miura, Pichl, and Kaizoji (2019) focuses on volatility prediction in cryptocurrency time series with the use of an ANN. They used the heterogeneous autoregressive model of realized volatility (HARRV) with three lags, MLP, convolutional NN (CNN), LSTM, gated recurrent unit (GRU), SVM and ridge regression as their method. For the LSTM, GRU they implemented a dropout mechanism, and the MLP and CNN had the batch normalization.

For data, they used high-frequency Bitcoin to USD prices. The data length is about six years, where they aggregated a minute-based log return and thereon the realized volatility in 3-hour intervals. For the model testing, they used 10-fold cross-validation with 100 runs on each, and the benchmark used 30 random weights for the test set. For the loss function, they used the mean square error and the root mean square error.

The best model in their study was the ridge regression and the GRU model. However, also the HARRV, MLP and LSTM and CNN were very close to the ridge regression and GRU model with their outcomes. Only the SVM had the worst result of all models for their data set.

It was also mentioned that most financial studies from the classical field use daily data. However, in the cryptocurrency field, there is a continuous market 24/7. This is why they chose a 3-hour interval to sample the realized data.

3.7.6 Overfitting of Out-Of-Sample Performances

In their research paper , Bailey, Borwein, Lopez de Prado, and Zhu (2014) focused on the overfitting of investment simulations and the effects on out-of-sample (OOS) performances. They assert that probably most of the published backtests are overfitted. Most of the papers do not indicate the number of trials made. They concluded that the in-sample (IS) Sharpe ratio should be greater when more trials are needed to select a strategy. Generally, increased backtesting leads to parameter tuning, and this will increase the overfitting of the models. They also mentioned that if financial advisors do not control overfitting, their results will be much lower or even negative as compared to the backtesting performance.

Wiecki, Campbell, Lent, and Stauth (2019) researched trading strategies which used backtests with historical data. The main goal was to find out if ML approaches often tend to overfit in the backtest and can, therefore, lead to a wrong conclusion in most research papers. Furthermore, the relationship between the performance results using IS and OOS data sets is compared with metrics from the literature.

For their research, they used 888 US equities algorithmic trading strategies which were developed and backtested on the Quantopian platform from 2010 to 2015 with at least six months of OOS performance. The trading algorithms for this study included different strategy styles, trading frequencies, portfolio structures and portfolio sizes. They filtered duplicates, outliers and algorithms which have no meaningful representation power out of the data. These measurements led to the number of 888 trading algorithms mentioned earlier.

The analysis of the data was carried out with linear regression, Bayesian linear regression and nonlinear regression methods. Furthermore, to achieve a cross-sectional comparison, they extracted different universal features such as risk and performance metrics, descriptive statistical data, e.g., Sharpe ratio, alpha, beta, annual returns, volatility, skewness, kurtosis, standard deviation, median, etc. They also included a feature to capture overfitting by measuring the total number of backtest days.

For the nonlinear regression methods to predict the OOS Sharpe ratio, they used random forest and gradient boosting. These tests were made with 5-fold cross-validation, and 20% of the data were for a holdout set to analyze the performance.

They started with the analysis of the correlation of the IS performance metrics and the OOS period. A weak negative correlation (significant) was found between the annual return of the IS and OOS periods. The Sharpe ratio was positively correlated between IS and OOS. Of interest is the fact that no performance metrics showed a significant correlation between the IS and OOS periods.

The negative correlation for the annual return and the positive correlation in the Sharpe ratio for IS and OOS is surprising. It turned out that this was due to an interaction of mean returns and volatility in the OOS Sharpe ratio. This is also validated by a high correlation between overfitting and volatility. What is more, risk and performance metrics do not come from the average returns. The measurements of standard deviation and maximum drawdown are stable across the IS and OOS periods.

Another finding that leads to a backtest overfitting is that when a user ran more backtests it was more likely that a significant outcome would be achieved. There is also a greater difference between IS and OOS performance.

Generally speaking, the OOS performance for the trading strategies was low. This could mean that it is not possible to forecast a profitable trading strategy from historical data. However, ML approaches can predict OOS performance with a well-known selection of features, better than other measures. They also found that essential elements for the prediction are the quantification of higher-order moments, skewness and kurtosis. These features can be taken from backtesting, but not in a univariate or linear way.

One of the limitations of their research is that it primarily employed algorithms used by Quantopian users. There is uncertainty as to whether the same would hold true for backtesting performed exclusively by quantitative finance professionals. These professionals might use methods for reducing overfitting in their strategies.

Another limitation of the result is the overlapping periods of the algorithms in the given sample. The IS period was from 2010 to June 2015 with a bull market and low volatility. In the OOS period from June 2015 to February 2016, there was a flat-to-bear market with higher volatility. Thus, it is unclear if the market would have continued its performance and that the backtests would have continued to perform as during the IS period.

3.7.7 Distinction

This master thesis differs from the papers mentioned above for various reasons. This thesis calculates a risk premium for a short-term period of cryptocurrency price fluctuation in the secondary market. To achieve this, we defined a risk premium calculation method using the ES approach. The ES needs the standard deviation as an input variable. Thus, the prediction of the future standard deviation (volatility) is the primary research focus.

The Main Points of Distinction:
First, we are focusing on short-term volatility prediction. This means that the volatility prediction deals with a period shorter than daily, which is used in most studies. Second, we are targeting the cryptocurrency market. Cryptocurrencies are highly volatile compared to FIAT money. What is more, almost all current papers on cryptocurrencies only include the BTC to USD in their research. As trading pairs, we are using cryptocurrencies to traditional currencies as well as cryptocurrencies to cryptocurrencies. Third, to predict the volatility, we are using NN models and different input variables to test it.

The three points mentioned above are very seldom taken in their own right, but when combined, they make the research unique.

4 Data

The data used for this thesis were acquired from a US-based cryptocurrency exchange named kraken.com (Kraken).

Kraken is one of the five biggest cryptocurrency exchanges worldwide, with a market share of 15% and an average trading volume of about 500K BTC in May 2020, according to Kacper Ciesla (2020). The data were acquired from Kraken via API for the symbols:

- BTC-USD
- BTC-EUR
- ETH-BTC
- ETH-USD
- ETH-EUR

The acquisition period started on the 11^{th} of October 2019 at 23:59, and the last observation was on the 15^{th} of April 2020 at 00:01. All trades and the order book in one-minute time steps were acquired from Kraken. The order book data (OB data) table for the period includes 267,842 rows (minutes in the given period) and 203 columns. The columns are timestamp, broker, symbol and the first 50 ask and bid prices and amounts in the order book. The trading data (TR data) columns are timestamp, broker, symbol, trade side, price and amount. The data rows (trades) differ for each symbol because they had different numbers of trades.

The first step with the data was pre-processing to make them useful for further usage. The OB data were given the timestamp as an index, and the mid-price for each minute was calculated. The mid-price is made up of the first ask price plus the first bid price from the OB divided by two. Over the whole acquisition period, only a minimal number of minute data points were missing. This is why we did a linear interpolation for the missing mid-price data points between the closest previous and subsequent data point.

As a next step, the returns were calculated as the percentage change of the last mid-price to the next mid-price. What is more, the log return was calculated with the logarithm of the mid-price and then the return was calculated. The logarithm return is used in the quantitative finance field for calculations with the return. The outcome of the OB pre-processing is the timestamp for each minute with the mid-price, return and log return. All other columns were deleted.

The pre-processing of the trades included setting the timestamp as an index. Then the columns "broker" and "symbol" were deleted. The data in the "trade side" column were changed from the string entries like 'TradeSide.SELL' and 'TradeSide.BUY' into an integer of 0 for sell and 1 for the buy side.

4.1 Data analysis

In the following section, we will take a closer look at the data specifics for the pairs BTC-USD, ETH-USD and ETH-BTC.

Table 4.1 presents the series of descriptive statistics for the mid-price and the return of the three trading pairs.

The average BTC-USD price in the period was about $ 8,026.67, which is lower than the median of $ 8,061.10. This is because the BTC price until March was relatively higher than in the last previous weeks. A massive price drop at the beginning of March dropped the average price below the median. For the pairs ETH-USD and ETH-BTC, the average price for the given period was higher than the median. In both cases, the price at the beginning of the period was relatively low and experienced a rally from the beginning of February until March.

In general, the standard deviation for the mid-price and the return shows a volatile market for all pairs as compared with the data from the classical financial market.

The skewness values of the return for the pairs BTC-USD and ETH-USD are positive, which shows a right-skewed distribution. This indicates that the mean is on the right side of the median and has a longer tail in the positive direction. The negative skewness value of ETH-BTC indicates a left-skewed distribution. The left-skewed distribution has the mean on the left of the median and a longer tail in the negative direction.

The kurtosis of the returns shows a very high value, which indicates a leptokurtic distribution. This distribution leads to a high peak and fat tails.

Table 4.1: Descriptive statistics of price and return for BTC-USD, ETH-USD and ETH-BTC.

	BTC-USD	ETH-USD	ETH-BTC
Min mid-price	3946.6	90.1	0.017
1st quarter mid-price	7194.9	141.59	0.019
Median mid-price	8061.1	165.5	0.021
Mean mid-price	8026.67	171.46	0.021
3rd quarter mid-price	8954.75	185.93	0.022
Max mid-price	10504.75	288.99	0.028
Min return	-3.9 %	-6.22 %	-3.47 %
1st quarter return	-0.022 %	-0.034 %	-0.023 %
Median return	0	0	0
Mean return	0.0000002 %	0.000005 %	0.000005 %
3rd quarter return	0.022 %	0.034 %	0.023 %
Max return	7.86 %	8.04 %	2.86 %
Mid-price standard deviation	1186.8	39.63	0.0026
Return standard deviation	0.12 %	0.14 %	0.077 %
Mid-price skewness	-0.17	0.98	0.81
Return skewness	2.93	1.09	-0.57
Mid-price kurtosis	-0.53	0.32	-0.057
Return kurtosis	280.27	213.18	113.37

4.1.1 Data Plots

The price return is calculated as mentioned above. The historical volatility (HV) is calculated using 10-minute rolling data with analyzed windows of five hours, 24 hours and five days.

Figure 4.1 shows the evolution of the BTC-USD price, return, and volatility for five hours, one day and five days. The price for BTC between October and the beginning of March was in the range of \$ 6,500 and \$ 10,500. In the middle of March, the price declined to less than \$ 4,000 per BTC. Then the BTC started an upward trend to \$ 7,000.

The returns clearly show a relatively tight band from October to March. As of the middle of March, there are more ups and downs.

The volatility figure allows us to see the increasing volatility in March. This tends to show the volatility clusters in the cryptocurrency market. Until March there is less volatility, and from the middle of March, the volatility increases for a specific time. This clearly shows the heteroscedasticity in this series. We can also see that the volatility calculation period has an impact on the volatility rate. When we use a more extended time range for the volatility calculation, the volatility is flattened.

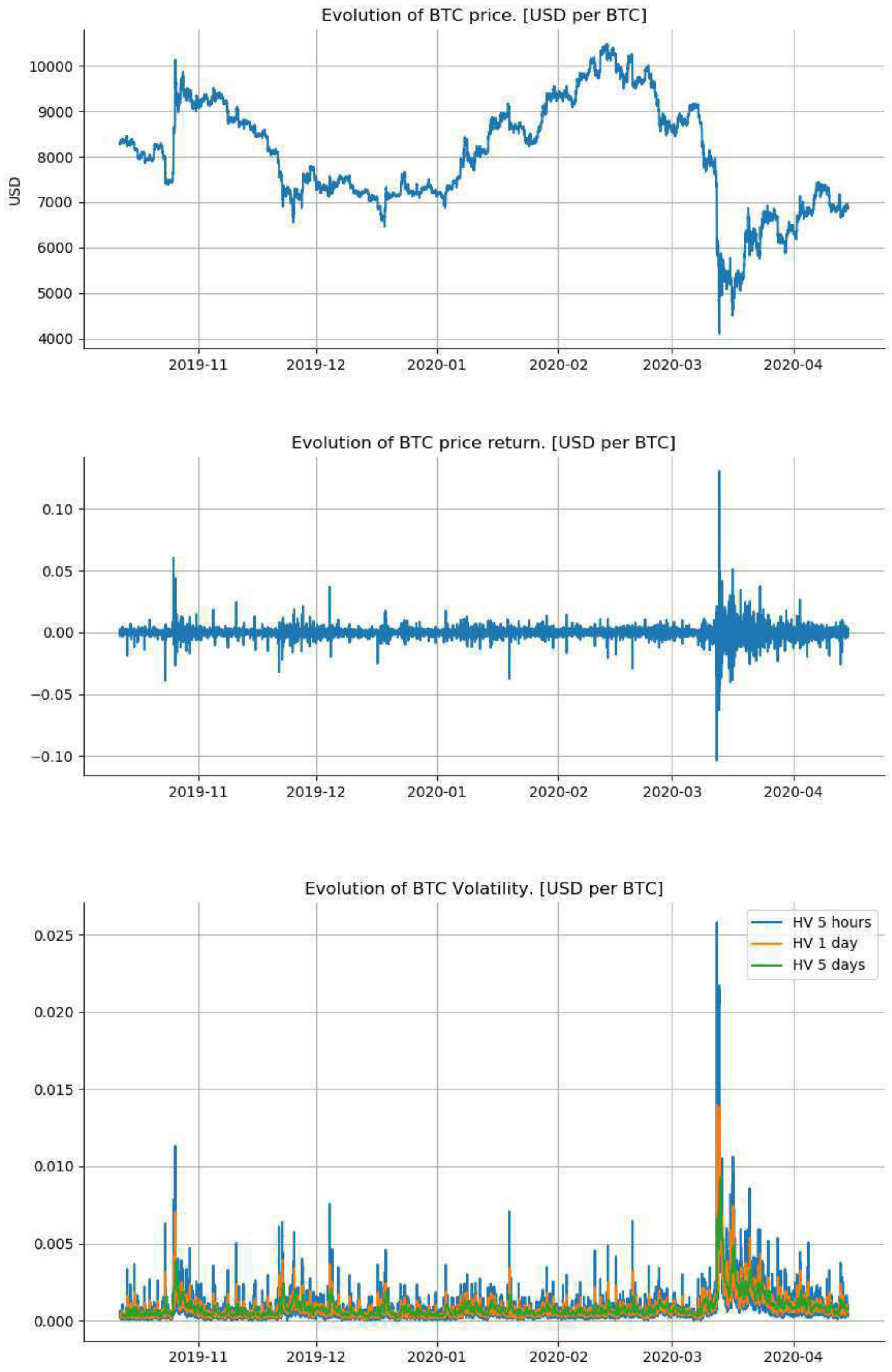


Figure 4.1: Time series of BTC-USD for 10-minute prices, returns and volatility with 5 hours, one day and 5 days.

