Universität Osnabrück

Institut für Kognitionswissenschaft

Master's Thesis

Topic: Explorative Study on Financial Time Series Analysis with Deep Machine Learning Models

Author: Tillmann U. Brodbeck, BSc.
Master's Program Cognitive Computing
tbrodbeck@uos.de
Matriculation 964383

Version: September 19, 2021

1. Supervisor: Prof. Dr. Kai-Uwe Kühnberger
2. Supervisor: Ludwig Schallner, MSc.

Abstract

Due to low interest rates and more availability of low-commission brokers, financial markets are consulted by retail investors more than ever before. Accordingly, the interest in financial advice raises when the general trading volume increases. Moreover, there is a lot of progress in the field of Artificial Intelligence. In this work it was analyzed how easy it is for Artificial Intelligence to understand and successfully predict financial markets. Therefore, this work analyzes the behavior of a range of Machine Learning models on several financial and non-financial datasets. The financial data comprise the stock Index MSCI World, Forex data of the euro price and cryptocurrency data of the Bitcoin price. Several configurations of XGBoosts, MLPs and LSTMs were then used to predict this data. Thereby, preprocessing techniques such as scaling and technical indicators were applied and evaluated. At final, most models were unable to outperform the main baseline on financial datasets even though the models were able to predict other time series types successfully. As a reason for this behavior, it is suggested that financial institutions are using quantitative finance at larger scale already thus making the markets efficient. Overall, it is suggested while it is possible to some degree, it is difficult for small scale investors to predict financial time series with Artificial Intelligence on time series characteristics alone.

Contents

List of Figures

List of Tables

Acronyms

Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Network
ATR	Average True Range
DeFi	Decentralized Finance
EMH	Efficient Market Hypothesis
EMA	Exponential Moving Average
ETF	Exchanged Traded Fund
Forex	Foreign Exchange
GBLinear	Gradient Boosting Linear Model
GBTree	Gradient Boosting Tree
LR	Linear Regression
LSTM	Long Short-Term Memory
MACD	Moving Average Convergence Divergence
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
MXWO	MSCI World
OHLCV	Open-High-Low-Close-Volume
OTC	Over-The-Counter
P2P	Peer-to-Peer
Prev	Previous value
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RSI	Relative Strength Index
TR	True Range
XGBoost	eXtreme Gradient Boosting
VWAP	Volume Weighted Average Price

1 Introduction

1.1 Motivation

Investing into financial markets has become an increasingly popular topic in recent times. This is the case due to several reasons. First, since pension systems are becoming more unreliable, more old people are prone to face poverty after retirement. Therefore, saving money is becoming more important to create a passive retirement income. And because traditional bank account savings are yielding negative interest, saving the money in financial securities becomes more important. Especially since Coronavirus with people spending more time at home, a cohort of retail investors started investing and trading on financial markets [Dom21]. Moreover, financial brokerages continuously allow cheaper commissions for buying and selling financial positions, what results in low barriers to enter financial markets for retail traders [Reg20].

Furthermore, there are new financial systems developing besides the classical financial markets like the stock or the Foreign Exchange (Forex) market. These are new systems based on decentralized cryptocurrencies that are stored on blockchain networks. The first and most popular cryptocurrency is Bitcoin, and it was created as a decentralized currency for Peer-to-Peer (P2P) transactions without the need of a third entity like a bank [Nak08]. In 2020 and 2021 the prices of cryptocurrencies experienced an extraordinary price trend. The two biggest cryptocurrencies Bitcoin and Ethereum[1] reached new all-time-highs in the beginning of 2021 and cryptocurrencies in total surpassed a total market capitalization of $2 trillion USD in April 2021 [Che21b]. This surge also increased the usage of Decentralized Finance (DeFi) markets which are decentralized financial systems governed by algorithms. These algorithms were written in smart contracts that are stored in blockchains like the Ethereum Blockchain or the Binance Smart Chain[2].

This surge of invested money into financial markets by retail investors also motivates investment banks to profit from these increased cash flows. For example, the British multinational bank Barclays created a profitable trading strategy based on volatility prediction on large-capital US tech stocks. With this strategy they are able to profit from the increased high risk trading strategies inspired through internet forums such as WallStreetBets[3] [SPGK21]. In general, investment banks often trade with algorithms that are specialized in detecting behavioral patterns that influence human investing and trading behavior, such as trend or momentum following [DP18].

Another field of technology that experiences rapid growth in recent years is the field of Artificial Intelligence (AI) and Machine Learning (ML). Due to the increased computational power that has become available over the course of the last decade, particularly

[1] https://ethereum.org/en
[2] https://www.binance.org/en/smartChain
[3] https://www.reddit.com/r/wallstreetbets

regarding high performance graphics cards, Artificial Neural Networks (ANNs) have been used more frequently and this has driven the research focus on computationally intensive AI models. Especially in the field of computer vision, ANNs like the Residual neural network [HZRS15] and the Inception Networks [SLJ+14] have shown the best performance in recent Kaggle competitions[4]. These competitions provide a recognized benchmark for ML algorithms. In the area of sequence learning, Recurrent Neural Network (RNN) are also becoming more popular. This is the case in the field of AI game playing, such as competing against professional players in popular video games like DotA 2 [SCS16], and StarCraft II [VEB+17], and also in natural language processing [GVS+16]. Additionally, to these domains, time series prediction is another area that is frequently solved using ML algorithms. Time series prediction encompasses weather forecasting, predictive logistics or financial market prediction.