Figure 4.2 gives the price, return and volatility of ETH-USD. The price, return, and volatility evolution in the given period shows a similar behavior as that of the BTC-USD pair. In November, the ETH price started off with a slight downward movement and then turned to an upward trend in January 2020. At the beginning of March, ETH plunged from more than $ 250 to less than $ 100 per ETH. Since the end of March, ETH has shown an upward trend.

As is the case for the BTC-USD pair, the returns for ETH show fewer outliers until March when the up and downs grew higher as of the middle of March. This movement can also be seen in the volatility chart. In the second and third week of March, the impact of volatility is greater than in the rest of the period.

The ETH-BTC price, return, and volatility are shown in Figure 4.3. The ETH-BTC pair shows a different behavior than that of BTC-USD and ETH-USD. The price for ETH in BTC had a downward trend from October until the middle of January 2020 when it bottomed out at 0.018 BTC. From then onward, the ETH price surged up to 0.028 BTC by the end of February. In March the price entered a downward phase, going to 0.021 BTC, and by the middle of April it was at 0.023 BTC.

The return figure reflects the picture with more up and downs during the whole period, but also clearly shows the higher movements at the middle of March. In relative terms, the volatility is less than in the other two pairs. However, the impact of the fast changes in volatility in March also shows up here in the clustering during this phase.

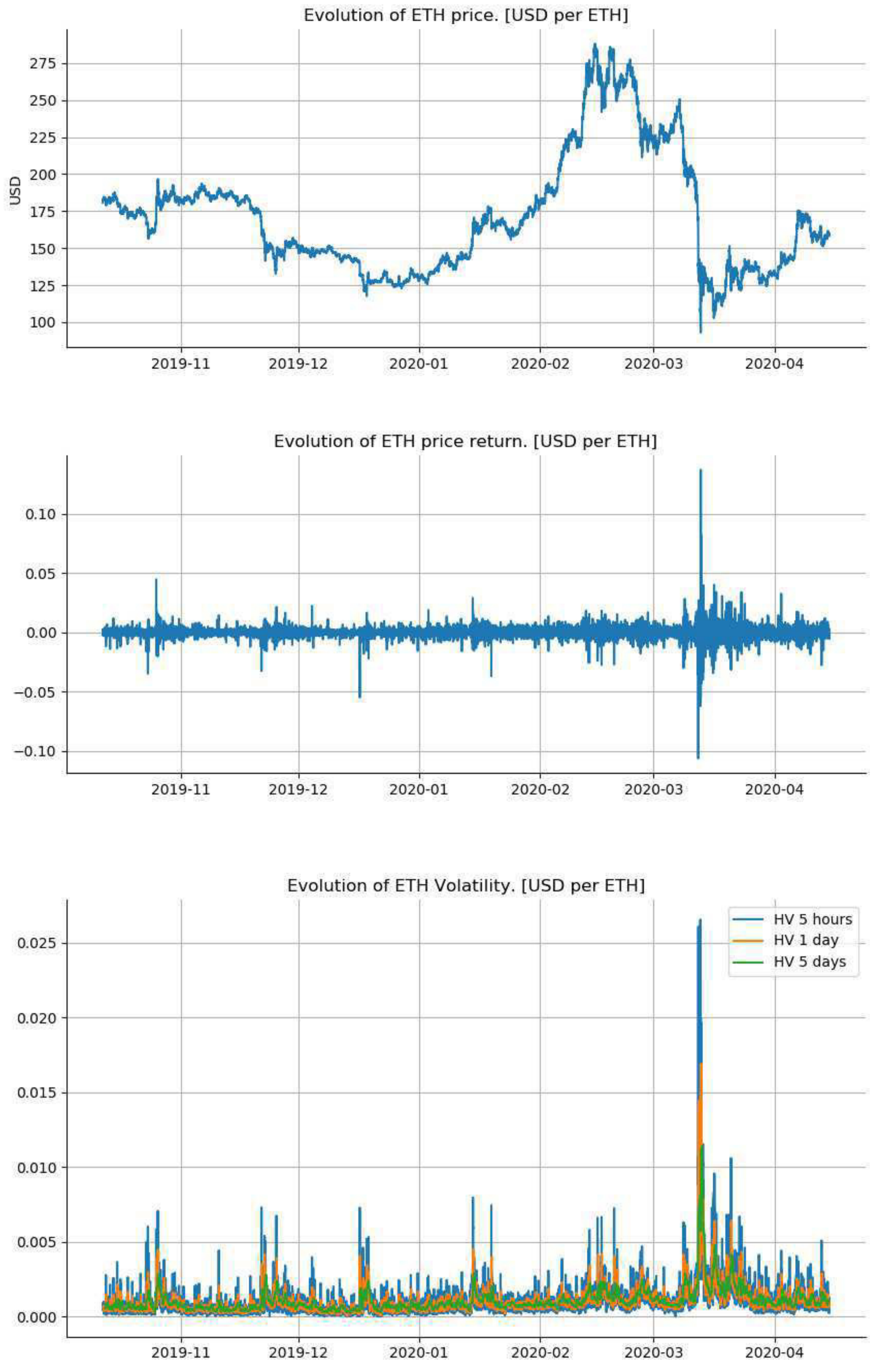


Figure 4.2: Time series of ETH-USD for 10-minute prices, returns and volatility with 5 hours, one day and 5 days.

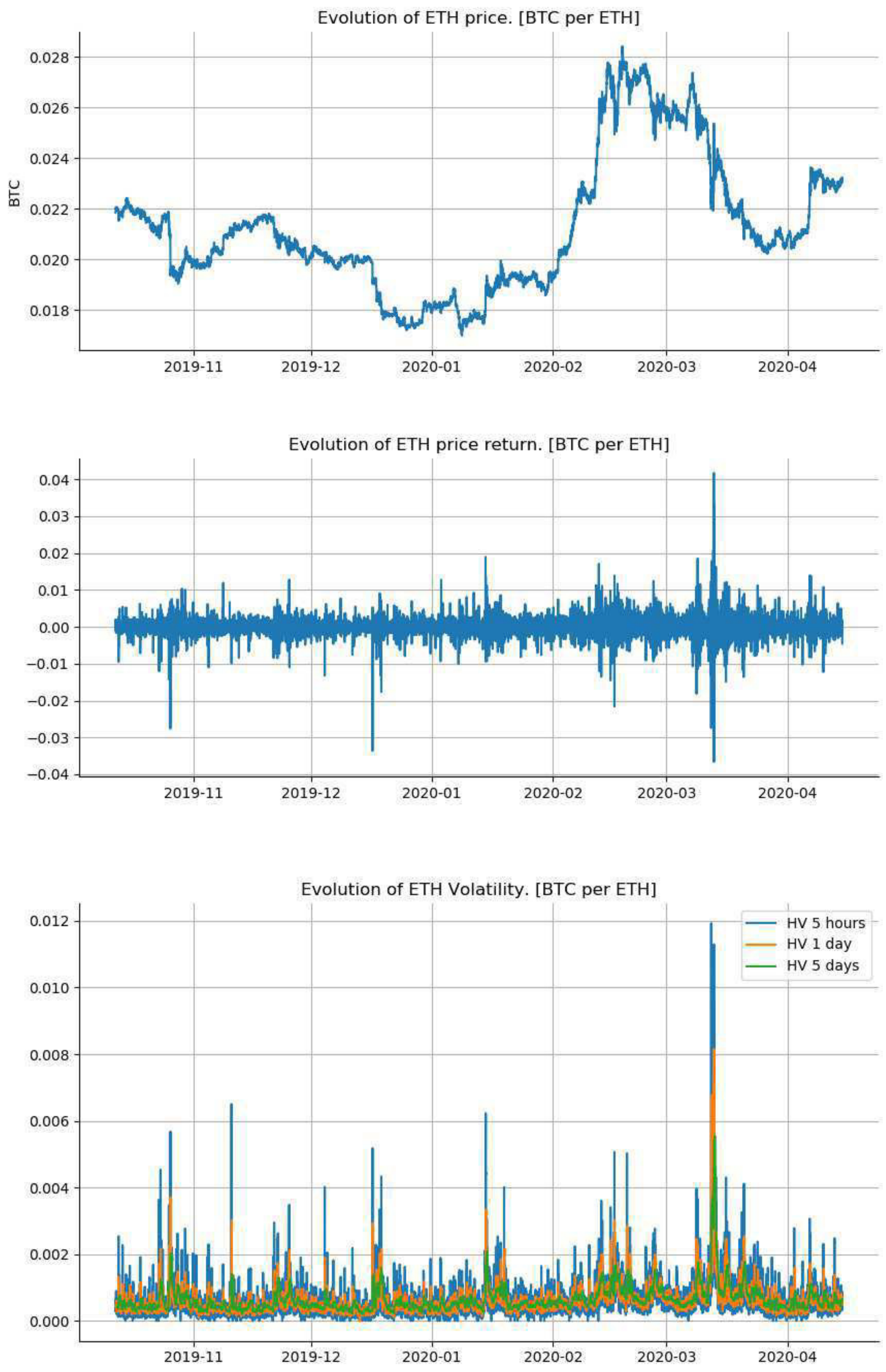


Figure 4.3: Time series of ETH-BTC for 10-minute prices, returns and volatility with 5 hours, one day and 5 days.

4.2 Data Split

The data for the GARCH and the ANN calculation starts on the 15^{th} of October 2019 at 00:00, and the last data point is on the 14^{th} of April 2020 at 23:50. This cut is because different input variables take a certain period to calculate the coefficient, such as the variable for the 72-hour volatility.

The data is split into a training data set and a test data set.

The training data set is from the 15^{th} of October 2019 at 00:00 to the 31^{st} of March 2020 at 23:59. With this data, the biases and weights of the neural network are trained, and the model parameters of the GARCH model are estimated. This process is also called in-sample set.

The test data set (holdoutset) is from the 31^{st} of March 2020 at 23:59:59.till the 14^{th} of April 2020 at 23:50.

This data set is to get an unbiased evaluation of the final model on the training data set. This is only used when the models are entirely trained and will be used to evaluate competing models. The testing with the test data set is also called out-of-sample testing.

5 Method

In this chapter, the methodology of the different models used in this thesis is presented. First, the benchmark for the volatility prediction of the next period is described with the GARCH model. Next, the ANN structure, input variables, and the different generalization methods are described. The last section explains the ES method, which is used to calculate the risk premium.

5.1 GARCH

With the GARCH model, it is possible to evaluate the volatility prediction using historical data. It takes the training data set to estimate the model parameters. With these parameters estimated, the GARCH model is used to make predictions for the test data sample.

The GARCH model functions as the benchmark for the ANN for the volatility prediction. The GARCH model is described as GARCH(p, q), where p is the lags of the conditional variance, and q is the lags of the squared error.

Brooks (2008, p. 392-399) concluded that the GARCH (1, 1) is sufficient to capture the volatility clustering in financial data. This is why in most academic economic literature, there is no higher-order model used. Therefore we chose the GARCH (1, 1) model.

As mentioned in section 3.4.6, the GARCH model will use the Student's t distribution of the error term.

The Student's t GARCH model was introduced by Bollerslev (1987) with the following form:

$$f(u_t/Y^p_{t-1})) = \frac{\Gamma\Big(\frac{1}{2}(\nu+1)\Big)}{\pi^{\frac{1}{2}}\Gamma(\frac{\nu}{2})}\Big((\nu-2)h_t^2\Big)^{-\frac{1}{2}}\Big(1+\frac{u_t^2}{(\nu-2)h_t^2}\Big)^{-\frac{1}{2}(\nu+1)} \tag{5.1}$$

The GARCH model chosen is a GARCH (1, 1) with a Student's t distribution, hold back with 10 data points and is updated every 5 data points. First, the different parameters of the model have to be estimated with the training data. The estimating process takes the log returns as the dependent variable, the constant mean as the mean model, the GARCH as the volatility model, the standardized Student's t as the distribution and the maximum likelihood as the estimation method. The frequency of iteration updates is set to 5.

The maximum likelihood method is trying to find the possible values of the parameters with the given data (Brooks, 2008, p. 395). The estimated parameters are μ, ω, α, β and ν. All mentioned parameters are explained in section 3.3.3, except the ν parameter. The ν parameter is part of the distribution error of the Student's t distribution and stands for the degrees of freedom.

In the GARCH model, the estimation of the parameters with the training data set is resampled in 10-minute time steps. The estimated parameters are used for the forecast of the period of the training data and for the test data, as mentioned in section 4.2. The forecast itself is one step ahead for the next ten minutes. The outcome there is the predicted variance. The variance is converted into the standard deviation.

The results are saved as a data frame for the later comparison for every 10 minutes as the GARCH variance and as the GARCH volatility prediction.

To compare the predictive power of the GARCH, we use the MSE approach. The MSE is also used for the ANN volatility prediction. The MSE is explained in section 5.2.2.

5.2 Artificial Neural Network

ANNs are often used in finance applications. We are focusing on the prediction of the volatility to compare the outcome from the ANN model with the result from the GARCH model. The model is given different input variables and uses these to predict the volatility for the next 10-minute period.

We are using Keras, which is an open-source Python library for developing and evaluating NN models. Keras is an API of Tensor Flow for deep learning models.

As described in section 4.2, the data set consists of a training set and a test set. The training set is used to find the optimal weights and biases. The test set is used to validate the ANN and compare it with the GARCH model.

5.2.1 Variables

The input variables for the ANN are calculated with the related market data. There were no classical technical indicators available for this short-term perspective. The input variables are used to train the ANN.

Input Variables:

- Log Return for 10 minutes and 2 hours
- Volatility for 10 minutes, 5 hours, 24 hours and 72 hours
- Trade Volume Total, Buy and Sell
- Number of Trades
- Log Return and Trade Volume from a Related Pair

Log Return:
The log returns are taken from the one-minute pre-processed OB data set. Then the log returns are rescaled by multiplying by 100 to get percentage values. Another reason for manipulating the data by multiplying it by 100 is that without the manipulation, during the training phase the training model returns a convergence warning because of the small return values in the data set.

The log returns on a one-minute basis are then resampled to obtain 10-minute log returns. As the input variable, the model is given the log return for 10 minutes of t_0 and the lagged returns of the last three periods for t_{-1}, t_{-2} and t_{-3}.

The log return of the last 2 hours is calculated as the sum of the previous 12 10-minute log returns. The 2-hour log return is provided to the model for t_0 as an input variable.

Volatility:
The volatility of the market is calculated for different periods. These periods are 10 minutes, 5 hours, 24 hours and 72 hours. The volatility is calculated with the pre-processed minute OB data. The 10-minute volatility is calculated as the standard deviation of 10 one-minute log returns. Then the data is resampled to 10-minute data, and the mean is taken. The 5-hour, 24-hour and 72-hour volatility is calculated in the same way, with the difference that each of them get the standard deviation and the resample process for the specific time frame.

The input variables given to the model are the 10-minute volatility of t_0 and the lagged 10-minute volatility of the last three periods for t_{-1}, t_{-2} and t_{-3}. What is more, the volatility of the last 5 hours, 24 hours and 72 hours for t_0 is also taken as an input variable.

Trade Volume Total, Buy and Sell:
The total buy and sell trading volume is calculated from the TR data.

The total trading volume for the last 10 minutes is the sum of the trade volume in the previous 10 minutes.

To generate the buy and sell trading volume, we first split the data into the buy and sell side. Then the sum of the trade volume for each side is calculated for 10-minute periods.

All three variables are normalized here and are input variables for t_0. Normalization is explained in section 5.2.1.1.

Number of Trades:
The number of trades is calculated from the pre-processed TR data. For this variable, the number of trades is taken. Therefore, we count the number of trades for every 10 minutes.

The outcome of this number of trades is then normalized. Normalization is explained in section 5.2.1.1.

The normalized total amount of trades for t_0 is added to the model as an input variable.

Log Return and Trade Volume from a Related Pair:
The log return and trade volume are calculated in the same way as mentioned above. The only difference here is that the data is taken from a related pair. We assume that potential synergies from a related market can have a positive impact on the training model and the outcome of an ANN.

The related pairs are:

- BTC-USD: related pair BTC-EUR
- ETH-USD: related pair ETH-EUR
- ETH-BTC: related pair BTC-USD

For these related pairs, the log return and the volatility for 10 minutes are calculated for t_0 and added as input variables.

Output Variable/Target Value:
The output variable is the 10-minute volatility, calculated with the one-minute OB data. The calculation schema is the same as the input volatility mentioned above.

However, to predict the future 10-minute volatility, the variable is shifted back one period.

Now, for each period we have the 10-minute volatility for t_1 and are in a position to predict the right outcome.

5.2.1.1 Normalization of Data:

We are using min-max scalar normalization. Here, the range is fixed between 0 and 1. This method changes the values of the numeric data to a scale without distorting differences in the range. This approach is highly sensitive to outliers.

The equation of the min-max scalar is:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{5.2}$$

5.2.2 Proposed Model

The ANN model used for volatility prediction uses all input variables as mentioned in section 5.2.1. The period for the training data for the model is defined in section 4.2.

The ANN for the prediction of the volatility is a back-propagation, feed-forward neural network with two hidden layers. In the training period, all input variables from the training data are presented as one data row at a time. Then the network tries to predict the next period's volatility. The outcome is then compared to the correct result, and the loss is calculated. With the calculated loss, the network transforms the biases and weights. After the adjustment of weights and biases, the process starts again and goes through the data.

In the following, the settings for the ANN used are defined, as we are using it for our final prediction for each pair. However, multiple experiments and tests were made to define these variables. In section 6.1, models are tested with different settings.

The model is given 18 variables as input. It consists of two hidden layers used in a feed-forward manner. Each layer has 100 neurons. The output has to return one value. The prediction of the volatility is a regression problem. The regression problem means a real-value quantity has to be predicted. This leads to the loss function of the

MSE and the ReLU activation function. The optimizer used is the Adam optimizer with a fixed learning rate of 0.001. To prevent overfitting, we use early stopping with the patience value of 3. The model is trained with a maximum of 150 epochs and a batch size of 32.

With the two hidden layers and 100 neurons in each layer, this model results in a total of 12,101 parameters, where all are trainable parameters.

Activation Function:
As an activation function, we chose ReLU for the hidden layers and the outcome layer. Relu is defined in more detail in section 3.6.3.

Optimizer:
We chose Adam as an optimizer for the ANN. Adam is a stochastic gradient descent method. This optimization approach is based on an adaptive estimation of first- and second-order movements of the gradients. The name Adam comes from adaptive moment estimation (Kingma & Ba, 2014). We chose a learning rate of 0.001 for our optimizer.

Loss Function:
As mentioned above, we are using the mean square error function as the loss function. This function is defined as:

$$MSE = \frac{1}{N-1}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{5.3}$$

Generalization Methods

Early Stopping:
Early stopping stops the training when the model stops improving. We have set the patience value to 3, which means that after three epochs with no improvement, the training ends.

Batch Size:
The batch size controls the accuracy of the error gradient during the training phase. The number of samples per gradient update is specified to be 32.

Predictive Power:
The predictive power of the model is measured with the R^2 KPI for the training and test sets for each pair. The R^2 score is denoted as the coefficient of determination. In general, the R^2 score tells us how much of the volatility can be explained by the model. The R^2 score is a relative number. One definition of the R^2 score is the correlation coefficient of the square between the observed outcome and the forecast value (Devore, 2011, p. 508-510).

We use the MSE approach to compare the prediction from the ANN with the GARCH prediction.

5.3 Comparison of the Prediction Models:

With the trained GARCH model from section 5.1 and the trained ANN model from section 5.2, we predict the volatility for the test period in 10-minute steps.

To choose the best possible outcome, we compare the prediction models with the MSE, defined in section 5.2.2. The MSE gives the average quadratic difference between the GARCH prediction and the ANN prediction.

5.4 Expected Shortfall

The ES approach is used to define the risk premium for the price quotation. In this thesis, we are focusing on calculating the risk premium for the investor's buying process. Therefore, we have to focus on the right side of the return distribution to calculate the premium. This is because, during a price quotation for buying, from the investor's perspective, an increasing price is harmful to the broker. Thus, the risk premium helps to prevent potential arbitrage trading by investors.

For the calculation of the ES, we use the Student's t linear VaR model. The ES equation is mentioned in section 3.4.6. The equation of the ES with Student's t distribution can also be written as:

$$ES_{h,\alpha,n}(X) = -\alpha^{-1}(1-n)^{-1}[n-2+x^2_{\alpha,n}]f_n(x_{\alpha,n})\sigma_h - \mu_h \tag{5.4}$$

where n is the degrees of freedom, α is the confidence level, h is the period length of 10 minutes, and X represents the 10-minute returns. x^2 is the percentage point of the Student's t density function with regard to the degrees of freedom and α. The f_n stands for the probability density function of the Student's t distribution. μ is the expected value, and σ is the standard deviation which we predicted with the GARCH and ANN models (Pawel Lachowicz, 2016).

The outcome of the ES is expressed in percentage of the pair value. For the ES calculation, we set the parameter μ to 0 because of the brief period of the risk premium calculation. We defined the degrees of freedom to be 5. The greater the degrees of freedom value becomes, the more similar the Student's t distribution becomes to the normal distribution. We define α as 1%. This means that the risk premium will cover the worst 1% of the density. Because μ is 0 and with the Student's t distribution we have a symmetrical distribution, it getting the risk premium for the worst 1% or the best 1% is the same (left and right side of the distribution).

To define the exact risk premium, we divide the potential ES value by 100. This gives us a proportionate premium for each price quotation.

5.4.1 Comparison of the Risk Premium

The outcome of the risk premium for the predicted volatility from the GARCH model and the ANN will be compared. The real price change and the prediction will be measured using the MSE. The calculation of the MSE is mentioned in 5.2.2.

6 Results

In this chapter, we will provide the results of the volatility prediction and the risk premium calculation. It starts with the experiments where we focus on the tests with the ANN. Then the outcomes of the GARCH and the ANN volatility prediction are presented and compared. At the end of this chapter, we calculate the risk premium for the test period using the predicted volatility and compare the results.

6.1 Experiments

The Experiments section mainly focuses on the different experiments we made with the ANN. Therefore, we show the outcomes of different parameter settings and also give a short explanation of why the basic parameter settings used were chosen. The basic settings were mentioned in section 5.2.2. If nothing different is mentioned in the Experiment section, the basic settings were used.

The experiments and tests for the ANN in this chapter were performed primarily with the BTC-USD pair. However, we validated the tests with the ETH-USD and ETH-BTC pairs. In the following experiments, we will report on the data of the BTC-USD pair, which is representative for all three pairs.

The comparison of the accuracy of each experiment is measured with the R^2 score. The R^2 is calculated for the training set and the test set. With the outcome of each score, we can define how well the model is trained and how well the trained parameters will fit for future predictions. The R^2 is defined in section 5.2.2.

All tests were made with at least three independent runs where we used the same parameters for the ANN three times in a row to get a bandwidth of different R^2 scores. Thus, the R^2 score will be written from the lowest value to the highest in the following comparison.

Layers:
We decided to use two hidden layers in our ANN. The reason was that in the testing phase, we saw that with only one hidden layer, the output showed a lag of predictive

power. With more than two layers, we could not find a significant improvement in the model compared to 2 layers. Each hidden layer has 100 neurons.

The outcome of the test runs related to the R^2 score is defined in Table 6.1.

Table 6.1: R^2 score with different layers

	Training Set	Test Set
Basic	0.759 - 0.787	0.503 - 0.521
Three hidden layers	0.723 - 0.735	0.506 - 0.511
Five hidden layers	0.672 - 0.726	0.474 - 0.49
One hidden layer	0.773 - 0.775	0.483 - 0.496

Neurons:
For the experiments, we tested using different numbers of neurons. In section 3.7, most of the studies used a minimal number of neurons per layer. The average number of neurons per layer was between 5 and 50. We chose the primary setting of 100 neurons for each hidden layer.

In Table 6.2 we compare the R^2 score of the model with different numbers of neurons.

Table 6.2: R^2 score with different neurons

	Training Set	Test Set
Basic	0.759 - 0.787	0.503 - 0.521
10 neurons per layer	0.798 - 0.808	0.464 - 0.495
50 neurons per layer	0.782 - 0.785	0.496 - 0.514
500 neurons per layer	0.744 - 0.755	0.478 - 0.50

For the test run with 500 neurons per layer and the basic settings, the early stop was triggered between 7 and 20 epochs.

Figure 6.1 shows the loss of the model with 500 neurons and the improvement after each epoch.

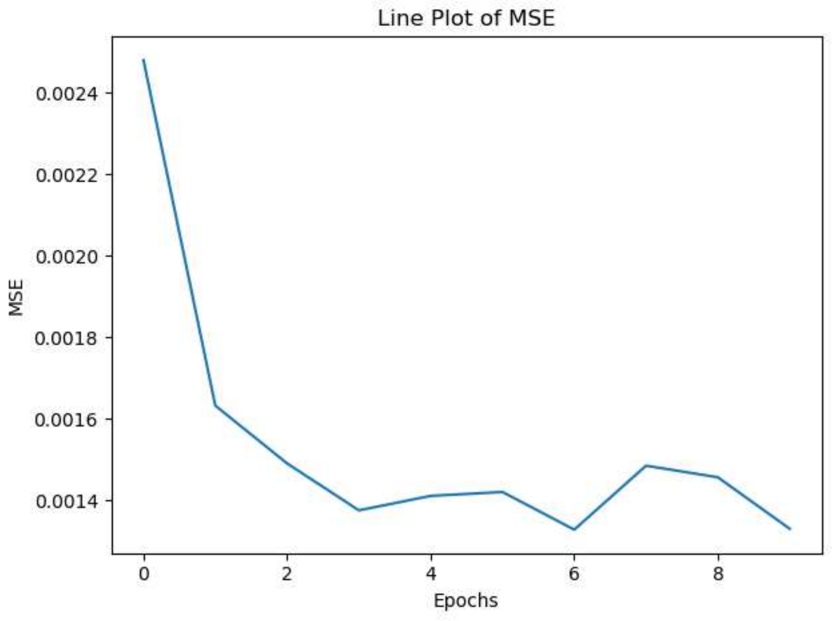


Figure 6.1: Line plot of MSE for each epoch with 500 neurons in each hidden layer for the training set.

Optimizer:
We chose Adam as an optimizer for stochastic optimization. We also did tests with the stochastic gradient descent (SGD) optimizer. Table 6.3 below shows the comparison of the basic settings of the Adam and the SGD optimizers.

The settings for Adam here are a learning rate of 0.001, a β_1 of 0.9 and β_2 of 0.

The standard settings for the SGD optimizer are a learning rate of 0.01 and a momentum of 0.