From these two technological trends the motivation for this work arises to explore how financial systems can be predicted using ML techniques and neural networks. For the majority of cases it is not publicly available what kind of algorithmic trading investment banks are doing. The reason for this is that if their strategies would be public it would not be possible for them to profit from them anymore. This is because financial trading is a Zero-sum game [Ken21]. Therefore, this work wants to analyze the performance of ML models and AI for financial time series prediction.

1.2 Objective

As motivated in the previous section the main goal of this work is to explore different possibilities how to predict financial time series with AI. The following research questions Q1-Q4 will be analyzed in this work:

- Q1: Is it possible to build an algorithm to predict financial time series based on its price and volume data using Machine Learning techniques?
- Q2: How are different Machine Learning prediction models differing in their prediction quality?
- Q3: How are those prediction models behaving on different datasets?
- Q4: In what ways is it possible to improve those models' prediction qualities?

1.3 Structure

This work will be structured as follows: chapter 2 presents the underlying background knowledge and aids understanding of the algorithms used later. It covers the basic principles of financial markets and introduces different types of markets. It also explains

[4]https://www.kaggle.com/competitions

the most important properties of financial time series and the basic principles of technical market analysis. Furthermore, the theoretical foundations of the used Machine Learning models is explained. Chapter 3 explains the specific data sets, preprocessing techniques and evaluation methodology used in this work. The following chapter 4 describes the conducted experiments and their results. The final chapter 5 discusses and interprets the results of the experiments.

2 Fundamentals

This chapter will cover the foundational concepts upon which this work is based. The aim of this section is to provide any reader with enough background information about the models and learning algorithms used, to enable them to understand all the following chapters.

The first section introduces financial markets. Doing this, the stock market, the Forex market and the cryptocurrency market are introduced. In the following section 2.2 the concept of financial time series is explained. A selection of the most important characteristics of financial time series are described in detail. After that, in section 2.3, the foundations of the technical analysis of financial time series are elaborated. For that the concepts of market action and indicators are introduced. Subsequently, in section 2.4 the prediction metrics used in this work are presented. And finally, the different types of prediction models used in this work are explained in section 2.5.

2.1 Financial market

A financial market is any marketplace where the trading of financial securities occurs, including among others the stock market, Forex market, and the cryptocurrencies market. These markets allow for fast and transparent buying and selling of financial holdings. Financial markets create products that provide a security as well as returns for those who have excess funds, i.e. investors, while on the other hand these securities make the funds available to businesses who can use additional money. The assets or securities of these markets are either listed on regulated exchanges or are traded Over-The-Counter (OTC). Financial markets are important for the smooth operation of capitalist economies, because they help entrepreneurs and companies to form capital and have liquidity. When the dynamics of financial markets fail, as it the case in a financial crisis, the economic disruption can lead to recession and unemployment [Hay21a].

There are two kinds of participants on financial markets: the investors and traders on one side, and market makers and liquidity providers on the other side. Additionally, there are third parties that facilitate the trades between the sellers and the buyers. These third parties are the brokers and exchanges [Hay21a]. When securities are bought or sold, investors are exposed to transaction costs in the form of commissions, fees, and bid-ask spreads [CZ05]. This is dependent on the exchange or brokerage where the trade was made.

In the further course of this section, the relevant markets used in this work are introduced: the stock market, Forex, i.e. the fiat currency market, and lastly the cryptocurrency market.

2.1.1 Stock market

The most popular financial market is the stock market [Ama21]. In this market companies list their equity shares, such that they can be traded by retail and professional investors [Hay21a].

Mutual funds enable buying a lot of stocks at once. In a way, this makes mutual funds an easier tool to invest in than individual stocks. By reducing stock market volatility, they have a calming effect on an economy as well [Ama21]. The composition of securities in a fund are classically managed by a person. In contrast, the composition of Exchanged Traded Funds (ETFs) are algorithmically managed. There are indices that represent weighted averages of security prices that can be traded as ETFs: some of the most popular indices are the S&P 500[5], Dow Jones Industrial Average[6], Nasdaq-100[7] [CZ05] and MSCI World[8] indices. Hence, through an ETF an investor can invest into a complete market area.

The largest listed stock exchanges are the New York Stock Exchange[9] (NYSE) or Nasdaq, and additionally stocks can be traded OTC. Most trading is done on these regulated exchanges, and therefore these play an important role in the economy. The traded prices measured on an exchange can gauge of the overall health of particular economy [Hay21a]. The stock exchanges are opened depending on the working hours of the country where they are located. An overview of the opening hours of international stock exchanges can be viewed at WorldTimeZone[10].

2.1.2 Forex market

The Forex market allows buying and selling on the exchange rates between fiat currency pairs. Doing this, the Forex market forms the most liquid market in the world, because currencies are the most liquid financial asset. In 2021, the Forex market handled more than $5 trillion USD of daily transactions. This is more than double the volume of the stock market. Similar to OTC markets, the Forex market is decentralized. It consists of a global network of algorithmic brokers that regulate the currency trading [Hay21a]. There are five Forex market centers in Frankfurt, London, New York, Tokyo and Sydney. Each market center has specific opening hours depending on local working hours [Tim21]. The forex market opening hours are displayed in table 2.1.

[5] https://www.spglobal.com/spdji/en/indices/equity/sp-500
[6] https://www.spglobal.com/spdji/en/indices/equity/dow-jones-industrial-average
[7] https://www.nasdaq.com/nasdaq-100
[8] https://www.msci.com/documents/10199/178e6643-6ae6-47b9-82be-e1fc565ededb
[9] https://www.nyse.com/index
[10] https://www.worldtimezone.com/markets24.php

	Time zone	Opens	Closes
Frankfurt, Germany	Berlin	06:00	14:00
London, United Kingdom	London	07:00	15:00
New York, United States	New York	12:00	20:00
Sydney, Australia	Sydney	22:00	06:00
Tokyo, Japan	Tokyo	23:00	07:00

Table 2.1: Forex market centers' opening and closing times in GMT (Greenwich Mean Time) format [Tim21]

2.1.3 Cryptocurrency market

Cryptocurrencies are decentralized digital financial assets based on blockchain technology. In 2021 there are a plethora of cryptocurrency tokens available to be traded globally and at all times across several independent online crypto exchanges. These exchanges have digital cryptocurrency wallets for traders to swap cryptocurrencies into other cryptocurrencies, as well as to trade cryptocurrencies against fiat money such as dollar or euro.

The majority of cryptocurrency exchanges are centralized platforms. There are also decentralized exchanges available that operate without any central authority instead they are governed by algorithms. These exchanges are based on smart contracts written into blockchain technology such as the Ethereum Virtual Machine[11] or the Binance Smart Chain. Decentralized exchanges allow direct P2P trading of digital currencies trustless, i.e. without the need for third party to facilitate the transactions. Because cryptocurrency exchanges are far less regulated than other financial markets they are more susceptible to hacks or fraud [Hay21a].

2.2 Financial time series

For the stability of financial markets it is considered essential to be able to form predictions about the time series of the securities traded on this market e.g. for risk modeling or capital planing. There are many effects on the financial time series of a security according to modern financial market theory [Vog21]. When creating predictions about financial time series these effects must be considered. Said effects are especially strong in times of crisis and represent the influence of the non-rational trading behavior of agents in the market. The most important properties of financial time series [Vog19] are listed in this section.

2.2.1 Non-stationarity

A time series is said to be stationary if it has a constant mean, a constant variance and a covariance which depends on the time between lagged observations at t_1 and t_2

[11] https://ethereum.org/en/developers/docs/evm

only. The covariance is described by this formula:

$$cov_{x,y} = \frac{\sum_{t=1}^{T}(x_{t_1} - \bar{x})(x_{t_2} - \bar{x})}{T-1} \tag{2.1}$$

With the Dickey-Fuller test it can be tested whether a time series is stationary [WP97].

Nevertheless, financial time series contain trends as well as seasonality. Therefore, the values of it are affected differently at different points in time. Therefore, financial time series are not stationary. [HA18]

2.2.2 Non-linearity

When a time series is linear each value x_t can be modeled successfully as a linear combination of past values or differences. For example, by applying this approach a linear model described by

$$x_t = .9x_{t-1} + .81x_{t-2} + ... + \epsilon_t \tag{2.2}$$

could be created. In this equation ϵ_t denotes an error term.

This approximation is not able to model financial time series successfully, therefore a financial time series can be considered as non-linear [Vog19].

2.2.3 Interdependency

The values of a financial time are interdependent. For example, the price value x_{t_2} of a time series is dependent on its previous price x_{t_1} for $t_1 < t_2$. Furthermore, the trading volume of a financial time series is interdependent as well [Vog19].

2.2.4 Asymmetry

In addition, financial time series have asymmetric characteristics such as a dynamic volatility. Volatility in finance describes the extent how much the values of the time series fluctuate around the mean value of a time period. Volatility is autocorrelative, that means it is dependent on its own historical development. Besides, it is highly persistent, what means its effect stays for a long time.

The volatility of a time series consists of both, positive and negative volatility, which behave differently. Positive volatility is the volatility that is caused by good news, whereas negative volatility is caused by bad news. In contrast to positive volatility, negative volatility contains a high-frequent mean-reverting component. This means that the values of the time series swing around the mean of the series with a high frequency. The effect of this component decreases over time [Vog21]. Because of these effects, financial time series do not have heteroscedasticity. Heteroscedasticity refers to the changing of the standard deviation over time [Hay21b]. Therefore, the volatility

cannot be modeled by a Gaussian normal distribution [Vog21] and financial time series are considered as asymmetric.

2.2.5 Trend

It can be observed that there is the effect that positions that performed well in the past tend to perform well in the future as well. Furthermore, these trends in financial time series follow a log-normal distribution [Vog19][Ros21].

There are positive and negative trends. In financial terminology a rising market is called bullish, while a falling market is called bearish.