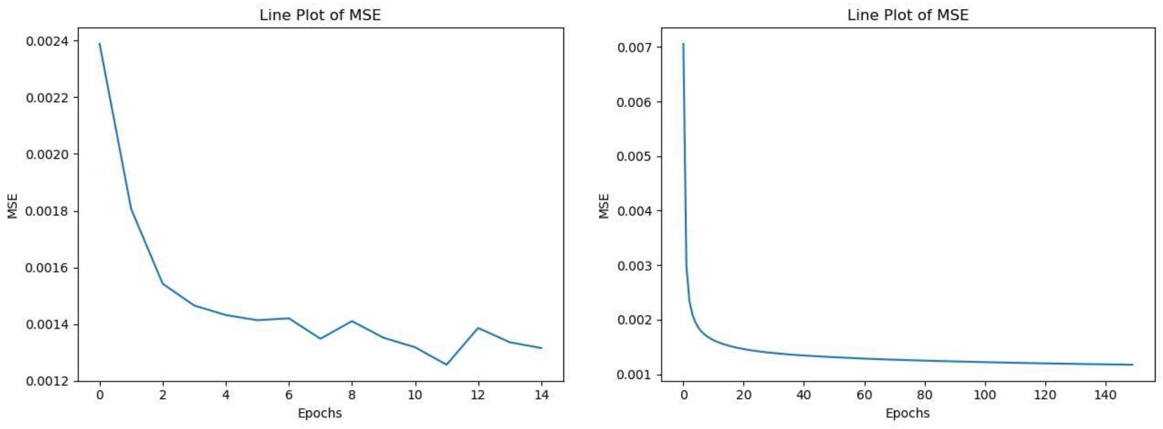
Table 6.3 shows that the SGD optimizer has a higher R^2 value for the training set than the Adam optimizer. What is more, it can clearly be observed that with the SGD optimizer the overfitting is greater (higher R^2 score in training and less in the test set) as compared to the Adam optimizer. The Adam optimizer provides a more a top model explanation of the volatility for the test data.

Table 6.3: R^2 score with different optimizer

	Training Set	Test Set
Basic (Adam)	0.759 - 0.787	0.503 - 0.521
SGD	0.853 - 0.855	0.481 - 0.495

Figure 6.2 shows the outcome of the MSE line plot. The basic optimizer settings only need 14 epochs to trigger the early stop with a patience of 3. The SGD requires all 150 epochs and does not trigger the early stop. This is due to the different learning rate between the optimizers.

Generally speaking, the SGD needs more time to reach the (local) optimum than the Adam optimizer.



(a) Adam Optimizer (b) SGD Optimizer

Figure 6.2: Line plot of MSE for Adam and SGD optimizers.

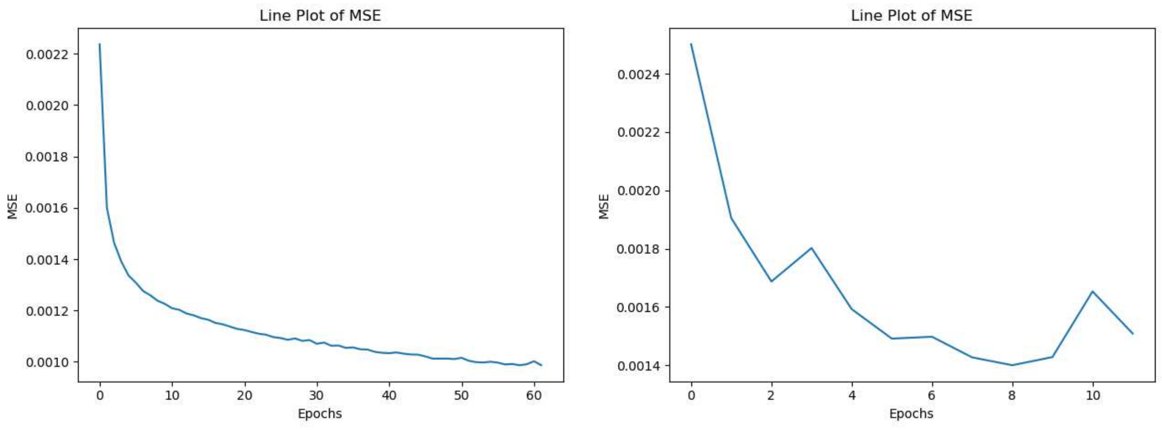
Learning Rate:
We tried different learning rate values to increase the predictive power of our model. If we take the learning rate too high, we will always jump from one local minimum to another and will not come close to a minimum there. If the learning rate is too low, we will be stuck in a single local minimum.

Table 6.4 shows the comparison of the R^2 score with different learning rates.

Table 6.4: R^2 score with different learning rates

	Training Set	Test Set
Basic learning rate 0.001	0.759 - 0.787	0.503 - 0.521
Learning rate 0.0001	0.869 - 0.874	0.504 - 0.519
Learning rate 0.0005	0.785 - 0.794	0.497 - 0.511
Learning rate 0.005	0.711 - 0.743	0.409 - 0.513

When we use a lower learning rate, it takes more epochs until the early stop is triggered. This makes sense because with a lower learning rate, we also make smaller steps, and it is more unlikely to have the same loss three times in a row. We can see the different number of epochs in Figure 6.3



(a) MSE with learning rate of 0.0001 (b) MSE with learning rate of 0.005

Figure 6.3: Line plot of MSE for learning rates of 0.0001 and 0.005.

Epochs:
We chose the value of 150 epochs for our ANN. During testing, we saw that most of the time, the early stop was triggered and the learning process was stopped before the total number of 150 epochs was reached. When we used a lower learning rate, the test often ran for all 150 epochs. The MSE curve always became flattened very early relative to the improvement process.

Loss Function:
For the loss function, we chose the MSE to calculate the loss. We tested different loss functions, such as the mean absolute error and the mean squared logarithmic error. The R^2 scores for the two other methods were slightly higher for the training and the test data than with the MSE. In the related work, most of the networks use the MSE as the loss function. Therefore, we decided to stay with this method.

Activation Function:
We compared the ReLU and the softplus activation functions. The results with the softplus function, when used in all layers (hidden and output layers) and when used only in the output layer, were very similar to those of the ReLU function. We decided to use ReLU as the activation function for the hidden layers and the output layer.

6.1.1 Overfitting:

In the different tables above we saw that the differences in the R^2 score between the training and the test data are high. This is an indicator that the model might overfit with the training data and therefore have difficulty predicting the test data. To reduce this phenomenon, we tried early stop, weight constraints and dropout to reduce this effect.

Early Stop:
The early stop ends the training of the model when the monitored quantity has stopped improving. This quantity is called patience and is set to 3. This means that when the loss of the model has the same value three times, the model stops.

We did tests with higher and lower values for patience. We can see the results in Table 6.5.

Table 6.5: R^2 score with different patience

	Training Set	Test Set	Epochs
Basic patience 3	0.759 - 0.787	0.503 - 0.521	22 - 33
Patience 1	0.755 - 0.773	0.481 - 0.495	6 - 8
Patience 10	0.878 - 0.857	0.527 - 0.538	58 - 76
No early stop	0.892 - 0.897	0.44 - 0.51	150

We can see that using the early stop, a model with less overfitting with the basic settings is achieved when the differences between R^2 of the training and the test set are compared. What is more, with higher patience, a higher R^2 for the test set is achieved, but the difference between the training set and the test set also increases. Therefore, we decided to use a patience value of 3 for the early stop.

Weight Constraints:
We tested weight constraints in the kernel constraints for the weights of the hidden layers. We used unit norm as the constraint. Unit norm forces the weights to normalize. The norm of unit norm is equal to 1.

In Table 6.6 we compared the R^2 score with the weight constraint and without the weight constraint.

Table 6.6: R^2 score with and without weight constraints

	Training Set	Test Set
Basic, no weight constraint	0.759 - 0.787	0.503 - 0.521
Unit norm	0.777 - 0.808	0.501 - 0.512

The weight constraint shows no improvement with the unit norm approach. Therefore, we did not use the weight constraint for our model.

Dropout:
Dropout is a method which randomly sets a fraction rate of input units to 0 every update. This input elimination happens during the training and is intended to prevent overfitting.

In Table 6.7, we can see the different fraction rates and their R^2 score. We used the dropout approach in the two hidden layers.

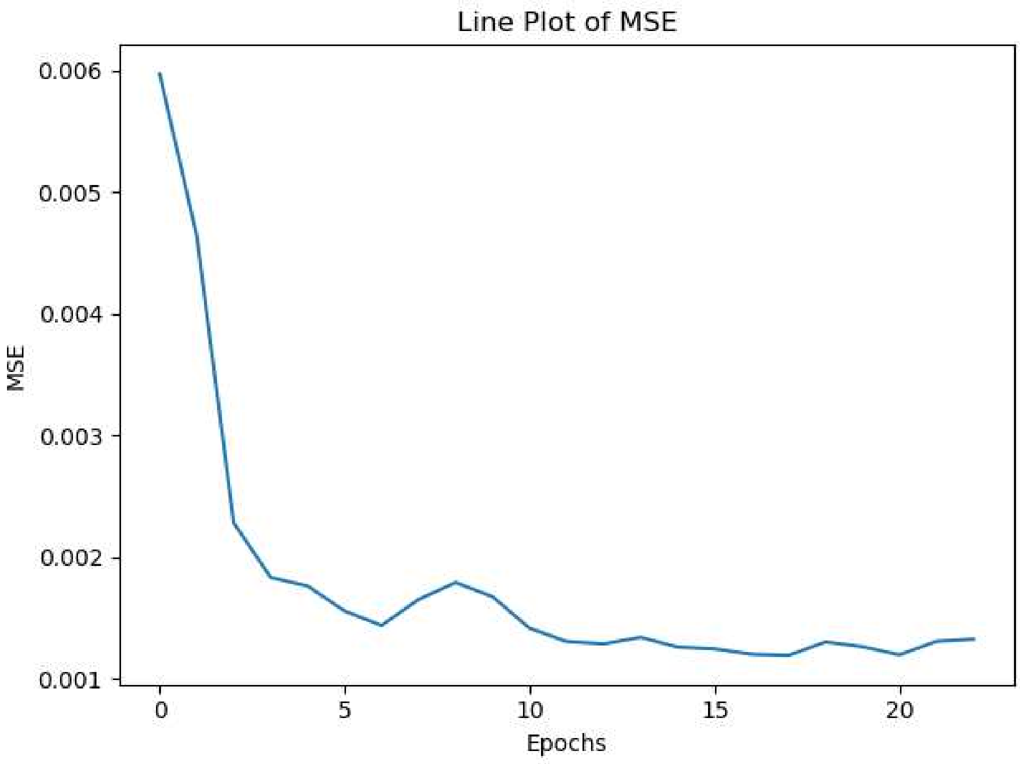
Table 6.7: R^2 score with different fraction rates of dropout

	Training Set	Test Set
Basic, no dropout	0.759 - 0.787	0.503 - 0.521
Fraction rate 0.2	0.672 - 0.674	0.478 - 0.484
Fraction rate 0.5	0.446 - 0.456	0.33 - 0.374
Fraction rate 0.05	0.739 - 0.777	0.51 - 0.52

We can see that a very large fraction rate drastically corrupts the model. A small fraction rate of 0.05 shows a better outcome for the model. However, the difference between no dropout and the fraction rate of 0.05 is very small. Therefore, we decided to not include dropout in our model.

Hybrid Model:
As input, hybrid models are given the GARCH prediction compared with other input variables. In section 3.7, we have mentioned papers which focused on hybrid ANN models. With the GARCH as an input variable, the model will easily outperform the vanilla GARCH model, because it has its prediction outcome as a variable input. In this thesis, we are comparing the GARCH and the ANN models. Therefore, we decided not to take a hybrid model as the primary model. However, we tested the hybrid model during the testing phase for the model parameters. In Table 6.8 and in Figure 6.4 we can see the outcome for the BTC-USD pair.

The basic settings stay the same, except that the GARCH prediction of the 10 minutes is given the ANN as an additional input variable.



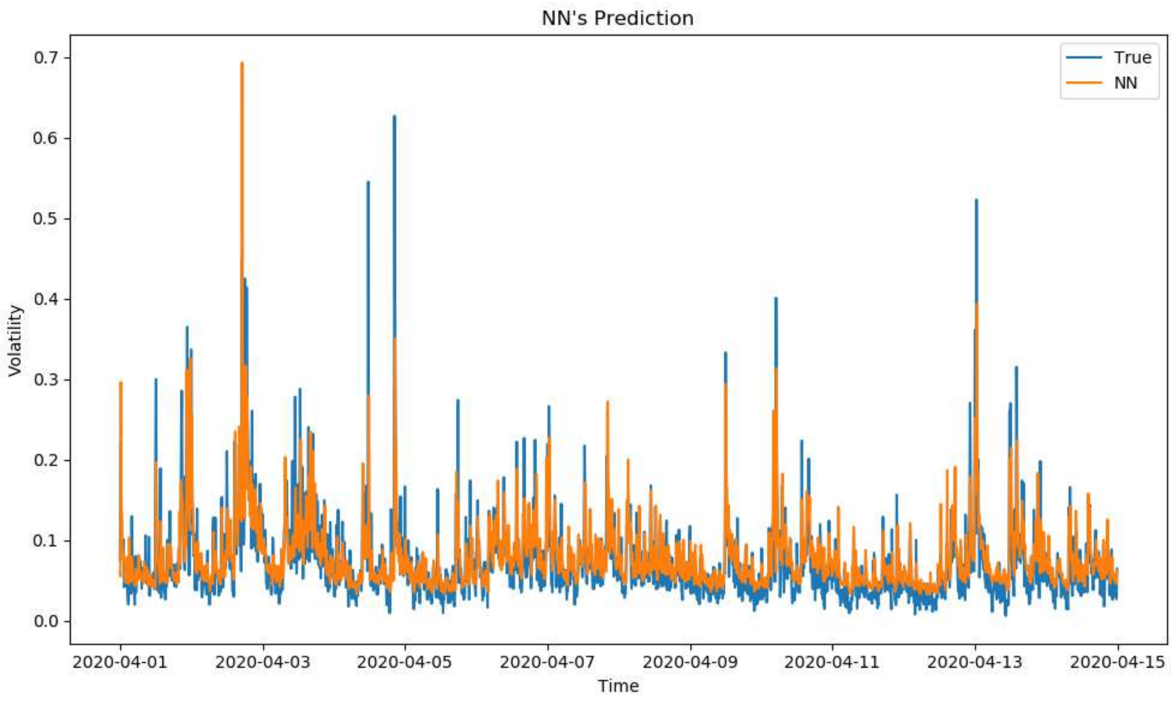


Figure 6.4: Hybrid model of the ANN. The upper chart shows the line plot of the MSE learning rate. The lower chart shows the predicted and the real volatility compared over the testing period.