2.2.6 Frequency information

Additionally, financial time series contain frequency information. This information can be exploited using wavelets [Vog19] or similar filters. These filters have to be applied to a time series transformed into a frequency space.

2.2.7 Fractality

Furthermore, financial time series contain fractal patterns. These patterns can be observed invariant of the scale. This phenomenon is based on the facts that irrational trading is happening using every investment horizon, short term day-trading or long term investing over years. Therefore, at every timescale effects of time series can be found that originate from irrational trading behavior [Vog21].

2.2.8 Semi-efficiency

The Efficient Market Hypothesis (EMH) is a backbreaker for forecasting. In its simplest form it says that financial time series are unforecastable [TG04]. Statistical research has shown that to a close approximation financial time series seem to follow a random walk with no predictable patterns that are exploitable for investors. Such findings are now taken to be evidence of market efficiency. Market efficiency or the EMH states that market prices reflect all currently available information. According to the EMH, only new information will move stock prices, and this information is equally likely to be good news or bad news [BKM08].

Three forms of market efficiency are commonly entertained in the EMH literature based on the set of variables contained in the information set Ω_t on which the trading is based upon [Rob21][Fam70]. If Ω_t only comprises past and current asset prices (as well as possibly variables such as trading volume), the EMH in its weak form is being tested. Expanding Ω_t to include all publicly available information gives rise to the EMH in its semi-strong form. Finally, if all public and private information is included in Ω_t, market efficiency in the strong form is being tested [TG04].

In this section it was shown that there are effects of financial time series like momentum or asymmetry. This renders the time series' market not fully efficient. These effects result from the irrationality of trading entities such as memory bias, overconfidence, conservatism, and representativeness. But even though the markets are inefficient, even most professional funds cannot create strategies that successfully outperform the markets. Thus, it can be concluded that security markets are nearly efficient, or semi-efficient. This means most securities are usually priced appropriately respective to their actual performance evaluation [BKM08].

2.3 Technical analysis

Technical analysis is the search for recurring and predictable patterns in security prices [BKM08]. The technical analysis stands in contrast to the fundamental analysis of securities. The fundamental analysis of securities analyzes the economic forces of supply and demand that affect the securities. Technical analysis is different from this because it studies the fundamental properties of securities indirectly. Technical analysis is the study of market action, which is the information given by the price and the volume of a security. Because markets are (semi-)efficient everything that affects market price is ultimately reflected in the market price itself. Following from that, the study of the market price is all that is necessary [Mur99].

Additionally, technical analysis assumes that prices moves in trends as explained in section 2.2.5 and that history repeats itself. When a price trend can be observed it is more likely for the price to continuing the trend than a reversion of the trend. This is based on behaviorism and psychology. Because human psychology tends not to change over time the tools of analyzing and classifying market movements have been developed to predict future market movements [Mur99].

2.3.1 Market action

The market action of a security can be analyzed by its price and volume attribute and its change over time. Classically, price is measured as a four-tuple of open, high, low and close price in a certain time frame. The most prominent way to create a price chart is the display of this four-tuple as Japanese candlesticks [Mur99]. These are exemplified in figure 2.1.

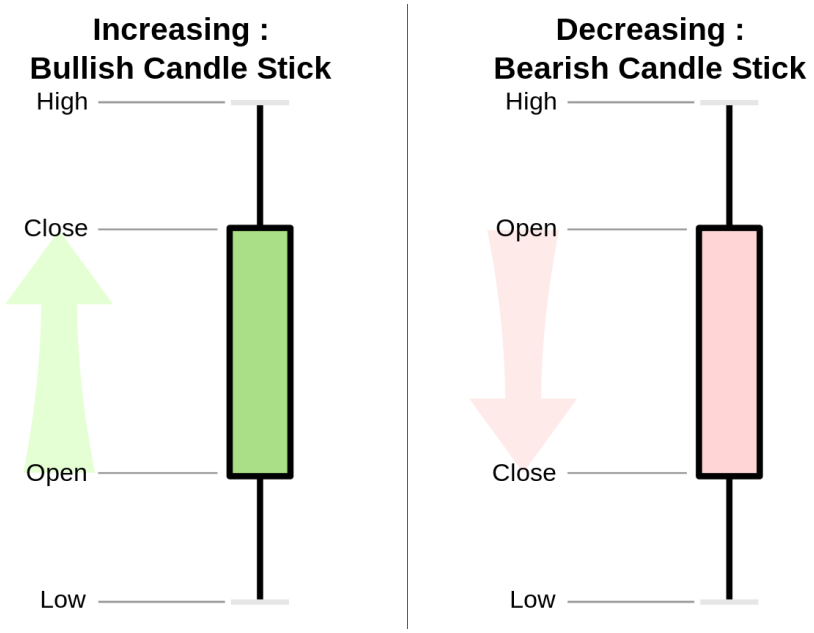


Figure 2.1: Japanese candlesticks[12]; bullish refers to up-trending markets and bearish refers to down-trending markets

The combination of the price four-tuple and the volume is called Open-High-Low-Close-Volume (OHLCV) data. In figure 2.2, an example of a time series of OHLCV market actions is shown.