Table 6.8: R^2 score of basic model and hybrid model

	Training Set	Test Set
Basic	0.759 - 0.787	0.503 - 0.521
GARCH prediction as input	0.761 - 0.804	0.515 - 0.525

In Table 6.8, we can see that the hybrid model gets a slightly better R^2 score performance as compared to the basic model. In Figure 6.4, we can see the MSE learning rate and the comparison of the predicted and the real volatility.

Different Period:
Because of a sharply increased volatility in the middle of March, most likely due to the corona crisis, we tried to train the model with data from the 1st of November 2019 at 00:00 to the last data point on the 14th of April 2020 at 23:50. The test data were take from the past from the 15th of October 2019 at 00:00 to the 31st of October 2019 at 23:50. All settings for the ANN are the basic settings.

Here we tried to test if it is easier for the model to predict past volatility using data from the future.

We can see the results in Table 6.9.

Table 6.9: R^2 score of basic model with different test data

	Training Set	Test Set
Basic	0.738 - 0.784	0.669 - 0.709

The results of the prediction look very promising. However, the testing period was a period of very low volatility, and the market was experiencing sideways movement.

6.2 Outcome

In this section, we will show the results of the basic model for each of the pairs, the benchmark outcome and the comparison of the models. Then the expected shortfall is calculated, and the risk premium is defined.

6.2.1 GARCH prediction

The GARCH prediction is made as described in section 5.1. Table 6.10 shows the prediction of the different parameters of the model. The GARCH model is calculated on the basis of the Student's t distribution for the pairs BTC-USD, ETH-USD and ETH-BTC.

Table 6.10: The parameters of the GARCH volatility models

	BTC-USD	ETH-USD	ETH-BTC
Ω	0.0021	0.0028	0.0007
(s.e.)	0.0003	0.0016	0.0001
(p-value)	0.0000	0.0875	0.0000
α_1	0.2000	0.1972	0.1792
(s.e.)	0.0144	0.0471	0.0176
(p-value)	0.000	0.0000	0.0000
β_1	0.7800	0.7831	0.8208
(s.e.)	0.0189	0.0594	0.0175
(p-value)	0.0000	0.0000	0.0000
ν	4.0959	4.4627	3.6840
(s.e.)	0.2370	0.241	0.0866
(p-value)	0.0000	0.0000	0.0000

We can see in Table 6.10 that the estimated Ω of the pair ETH-USD is not significant on the basis of the 5% significance level. For all other values of the GARCH (1, 1) model, the estimated parameters are significant.

Table 6.11 shows the result of the MSE comparison. The real 10-minute volatility is compared with the MSE of the predicted volatility from the GARCH model.

Table 6.11: The MSE results of GARCH volatility models

	BTC-USD	ETH-USD	ETH-BTC
MSE GARCH (1, 1)	0.03668	0.05707	0.01650

6.2.2 ANN Prediction

The prediction of the volatility was made using the basic settings, as mentioned in section 5.2. The forecast was made for the pairs BTC-USD, ETH-USD and ETH-BTC. In the following, we will go into more depth on each pair of prediction parameters. Then the MSE of each pair is calculated and shown in Table 6.13.

The ANN for each pair has as an output shape of 100 neurons for hidden layers 1 and 2, and the output layer has one value. This value is the predicted volatility for each pair. The total number of parameters for each pair is 12,101 and all of them are trainable parameters.

BTC-USD ANN:
The R^2 score for the ANN model of the BTC-USD pair is shown in Table 6.12. The MSE learning rate for the epochs and the comparison of the real volatility with the predicted volatility for the test period can be found in Figure 6.5. Here, the ANN model took 20 epochs before the early stop was triggered and the model stopped training.

Table 6.12: The R^2 score of BTC-USD ANN volatility model

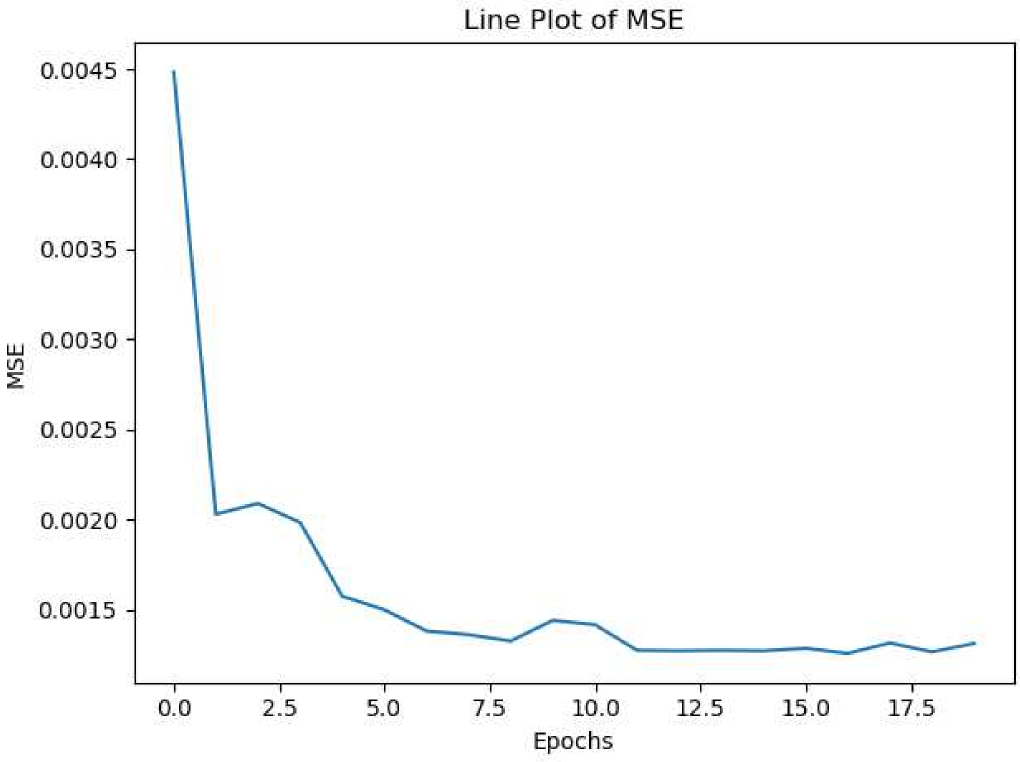
	R^2
Training Set	0.764
Test Set	0.521

With regard to the R^2 score, Table 6.12 shows that there is a discrepancy between the training data and the test data. Figure 6.5 shows the line plot of the MSE. Here we can see that the improvement of the model after 6 epochs flattens drastically. In the figure NN's Prediction we can see that the model increases the prediction value when the volatility increases. However, it has difficulty predicting the exact outcome of highly volatile results.

The MSE of the BTC-USD pair is shown in Table 6.13

Table 6.13: MSE of BTC-USD

	BTC-USD
MSE ANN basic	0.00046



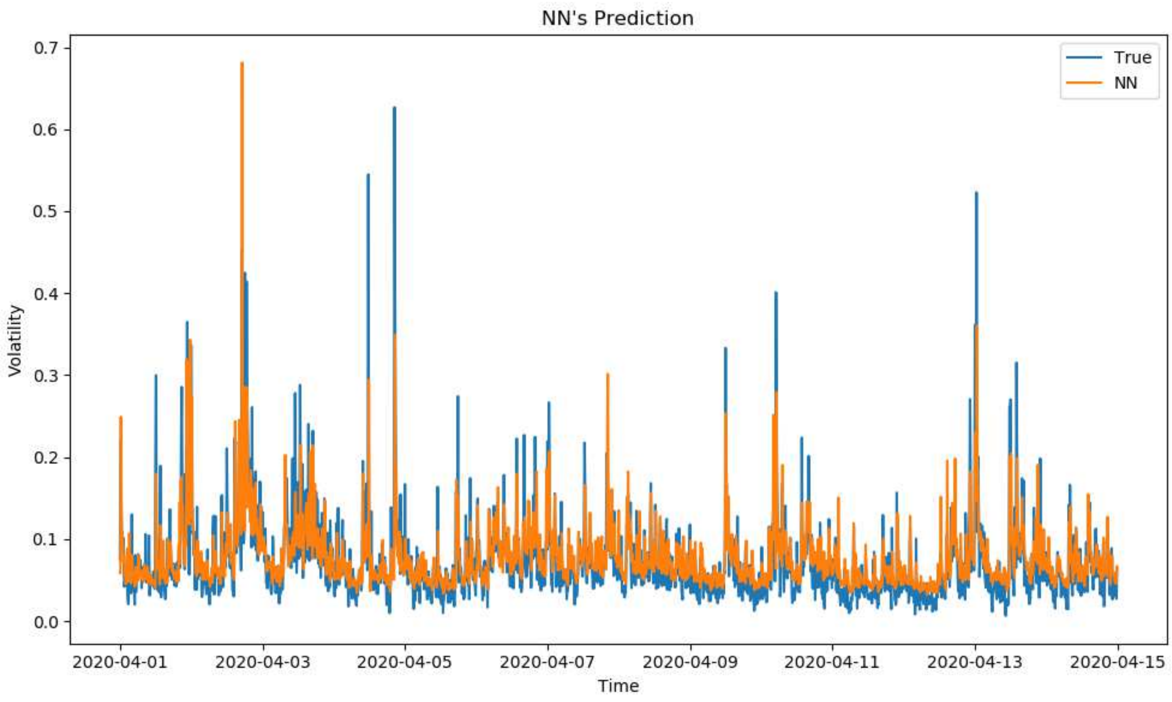


Figure 6.5: ANN model of the BTC-USD pair. The upper chart shows the line plot of the MSE learning rate. The lower chart shows the predicted and the real volatility compared over the testing period.

ETH-USD ANN:
For the pair ETH-USD, the R^2 score of the ANN model is shown in Table 6.14. In Figure 6.6, we can see the MSE learning rate and the NN prediction compared with the real volatility. The early stop of the model was triggered after 12 epochs.

Table 6.14: The R^2 score of ETH-USD ANN volatility model

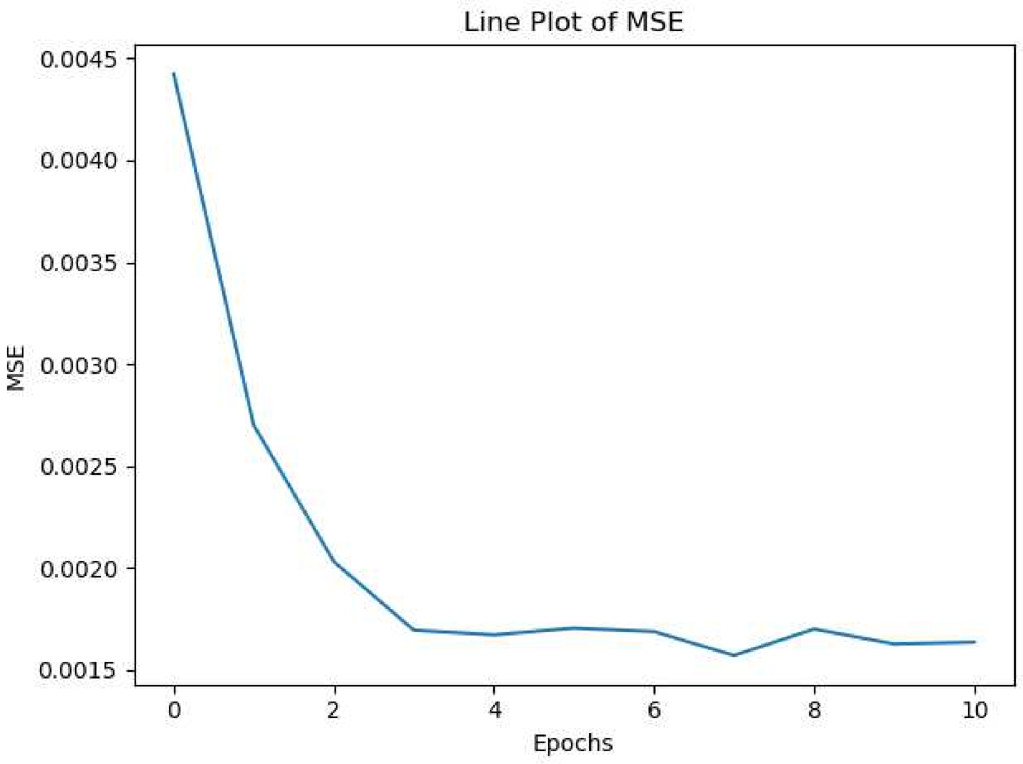
	R^2
Training Set	0.819
Test Set	0.572

Also for the pair ETH-USD, we can see a gap in the R^2 score between the training and the test data in Table 6.14. In Figure 6.6, we can see the line plot of the MSE begins to flatten after epoch 4. After this, no significant decrease is recognizable for the MSE of the model. In the figure showing the NN's prediction, we see that the model has difficulty predicting the exact value of the volatility.

The MSE of the ETH-USD pair is shown in Table 6.15

Table 6.15: MSE of ETH-USD

	ETH-USD
MSE ANN basic	0.00125



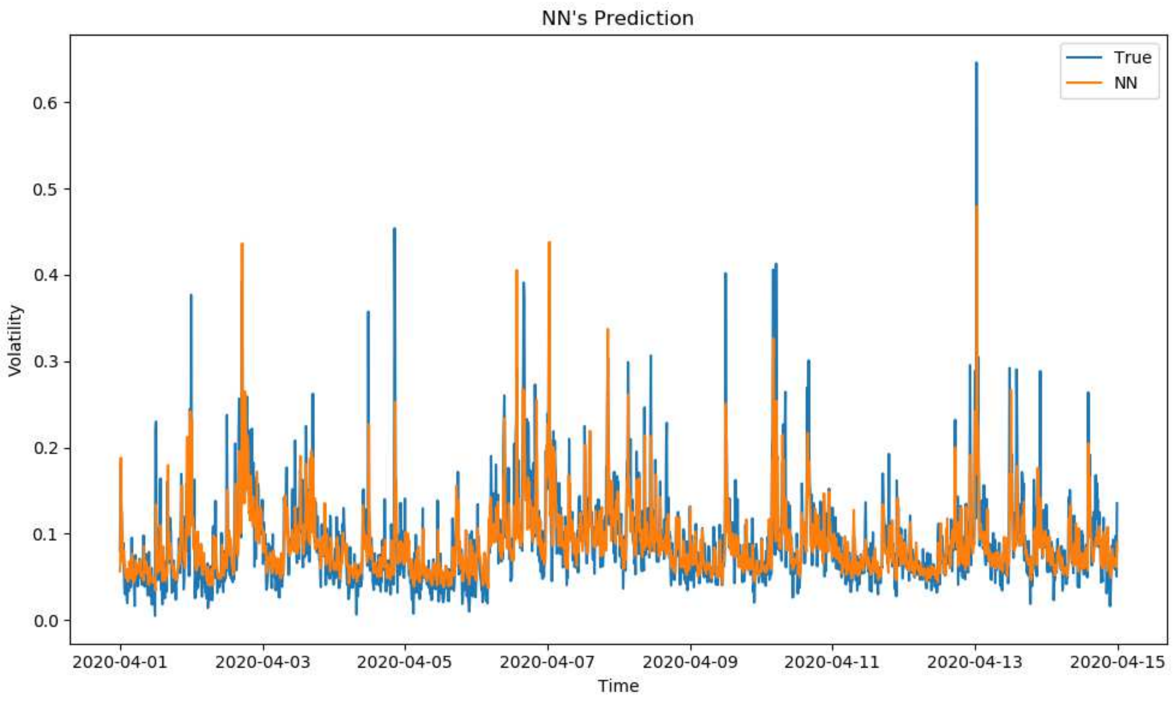


Figure 6.6: ANN model of the ETH-USD pair. The upper chart shows the line plot of the MSE learning rate. The lower chart shows the predicted and the real volatility compared over the testing period.

ETH-BTC ANN:
The R^2 score of the pair ETH-BTC for the ANN model is in Table 6.16. In Figure 6.7, we can see the MSE learning rate and the NN prediction compared with the real volatility. The early stop of the model was triggered after 12 epochs.

Table 6.16: The R^2 score of ETH-BTC ANN volatility model

	R^2
Training Set	0.713
Test Set	0.535

The R^2 score of the pair ETH-BTC is in Table 6.16. For this pair as well, we can see a difference in the R^2 score between the training and the test data. Figure 6.7 gives shows that the MSE stoped decreasing after epoch 8.

The MSE of the ETH-BTC pair is shown in Table 6.17

Table 6.17: MSE of ETH-BTC

	ETH-BTC
MSE ANN basic	0.00059

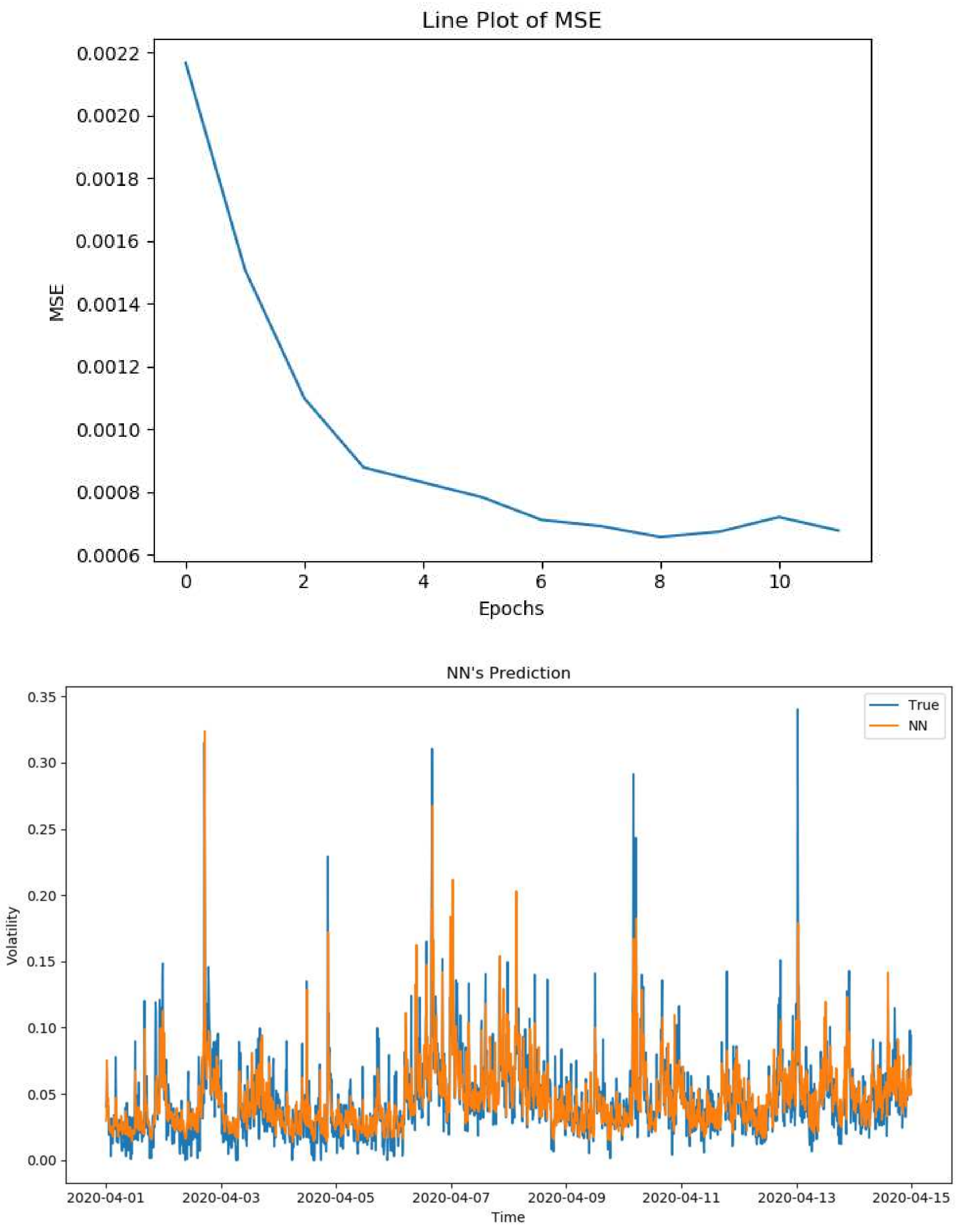


Figure 6.7: ANN model of the ETH-BTC pair. The upper chart shows the line plot of the MSE learning rate. The lower chart shows the predicted and the real volatility compared over the testing period.

6.2.3 Comparison of the Prediction Models

In this section, we will compare the MSE of the GARCH and the ANN prediction for the pairs BTC-USD, ETH-USD and ETH-BTC.

Table 6.18: MSE comparison of the different volatility prediction models

	BTC-USD	ETH-USD	ETH-BTC
MSE volatility ANN basic	0.00046	0.00125	0.00059
MSE volatility GARCH (1, 1)	0.03668	0.05707	0.01650

In Table 6.18, we can see that the ANN prediction has a lower MSE for each pair. The table shows that the volatility prediction using the ANN outperforms the GARCH model within the training period.

6.2.4 Risk Premium

In this section we will calculate the risk premium for the testing period. We define the quotation price as the mid-price at time t_0 plus the risk premium. The quotation price is then compared to the mid-price at t_1. Then we use the MSE to measure the difference between the quotation price and the mid-price at t_1. The goal here is to stay positive but as close as possible to zero.

Table 6.19 shows the MSE of the estimated risk premium compared with the mid-price of the next period. The number of times the risk premium was too small so that the mid-price of the next period exceeded the quotation price are measured. The risk premium compared with the MSE is lower with the ANN approach than the GARCH model. The number of times the mid-price exceeded the risk premium is much higher with the ANN volatility prediction for all pairs.

Table 6.19: MSE of the risk premium and the number of times the mid-price of the next period exceeded the risk premium

	BTC-USD	ETH-USD	ETH-BTC
MSE risk premium ANN basic	515.546	0.663	3.972 e-09
MSE risk premium GARCH (1, 1)	5389.052	4.331	2.403 e-08
ANN quantity $riskpremium < price_{t1}$	402	197	155
GARCH quantity $riskpremium < price_{t1}$	6	7	7

For the pair BTC-USD, we can see that the GARCH approach for the risk premium calculation during the test period was only too low six times, whereas with the ANN approach, it was too low more than 402 times. A similar outcome applies for the ETH-USD and ETH-BTC pairs, as we can see in Table 6.19. On the other hand, the MSE from the ANN prediction method outperformed that of the GARCH model very clearly for all three pairs.

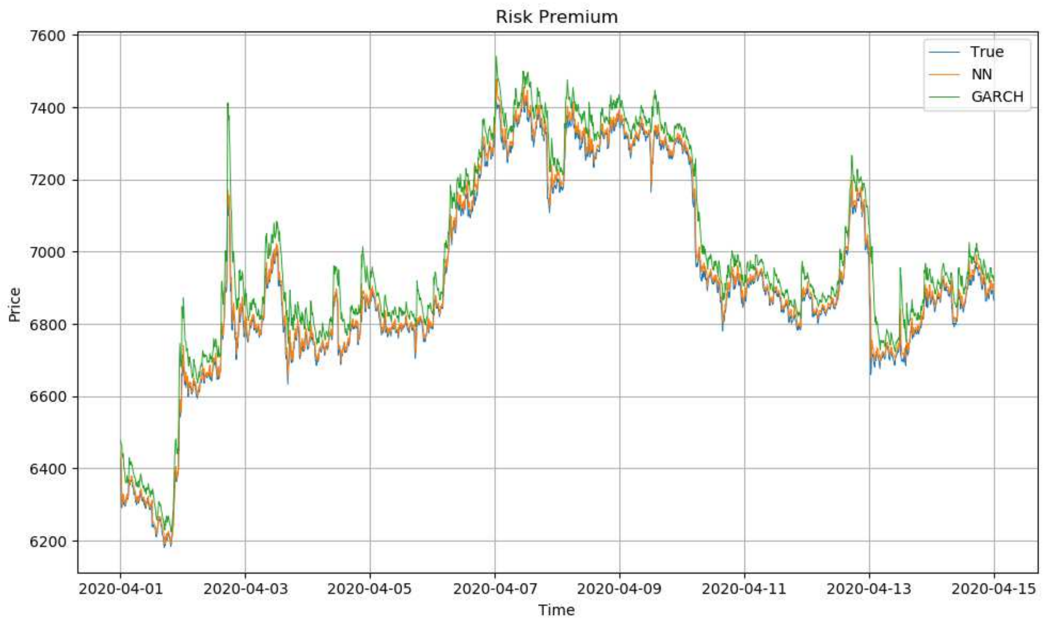


Figure 6.8: Comparison of the real mid-price and the risk premium with the calculated volatility of the ANN and GARCH models for BTC-USD.

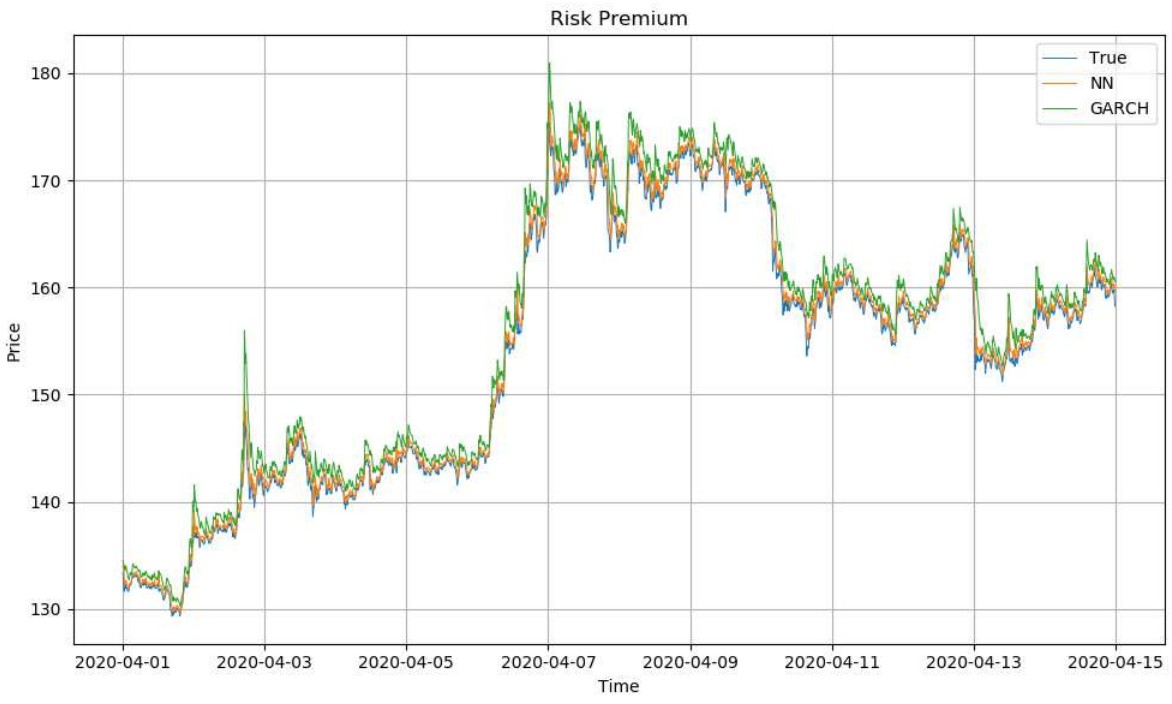


Figure 6.9: Comparison of the real mid-price and the risk premium with the calculated volatility of the ANN and GARCH models for ETH-USD.