Figure 2.2: Example OHLCV chartwith data of the IBM[13] stock; the candles represent the price of a month; the trading volume is represented by the blue bars

[12]https://en.wikipedia.org/wiki/File:Candlestick_chart_scheme_03-en.svg
[13]https://www.ibm.com/us-en?ar=1

2.3.2 Indicators

Technical indicators are signals calculated from OHLCV market action of a time series mathematically through technical analysis techniques. The aim of the application of indicators is to create informative signals that can help to predict future price movements. The indicators are mostly applied to understand short-term price movements of a time series. But the indicators can also be applied by investors to choose preferable entry and exit strategies for long-term investments. Often a combination of several technical indicators is used to analyze a financial asset. There are thousands of different technical indicators, so only a small subset can be applied at once. Regularly, the technical indicators are then also combined more subjective forms of technical analysis, such as watching charts or fundamental analysis. Additionally, technical indicators can also be incorporated into trading algorithms. This is what has been done in this work.

Generally, there are two basic types of technical indicators: First, so-called overlay indicators are technical indicators that use the same scale as the price of a series. An example for overlay indicators are moving averages. These indicators are usually plotted on top of the prices on a stock chart. Secondly, there are oscillator technical indicators that move between a fixed minimum and maximum. These indicators are usually plotted above or below a price chart. Examples for oscillators are moving averages, Moving Average Convergence Divergence (MACD), or Relative Strength Index (RSI). Visual examples of these indicators can be seen in figure 2.3 [Che21a].

2.4 Prediction metrics

To be able to evaluate and compare predictions of time series, a sophisticated approach of measuring predictions is needed. Therefore, a metric is calculated that describes the deviation of a forecasted prediction from an actual measurement. Each metric used provides a measure of how well the algorithm performs according to its specific criteria. The different metrics that are applied in the context of this work, the Mean Absolute Error (MAE), Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE), will be listed in the following sections.

2.4.1 Mean Absolute Error

The MAE is a metric that calculates the absolute differences at every time step t between the predicted data point y_t and its actual value, or target value, a_t and takes the mean value of these differences [SW10]. For a time series of length T this can be calculated with the following equation:

$$MAE = \frac{\sum_{t=1}^{T} |a_t - y_t|}{T} \tag{2.3}$$

Figure 2.3: Demonstration of indicator examples: the 50-day (dark-blue) and 200-day (light-blue) moving averages are plotted over the top of the prices to show where the current price stands relative to its historical averages; the 50-day moving averages is higher than the 200-day moving average in this case, which suggests that the overall trend has been positive; the RSI, displayed in yellow above the chart, shows the strength of the current trend—a neutral value of 49.07%; the MACD, shown as 3 graphs below the chart, indicates how the two moving averages have converged or diverged; in this case the negative value of the MACD indicates a down-trend [Fer21b]

2.4.2 Mean Squared Error

Instead of the absolute values in the MAE, the MSE calculates the squares of every pair, x_t and y_t, and then similarly, the mean of these pairs is then calculated [SW10]:

$$MSE = \frac{\sum_{t=1}^{T}(a_t - y_t)^2}{T} \tag{2.4}$$

In particular, a comparison of the MAE and MSE gives interesting insights, e.g. outliers are punished more strongly for the MSE. Comparing both can be used to track how stable the training of an ML model is, i.e. how similar the two metrics stay during the training of a neural network.

2.4.3 Mean Absolute Percentage Error

In contrast to the MAE and MSE, the mean absolute percentage error MAPE gives a relative estimation of the error. To calculate the MAPE, the mean of the division of the differences between a_t and y_t divided by a_t is taken. It returns a value between 0 and 1 for every difference as a percentage measure of how well a_t is approximated compared to its actual value. This value can be multiplied by 100% to create a percentage value:

$$MAPE = \frac{\sum_{t=1}^{T}|\frac{a_t - y_t}{a_t}|}{T} * 100 \tag{2.5}$$

The conventions for the case $a_t = 0$ are such that $\frac{0}{0} = 1$ and if $y_t \neq 0$, $\frac{y_t}{0} = \infty$. In general, the MAPE has the problem of calculating infinities for values close to zero [DMGGR15].

In summary, this error describes how much percent on average the target value of the time series was either overestimated or underestimated. As a percentage measure, it enables scale-independence and a more intuitive interpretability than the MAE and the MSE. Additionally, the values of the MAPE can be compared with each other when evaluating the performance of predictions regarding different datasets. Therefore, this metric was used as the preferred metric to interpret the results of the experiments of this work.

2.5 Machine Learning models

The prediction models that were used in this work are regression models and ANNs. The regression models, or regressors, are kinds of supervised learning models that are applied to fit a time series with respect to the MSE. Additionally, to the MSE based least squares regression, that this work will focus on, there are other kinds of regressions: there is logit, Poisson or negative binomial regression [Cox21].

In this work two types of regression models were used to create time series predictions,

the linear regressor and the eXtreme Gradient Boosting (XGBoost) regressor. The linear regressor was chosen because it is the simplest regressor, and therefore, it can be used as a benchmarking model. It is introduced in the first subsection, section 2.5.1. Additionally, an XGBoost model was used to create predictions, which is introduced in the second subsection 2.5.2.

Moreover, in this work ANNs, or neural networks in short, were used to generate forecasts. These types of models are based on the simulation of biological neurons, and thus, they allow complex calculations similar to the human brain. The most simple form of a neural network is explained in subsection 2.5.3. And in subsection 2.5.4 a more complex, sequentially processing neural network is introduced. And the final subsection 2.5.5 the training procedure of ANNs is described.

2.5.1 Linear regressor

A linear regressor allows for regressions in which a predicted value is calculated as a linear combination of the input features. With x is an input vector of size N, w as the weight matrix of size N and y as the predicted value, a model corresponding to this formula is generated:

$$y(w, x) = w_0 + w_1 x_1 + ... + w_N x_N \tag{2.6}$$

In this work, a least squares linear regression was applied. It fits a linear model with coefficients w to minimize the residual sum of squares between observed targets a, and the prediction of targets y, predicted by this linear approximation in equation 2.6. For this, a mathematical problem of this form is solved:

$$\min_w ||y - a||_2^2 \tag{2.7}$$

The least squares regression is computed using the singular value decomposition of x [Bis06]. Hereby, the estimates for w for the least squares rely on the independence of the features. When features are correlated and the columns of their design matrix have linear dependencies, the least-squares estimate can be sensitive to random errors of the targets a, and then, the model might produce a large variance. This situation could arise in the experiment of this work because financial time series are known to be interdependent, see section 2.2.3. If x is a matrix of shape $n_{\text{samples}} \times n_{\text{features}}$ this algorithm has a computational complexity of $O(n_{\text{samples}} n_{\text{features}}^2)$, assuming that $n_{\text{samples}} \geq n_{\text{features}}$ [PVG+11].

2.5.2 XGBoost regressor

Shortly after its development and initial release in 2016, XGBoost dominated structured or tabular datasets on classification and regression predictive modeling problems. The evidence is that it is the go-to algorithm for competition winners on the Kaggle

competitive data science platform [Bro21][CG16].

Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or time series regression tasks. The ensembles are constructed from decision tree models, an example of a tree generated by XGBoost is listed in the appendix in figures A.4 and A.5. The trees are added one at a time to the ensemble and fit to correct the prediction errors made by prior models. This is a type of ensemble machine learning model referred to as boosting.

Models are fit using any arbitrary differentiable loss function, such as the MSE that was used in this work. Similarly to neural networks, it uses a gradient descent optimization algorithm. Gradient descent is introduced in section 2.5.5. Moreover, a special type of XGBoost that was used in this work called XGBoost regressor. The XGBoost regressor is a kind of XGBoost that is optimized for regression problems like time series curve fitting.

Furthermore, in this work an open-source implementation of the gradient boosting algorithm from the Distributed Machine Learning Community[14] was used. It is implemented in C++, and therefore, it is fast and scalable as well [Bro21].

The XGBoost model can then be used with three different types of boosters. The default booster is called the Gradient Boosting Tree (GBTree) booster. There is also a booster called Dart that uses dropout similar to the dropout used for neural networks, as described in section 2.5.5. Therefore, the effect of a random number of trees in the ensemble is multiplied with 0. This leads to a different set of active trees in the ensemble for every training run and reduces the risk of overfitting the data. There is a third type of booster that can be used, the Gradient Boosting Linear Model (GBLinear) booster. This booster uses generalized linear regression, and additionally, it uses L1 and L2 shrinkage for to avoid overfitting the data [Pyk21].

Additionally to the capability of the XGBoost to create useful predictions in ML problems, it can be used for feature analysis. The resulting decision trees of the model can be extracted and a list of the most important features of the dataset can be created. The feature analysis is measured with the F-score which is a measure that is based on the analysis of precision and recall [Woo19].

2.5.3 Multilayer perceptron

The simplest ANN is the Multilayer Perceptron (MLP); it simulates calculations over multiple layers of artificial neurons in a straightforward manner. The MLP is a prediction model that tries to solve this problem by adapting layers of internal parameters over the course of training using examples, i.e. supervised learning. The training examples are provided in the form of a dataset containing samples, which in turn consist of a feature vector, that is fed into the network. Then, the correct target vector is tried