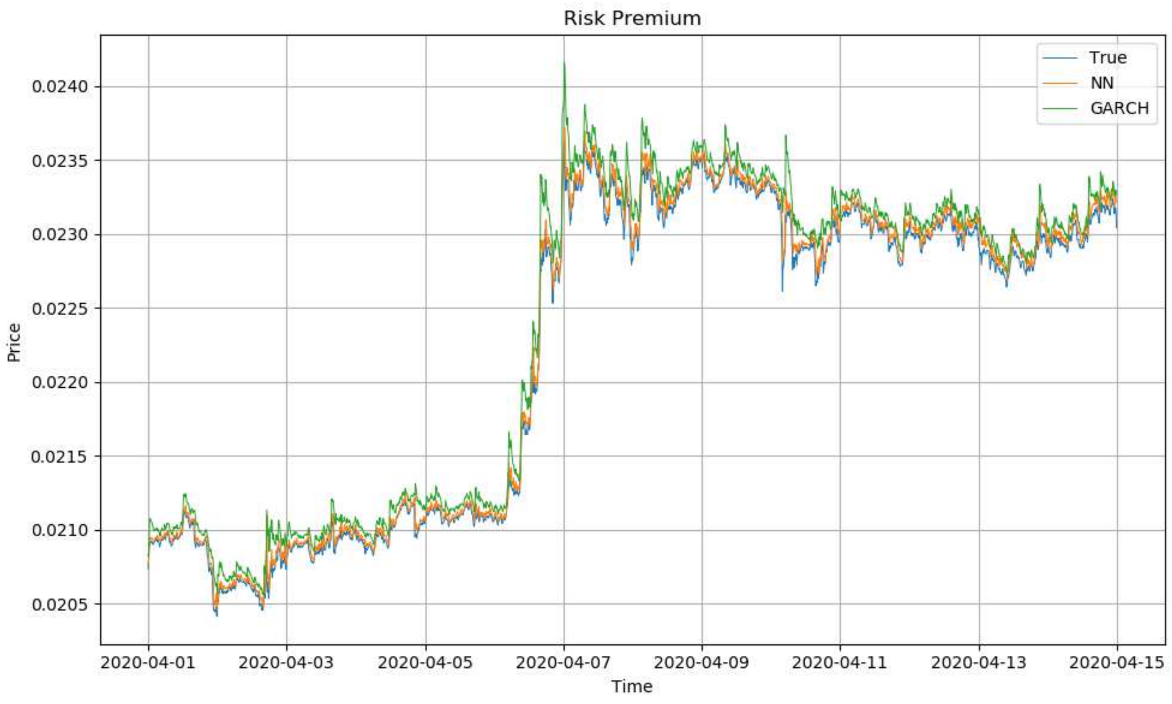


Figure 6.10: Comparison of the real mid-price and the risk premium with the calculated volatility of the ANN and GARCH models for ETH-BTC.

Figures 6.8, 6.9 and 6.10 show the comparison of the real mid-price and the quotation price with the calculated volatility of the ANN and the GARCH models. For all three pairs, we can ascertain that the quoted price using the volatility calculation of the ANN is closer to the real mid-price as compared to the volatility calculated using GARCH. The GARCH volatility, in general, tends to be higher than the real mid-price.

7 Discussion

In this thesis, we have focused on the calculation of a risk premium for a short-term period in the cryptocurrency market. With this risk premium, brokers can quote a price for potential investors and give them a period of time to decide whether or not to invest in this cryptocurrency. In this chapter, we will start with an interpretation of section 6. Then the implications of the results and the limitations will be discussed. At the end of this chapter, we will recommend potential studies for further research.

The results of this study indicate that the ANN approach provides a better volatility forecast for short-term cryptocurrency markets as compared to the GARCH models. With the predicted volatility, we calculated the risk premium and also found that the ANN approach shows better results for the pairs BTC-USD, ETH-USD and ETH-BTC.

7.1 Interpretation of Results

The exact results can be found in section 6. We found that the volatility prediction for short-term periods is a more complex field than a long-term period, e.g. using a daily basis. The research question was whether a NN approach can achieve a more precise risk premium than a standard GARCH approach. We expected that the ANN model could produce a more accurate outcome, and this was also shown in all our testing pairs. Thus, the results tend to support our hypothesis.

Data Quality:
The deep learning method in this thesis needs data for estimating the parameters. We used data from one exchange for a total period of 6 months. With more data, we could probably counteract overfitting. The cryptocurrency market itself is very dynamic and very volatile. Thus, a larger time frame of data from different exchanges could lead to a better working model.

GARCH Model for Volatility Prediction:
The results of the GARCH model for the volatility prediction were compared with the MSE approach. The MSE cannot be interpreted on its own. The analysis will be done using a comparison with the MSE of the ANN model.

ANN for Volatility Prediction:
By using deep learning algorithms such as ANN, we expected a more precise volatility prediction than that of a standard GARCH model. It was very challenging for us to identify which variables should be used as input for the ANN and to select the parameter settings for the ANN. Selection of the input and the parameter settings could go on forever because the possibilities are infinite.

We compared the results of the ANN and the GARCH volatility predictions using the MSE. The outcome of the test data was more critical for us than the training data. We determined that all volatility predictions using the ANN have a gap in the R^2 score between the training data and the test data. This gap is between 17.8 and 24.7 percent.

We came up with two different explanations as to why the R^2 score has this gap. The first is potential overfitting of the model. The second is that the volatility is primarily driven randomly and cannot be derived from past data.

To examine the first explanation, we tried to counteract the overfitting using different methods. However, the methods did not decrease the gap as expected. This is why we suspect that the second explanation is more applicable.

What also speaks for the coincidence explanation for the volatility in the short-term period is that we saw from the data that the volatility clustering, as mentioned from Mandelbrot (1963), cannot be seen in the same way as when using daily periods. We identified volatility clusters, but even in these clusters, there were also clusters of small volatility. For the short-term period, we saw that there were often volatility outliers without a cluster or volatility clusters with low volatility phases. This is a new perspective on the volatility cluster approach, which was surprising.

Comparison of the ANN and the GARCH approaches:
To compare the ANN and the GARCH approaches, we calculated the MSE for each pair and each model. We found that the ANN volatility forecast outperformed the GARCH approach for every pair. We do not find this outcome surprising because the ANN model has the same input variables as the GARCH model, but also includes additional variables. More variables should also provide a more precise outcome.

Risk Premium Calculation:
The risk premium calculation was made using the volatility prediction from both models. The outcome for the MSE risk premium showing that the ANN approach outperforms the GARCH method is not surprising. This is because we have already seen in section 6.2.3 that the ANN is more precise than the GARCH model for the prediction of volatility.

The total number of cases in which the mid-price with the risk premium was exceeded using the ANN approach was not expected to be so much greater than with the GARCH model. We have seen in the charts shown in Figure 6.2.4 that the GARCH model approach tends to lead to a higher risk premium than the ANN approach. Therefore, the instances of the mid-price exceeding the quoted price are fewer for the GARCH approach than the ANN approach.

The goal of the risk premium should not be that every quotation leads to a positive outcome. It is more important for the risk premium to cover the worst 1% of the fluctuation in the market with the previously earned premiums. The precise calculation achieved by the ANN, represented by a very low MSE, is therefore still a better outcome. However, due to the more accurate estimation, outliers cannot be perfectly covered with the ANN approach.

In our opinion, the critical measurement is the MSE, where we can see on average that the ANN approach resulted in a positive outcome and was close to zero.

Implication of the Results:
Previous research was mainly focused on the volatility prediction for daily-period data. We saw in most related studies that the ANN approach often outperforms the classical approaches, such as the GARCH models. These results demonstrate that the same is also true for short-term periods on the cryptocurrency market.

However, we are challenging the existing theory for volatility clusters. We saw that the clusters in the short-term period are not definable in the same way as with daily data. This could also imply that the volatility in the short term is mainly coincidental and cannot be calculated from past data.

7.2 Limitations

This paper shows that the prediction of volatility using the ANN approach results in a more precise risk premium than that of the standard GARCH approach for the test period in the cryptocurrency field for the pairs BTC-USD, ETH-USD and ETH-BTC.

It is not certain that the same outcome would be seen for other periods or for another time frame. We examined two pairs where cryptocurrencies are exchanged for FIAT money, and one where a cryptocurrency is exchanged for a cryptocurrency. The trading pairs we chose are among the ones with the highest trading volume in the cryptocurrency field. It is likely that this approach can also be used for other pairs, but this has not been tested by this study.

Another limitation is the possibility of volatility changes in the market being driven by coincidence. The model would then not be beneficial for volatility prediction if most of the volatility occurs with no relation to the influence of the past. If the period is shortened even further, the predictive power would most likely decrease.

Due to a lack of available data, the results cannot confirm whether, in any other market situation, the model could produce similar results. In our data, there were times of very low and high volatility. However, the high volatility phase was most likely influenced by the corona crisis. Therefore, any other crisis or volatility phase would lead to different market impacts. We cannot say whether a similar outcome is likely in other situations.

The outcome does not tell us that the ANN approach is the only useful approach. The limitation here is that we cannot retrace how the result is obtained (black box) – it is unclear which input variable has which impact on the outcome. Therefore, the classical approach should be used as a guideline for the prediction comparison.

It is beyond the scope of this study to say whether an ANN volatility prediction approach could also be used for short-term volatility prediction in the stock, bond, or any other financial market.

We have made different choices for the various methods employed, and this could also have an impact on the result. For example, we chose a 10-minute prediction period from the data of one cryptocurrency exchange for six months. Taking more data for a longer time and using a different prediction period could have led to a different outcome. Furthermore, the method for the benchmark or the risk premium calculation could have been different. The comparison method using the MSE has most the most significant impact on the result. By using a different approach to compare the various plans, the same data could have led to different interpretations and results. We also did not prepare a valuation data set to tune the parameters.

7.3 Future Work and Recommendations

Further research is needed to establish the short-term period of the cryptocurrency market returns related to the volatility and the reason for the volatility. Work needs to be done to find out if the volatility appears to be mostly random or due to coincidence connected with external causes, or if past data can also explain short-term volatility.

In this study we tested the basic approaches of the ANN and the GARCH models. Future studies could focus on short-term volatility measurements with different GARCH approaches such as the TGARCH, GJRGARCH, etc. The different GARCH models should also be tested with varying amounts of lagged conditional variances and lagged squared innovations.

The ANN related to short-term volatility prediction should also be tested with LSTM and CNN models.

Risk management can also be carried out using different approaches. We focused on the expected shortfall. Further studies could also take into account the option pricing method for risk premium calculation or other financial, market-related approaches.

7.4 Summary

The outcome of this study shows that the ANN approach gives a more precise volatility prediction for a short-term period of 10 minutes in the cryptocurrency market. The ANN approach was compared with the standard volatility prediction method of GARCH (1, 1).

The ANN volatility prediction made use of input 18 variables. The result of the ANN approach showed that the R^2 score yields a better outcome for the training data than for the test data. Therefore, different overfitting methods were tested to counteract this occurrence.

The comparison between the two different volatility prediction models was conducted using the MSE. The ANN method outperformed the GARCH approach for all three trading pairs we chose. Using the volatility prediction, we calculated the risk premium for the next period to provide a price quotation.

The risk premium was calculated with the expected shortfall where we defined that the worst 1% of the market movement would have to be covered by the risk premium. The risk premium was calculated with the ANN and the GARCH approaches for the volatility prediction. The outcome of the risk premium was compared with the MSE. The MSE also showed that the ANN volatility approach performed better for all three pairs as compared to the GARCH approach. However, we also looked at the absolute number of cases in which the risk premium resulting from the ANN was too small to cover volatility movements. There, the data showed that the ANN, as compared to the GARCH approach, more frequently calculated a risk premium which was too small to cover the price fluctuation. However, the amounts by which the quoted price with the risk premium was exceeded were very miniscule.

Thus, the risk premium calculation using the ANN approach provided a more precise prediction which was very close to the actual volatility. The GARCH approach tends to a higher estimate of the risk, and that is why, in absolute terms, the GARCH approach has fewer cases of the quoted price being exceeded. However, the predictive power for short-term volatility is still low, even with the ANN approach.

Overall, the ANN approach yields a more precise calculation for the volatility and thereby the risk premium compared to the GARCH approach.

8 Conclusion

This research aimed to identify the calculation of a risk premium for a short-term price quotation for the cryptocurrency market. We analyzed different volatility prediction models, and with the forecast of the volatility, we calculated a risk premium for a price quotation with a 10-minute period.

The study demonstrated the predictive power for volatility of the classic approach using the GARCH model and a machine learning approach with the use of an ANN. The results indicate that the ANN produces a more precise volatility prediction compared to the classical method for the given data and trading pairs. However, problems still occur with the volatility prediction of an ANN.

The GARCH model and the risk premium calculation were done using the Student's t distribution approach. This was a necessary step because the data clearly show a leptokurtic distribution of the returns. The forecasts were done for two weeks from the beginning of April until the middle of April 2020. The outcome showed that the ANN volatility prediction is more precise than that of the GARCH model. However, when faced with outliers – price changes in the market with no advance warning – the ANN cannot produce good results. Generally speaking, we observed that the GARCH model often overestimated the risk in the market. Therefore, the absolute number of cases in which the risk premium covered the price change was better with the classic GARCH models. However, the prediction precision was much better with the ANN.

The test period for the risk premium calculation and the volatility prediction took place at a time in which the market faced high volatility. We had a total of five and a half months from mid-October 2019 to the end of March 2020 for the training of our prediction models. In the first five months, the market evidenced a period of very low volatility. In the middle of March, the market started to fluctuate, most likely because of the impact of the corona crisis. Thus, we were testing our models in a very volatile period.

The predictive power of the ANN clearly outperformed the GARCH approach. However, for the ANN volatility forecasting model, we discovered that the model will only explain part of the residuals in the market. This could lead to the assumption that most of the volatility in the short-term view occurs by coincidence and cannot be predicted accurately with past data.

It must also be pointed out that we were only observing three cryptocurrency pairs for six months, where the prediction time was only two weeks. Previous research has mostly focused on daily volatility data. We discerned that the market assumptions which were valid for daily-basis approaches would probably be different for shorter terms. The short-term volatility data challenge the volatility cluster assumption from the traditional daily markets. We saw that in highly volatile times for the short-term, the volatility clusters were also split into clusters of high and low volatility.