[14] https://github.com/dmlc/xgboost

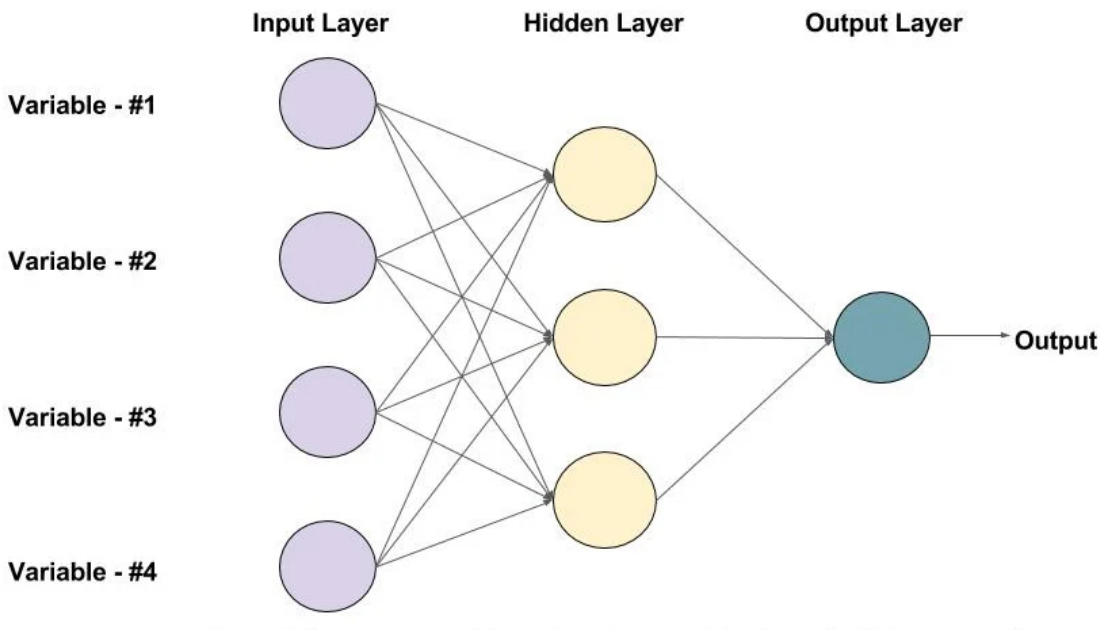


Figure 2.4: Exemplified schematic of an MLP with one hidden layer that is containing three neurons [Gup17]

to be predicted by the network, when given the respective input feature vector.

The schematic view in figure 2.4 shows the multiple layers that comprise the MLP. The layers contain units of artificial neurons that resemble biological neurons. In an MLP every neuron of a layer is connected to every neuron of the previous layer, i.e. a fully connected neural network. Each neuronal connection has a distinct weight that is held in a weight matrix w, and it has a bias vector b. Thus, w and b represent the trainable parameters of a layer. From a vector x, which represents the output of the previous layer (or the input feature-vector of the MLP, if there is no previous layer), the activation $wx + b$ can be calculated. Finally, an activation function f is applied to the activation on every neuron it contains. A resulting vector $y = f(wx + b)$ can be calculated with a vector size according to the amount of artificial neurons of that layer. Generally speaking, each layer can be understood as a representation that impacts the representation of the subsequent layers, together with the parameters between these layers.

Hence, this forms an arbitrary complex computational graph or a pipeline that is used to transport information from the input to the output, for making predictions, and from the output to the input, for the model parameter training. For example, a prediction is made by calculating the activations of the subsequent neurons layer by layer, starting with the input layer and finishing with the output layer of the MLP. The input vector of the MLP contains the values of the features the network perceives and the output vector contains the values of features it predicts.