Predicting the volatility of the market is a risky operation. For the use of realistic and live volatility prediction and the risk premiums calculated thereby, we have to take into account the trading or transaction costs.

Overall we can say that the ANN approach gives a more precise volatility prediction than the classical approach. Even if the ANN does not explain all of the asset, it does offer a more accurate forecast than the traditional method. Thus, our findings are in line with the hypothesis of this thesis.

List of Figures

List of Tables

References

Adhikari, R., & Agrawal, R. K. (2014). A combination of artificial neural network and random walk models for financial time series forecasting. *Neural Computing and Applications*, *24*, 1441-1449.

Agarap, A. F. (2018, 03). Deep learning using rectified linear units (relu).

Alpaydin, E. (2004). *Introduction to machine learning*. Massachusets London: The MIT Press.

Bailey, D., Borwein, J., Lopez de Prado, M., & Zhu, Q. (2014, 05). Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance. *Notices of the American Mathematical Society*, *61*, 458.

Balcilar, M., Bouri, E., Gupta, R., & Roubaud, D. (2017). Can volume predict bitcoin returns and volatility? A quantiles-based approach. *Economic Modelling*, *64*, 74-81.

Begusic, S., Kostanjcar, Z., Stanley, E., & Podobnik, B. (2018). Scaling properties of extreme price fluctuations in bitcoin markets. *Physica A: Statistical Mechanics and its Applications*, *510*, 400 - 406.

Bildirici, M., & Ersin, Ö. Ö. (2009). Improving forecasts of garch family models with the artificial neural networks: An application to the daily returns in istanbul stock exchange. *Expert Systems with Applications*, *36*(4), 7355 - 7362.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *The Journal of Political Economy*, *81*, 637-654.

Bodie, Z., Kane, A., & Marcus, A. J. (2018). *Investments* (Eleventh Edition ed.). New York: McGraw-Hill Education.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasdicity. *Journal of Econometrics*, *31*, 307-327.

Bollerslev, T. (1987). A conditionally heteroskedastic time series model for speculative prices and rates of return. *The Review of Economics and Statistics*, *69*(3), 542–547.

Brailsford, T. J., & Faff, R. W. (1995). An evaluation of volatility forecasting techniques. *Journal of Banking & Finance*, *20*, 419-438.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5-32.

Bridle, J. S. (1990). *Neurocomputing - Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Brooks, C. (2008). *Introductory econometrics for finance* (Second Edition ed.). New York: Cambridge University Press.

Brooks, C., Clare, A. D., Molle, J., & Persand, G. (2005). A comparison of extreme value theory approaches for determining value at risk. *Journal of Empirical Finance*, *12*, 339-352.

Cao, C. Q., & Tsay, R. S. (1992). Nonlinear time-series analysis of stock volatilities. *Journal of Applied Econometrics*, *1*, 165-185.

Corbet, S., Lucey, B., Urquhart, A., & Yarovaya, L. (2019). Cryptocurrencies as a financial asset: A systematic analysis. *International Review of Financial Analysis*, *62*, 182-199.

Corbet, S., McHugh, G., & Meegan, A. (2017). The influence of central bank monetary policy announcements on cryptocurrency return volatility. *Investment Management and Financial Innovations*, *14*, 60-72.

Cryer, J. D., & Chan, K.-S. (2008). *Time series analysis with applications in r* (Second Edition ed.). New York: Springer Science + Business Media.

Danielsson, J., & de Vries, C. G. (2000). Value-at-risk and extreme returns. *Annales D Economie et de Statistique*, *60*, 239-270.

Danielsson, J., & Moritomo, Y. (2000). Forecasting extreme financial risk: a critical analysis of practical methods for the japanese market. *Monetary Economic Studies*, *12*, 25-48.

Devore, J. L. (2011). *Probability and statistics for engineering and the sciences* (Eight Edition ed.). Boston: Cengage Learning.

Domingos, P. (1999). The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, *3*, 409-425.

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

Dowd, K. (2005). *Measuring market risk* (Second Edition ed.). West Sussex: John Wiley & Sons Ltd.

Eberlein, E., Kallsen, J., & Kristen, J. (2002). Risk management based on stochastic volatility. *Journal of Risk*, *5*, 19-44.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of uk inflation. *Econometrica*, *50*, 987-1008.

Fischhoff, B., Watson, S. R., & Hope, C. (1984). Defining risk. *Policy Sciences*, *17*, 123-139.

Gajardo, G., Kristjanpoller, W. D., & Minutolo, M. (2018). Does bitcoin exhibit the same asymmetric multifractal cross-correlations with crude oil, gold and djia as the euro, great british pound and yen? *Chaos, Solitons and Fractals*, *109*, 195 - 205.

Gencay, R., Selcukb, F., & Ulugülyagci, A. (2003). High volatility, thick tails and extreme value theory in value-at-risk estimation. *Insurance: Mathematics and Economics*, *33*, 337-356.

Gentsch, P. (2019). *Künstliche Intelligenz für Sales, Marketing und Service* (Second Edition ed.). Wiesbaden: Springer Gabler.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. (`http://www.deeplearningbook.org`)

Granger, C. W. J., & Ding, Z. (1995). Some properties of absolute return an alternative measure of risk. *Annales d Economie et de Statistique*, *40*, 67-91.

Granger, C. W. J., & Poon, S.-H. (2004). Forecasting financial market volatility: A review. *International Journal of Forecasting*, *20*, 273-286.

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, *38*(8), 10389 - 10397.

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

Hamid, S. A., & Iqbal, Z. (2004). Using neural networks for forecasting volatility of s&p 500 index futures prices. *Journal of Business Research*, *57*(10), 1116 - 1125. (Selected Papers from the third Retail Seminar of the SMA)

Hebb, D. O. (1949). *Organization of Behavior*. John Wiley & Sons Inc.

Heynen, R. C., & Kat, H. M. (1994). Volatility prediction: A comparison of the stochastic volatility, garch (1,1) and egarch (1,1) models. *Journal of Derivatives*, *2*, 50-65.

Hochegger, G. A. (2010). *Value at Risk - Risikomanagement von linearen und nicht linearen Finanzinstrumenten*. Graz.

Hull, J. C. (2018). *Risk management and financial institutions* (Fifth Edition ed.). Hoboken: John Wiley & Sons, Inc.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, *abs/1502.03167*.

Jonen, A. (2007). Semantische Analyse des Risikobegriffs - Strukturierung der betriebswirtschaftlichen Risikodefinitionen und literaturempirische Auswertung. *Beiträge zur Controlling-Forschung*, *11*.

Jorion, P. (2006). *Value at risk - the new benchmark for managing financial risk* (Third Edition ed.). New York: McGraw-Hill Education Ltd.

Kacper Ciesla. (2020). *Exchanges ranking.* Retrieved 01. Jun. 2020, from `https://data.bitcoinity.org/`

Katsiampa, P. (2017). Volatility estimation for bitcoin: A comparison of garch models. *Economics Letters*, *158*, 3-6.

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization.* Retrieved 10. Jun. 2020, from `https://arxiv.org/abs/1412.6980`

Koutmos, D. (2018a). Bitcoin returns and transaction activity. *Economics Letters*, *167*, 81-85.

Koutmos, D. (2018b). Liquidity uncertainty and bitcoins market microstructure. *Economics Letters*, *172*, 97 - 101.

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

Kristjanpoller, W., & Hernandez, E. (2017). Volatility of main metals forecasted by a hybrid ann-garch model with regressors. *Expert Systems with Applications*, *84*, 290 - 300.

Kristjanpoller, W., & Minutolo, M. C. (2015). Gold price volatility: A forecasting approach using the artificial neural network garch model. *Expert Systems with Applications*, *42*(20), 7245 - 7251.

Kristjanpoller, W., & Minutolo, M. C. (2018). A hybrid volatility forecasting framework integrating garch, artificial neural network, technical analysis and principal components analysis. *Expert Systems with Applications*, *109*, 1 - 11.

Kubat, M. (2017). *An introduction to machine learning* (Second Edition ed.). Cham: Springer International Publishing AG.

Lahmiri, S., & Bekiros, S. (2019). Cryptocurrency forecasting with deep learning chaotic neural networks. *Chaos, Solitons and Fractals*, *118*, 35-40.

Lahmiri, S., Bekiros, S., & Salvi, A. (2018). Long-range memory, distributional variation and randomness of bitcoin volatility. *Chaos, Solitons and Fractals*, *107*, 43 - 48.

Lee, K. Y. (1991). Are the garch models best in out-of-sample performance? *Economics Letters*, *37*, 305-308.

Lei, L. (2018). Wavelet neural network prediction method of stock price trend based on rough set attribute reduction. *Applied Soft Computing*, *62*, 923 - 932.

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

Mandelbrot, B. (1963). The variation of certain speculative prices. *The Journal of Business*, *36*, 394-419.

Marra, Stephen. (2015, December). *Predicting volatility.* Retrieved 29. Feb. 2020, from `https://www.lazardassetmanagement.com/docs/-m0-/22430/predictingvolatility_lazardresearch_en.pdf`

Masulli, F., Petrosino, A., & Rovetta, S. (2012). *Clustering high-dimensional data* (Second Edition ed.). Heidelberg: Springer-Verlag Berlin Heidelberg.

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

McNeil, R., Alexander J.and Frey, & Embrechts, P. (2005). *Quantitative risk management*. Princeton: Princeton University Press.

Mitchell, T. M. (1997). *Machine learning*. New York City: McGraw-Hill Science/Engineering/Math.

Mitra, S. (2015). The relationship between conditional value at risk and option prices with a closed-form solution. *The European Journal of Finance*, *21*, 400-425.

Miura, R., Pichl, L., & Kaizoji, T. (2019, 06). Artificial neural networks for realized volatility prediction in cryptocurrency time series. In (p. 165-172).

Monfared, S. A., & Enke, D. (2014). Volatility forecasting using a hybrid gjr-garch neural network model. *Procedia Computer Science*, *36*, 246 - 253.

Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with python a guide for data scientists* (First Edition ed.). Sebastopol: O'Reilly Media.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml.*

Nakamoto, Satoshi. (2008). *Bitcoin: A peer-to-peer electronic cash system.* Retrieved 04. Apr. 2020, from `https://bitcoin.org/bitcoin.pdf`

Neftci, S. N. (2000). Value at risk calculations, extreme events, and tail estimation. *Journal of Derivatives*, *7*, 23-38.

Nelson, D. B. (1991). Heteroskedasticity in asset returns: A new approach. *Econometrica*, *59*, 347-370.

Nielsen, M. (2018). *Neural networks and deep learning.* Retrieved 10. Jun. 2020, from `http://neuralnetworksanddeeplearning.com/`

Osterrieder, J., & Lorenz, J. (2017). A statistical risk assessment of bitcoin and its extreme tail behavior. *Annals of Financial Economics*, *12*, 19.

Pawel Lachowicz. (2016). *Conditional value-at-risk in the normal and student t linear var model.* Retrieved 10. Jun. 2020, from `http://www.quantatrisk.com/2016/12/08/conditional-value-at-risk-normal-student-t-var-model-python/`

Phillip, A., Chan, J. S., & Peiris, S. (2017). A new look at cryptocurrencies. *Economics Letters*, *163*, 6-9.

Pichl, L., & Kaizoji, T. (2017). Volatility analysis of bitcoin price time series. *Quantitative Finance and Economics*, *1*, 474-485.

Radityo, A., Munajat, Q., & Budi, I. (2017). Prediction of bitcoin exchange rate to american dollar using artificial neural network methods. In *2017 international conference on advanced computer science and information systems (icacsis)* (p. 433-438).

Rashid, T. (2017). *Neuronale Netze selbst programmieren* (First Edition ed.). Heidelberg: dpunkt.Verlag GmbH.

Reed, R., & Marksll, R. (1999). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. A Bradford Book.

Roh, T. (2007). Forecasting the volatility of stock price index. *Expert Systems with Applications*, *33*(4), 916 - 922.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386-408.

Rumelhart, D. E., & McClelland, J. L. (1987). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition: Foundations* (p. 318-362).

Runkle, D. E., Glosten, L. R., & Jagannathan, R. (1993). On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance*, *48*, 1779-1801.

Sheraz, M., & Preda, V. (2014). Implied volatility in black-scholes model with garch volatility. *Procedia Economics and Finance*, *8*, 658-663.

Sin, E., & Wang, L. (2017). Bitcoin price prediction using ensembles of neural networks. In *2017 13th international conference on natural computation, fuzzy systems and knowledge discovery (icnc-fskd)* (p. 666-671).

Spilak, B. (2018). *Deep neural networks for cryptocurrencies price prediction*. Berlin: Humboldt-Universität zu Berlin.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks fromoverfitting. *Journal of Machine Learning Research*, *15*, 1929-1958.

Stier, W. (2001). *Methoden der Zeitreihenanlayse*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg GmbH.

Student. (1908). The probable error of a mean. *Biometrika*, 1-25.

Takaishi, T. (2018). Statistical properties and multifractality of bitcoin. *Physica A: Statistical Mechanics and its Applications*, *506*, 507 - 519.

Taylor, J. W. (2004). Volatility forecasting with smooth transition exponential smoothing. *International Journal of Forecasting*, *20*, 273-286.

Vapnik, V. N. (1998). *Statistical Learning Theory*. New York: John Wiley & Sons Inc.

Wiecki, T., Campbell, A., Lent, J., & Stauth, J. (2019, March). *All that glitters is not gold: Comparing backtest and out-of-sample performance on a large cohort of trading algorithms.*

Yao, Y., Rosasco, L., & Caponnetto, A. (2007, 08). On early stopping in gradient descent learning. *Constructive Approximation*, *26*, 289-315.

Yi, S., Xu, Z., & Wang, G.-J. (2018). Volatility connectedness in the cryptocurrency market: Is bitcoin a dominant cryptocurrency? *International Review of Financial Analysis*, *60*, 98 - 114.