The training of such an algorithm is termed deep learning because of the deeply

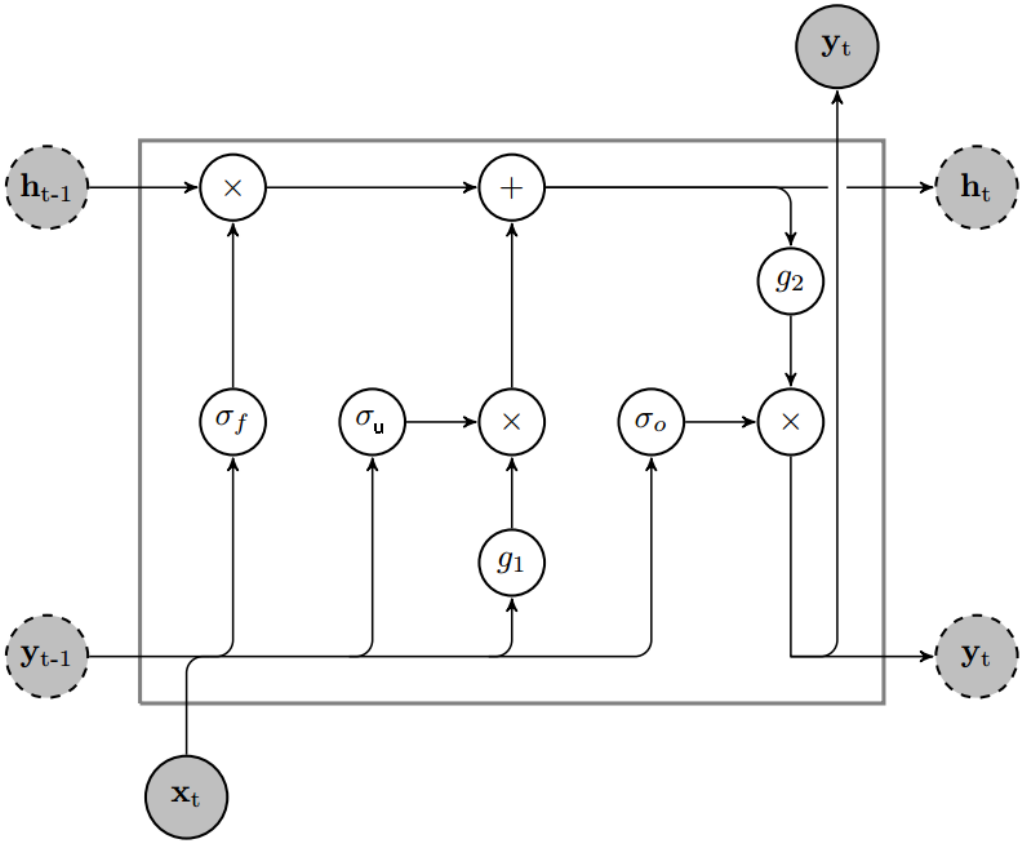


Figure 2.5: Illustration of a LSTM cell [BMK+17]: x represents input and y output; dark gray circles with a dashed border represent the internal state variables, i.e. the information which is passed along the cells; g_1 and g_2 are non-linear transformations; $+$ and $\times$ represent linear operations while σ_f, σ_u and σ_o are the sigmoid functions used for the forget, update and output gates, respectively

nested sequential instructions that have to be executed to evaluate the architecture [GBC16]. In addition to many possible parameters, a seemingly simple MLP comes with a plethora of possibly changeable hyperparameters. There are different activation functions to choose from, and the whole architecture can be modified, including the amount of layers, the form of the input and output vectors and the size of the layers. Furthermore, this list can be extended to include the preprocessing and training hyperparameters.

2.5.4 Long Short-Term Memory network

A Long Short-Term Memory (LSTM) model was first introduced in 1997 [HS97]. An LSTM cell propagates two states to a subsequent cell: its own output y_t and the hidden state h_t of the LSTM, which is also called cell state [BMK+17]. The hidden state h_t is

controlled by three gates: The forget gate, the update gate and the output gate. All of those gates contain trainable sigmoid activation functions $\sigma(x)$, which is defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{2.8}$$

This function $\sigma(z)$ inside the gates serves as a mask with values in the range of 0 and 1 that are element wise multiplied onto another matrix. This way it serves as an information filter. In total there are three filters, one for each gate. In detail, the calculation of every filter is similar: the current input x_t is connected through the weight matrix W, while the previous cells output y_{t-1} is connected through the weight matrix R. The two inputs are added up with a bias b resulting in the following equations, where f, u and o denote the components of the respective filters:

$$\text{forget filter: } \sigma_{f,t} = \sigma(W_f x_t + R_f y_{t-1} + b_f) \tag{2.9}$$

$$\text{update filter: } \sigma_{u,t} = \sigma(W_u x_t + R_u y_{t-1} + b_u) \tag{2.10}$$

$$\text{output filter: } \sigma_{o,t} = \sigma(W_o x_t + R_o y_{t-1} + b_o) \tag{2.11}$$

These functions are then applied in an LSTM cell as it is shown in figure 2.5. These LSTM cells can then be used as building blocks for complex LSTM networks.

2.5.5 Model training with gradient descent

In the case of the forecasting application developed in this work the objective is to create a prediction for a future time point with a minimal error according to a specific metric. In the context of training, this metric is then called a loss function L. Since there are different ways of measuring the deviation of a predicted and observed time series, different metrics can be used for the loss L, as described in the previous section. Because there are many trainable parameters for the neurons, as mentioned in section 2.5.3, every single parameter can be adapted by calculating the derivative of that parameter in respect to L and then adapting the parameter in the direction of the negated derivative. This procedure is called stochastic gradient descent. Because after the parameter adaptation L is likely to return a smaller value for a future evaluation of the network.

There are complex algorithms to calculate the exact parameter update — these are called optimizers. An overview of optimizers was made by Sebastian Ruder [Rud17]. In the experiments included in this work the Adaptive Moment Estimation (Adam) optimizer [KB17] has been chosen for several reasons. Firstly, it is the most frequently used for state-of-the-art ANNs [BMK+17]; and secondly, it typically requires relatively little tuning [KB17]. Having reliable hyperparameter choices helps to reduce the complexity of the experiments in section 4.

Due to the fact that the main goal of Machine Learning is to calculate accurate

predictions for unseen data, optimizing a network to calculate the right outputs to the respective features in the data that the model is trained on is not enough. The main goal is to make the network learn the underlying patterns in the data, i.e. to be able to generalize from it and eventually predict unseen data correctly. There are two main methods to support the learning capability of ANNs. The first method is to split the data into distinct sets, so that the model can be evaluated on a subset of the data it was not trained on. This is more broadly discussed in section 3.5. The second method to improve learning is the use of so-called regularization techniques. Regularization generally reduces the vulnerability of the model to overfit, i.e. the overly strong adaptation to the training data.

For regularization dropout will be used, which is a computationally inexpensive, yet effective, method for regularization. Furthermore, dropout is controlled by a dropout rate parameter that steers how many units of a neural network layer are multiplied with zero. In this way it temporarily deactivates the units effect during training time. This leads to a different set of active neurons in the network for every training run and reduces the risk of overfitting the data.

3 Methodology

After covering the theoretical foundations in the previous chapter, this chapter will focus on the actual methodologies and implementations used in this work. Doing that first the data that was used is shown and analyzed in section 3.1. Then, in section 3.2 the exact procedure how the data was further preprocessed using the indicators of the technical analysis is shown. After that, the preprocessing applied to the data is explained in section 3.3. Subsequently, the prediction models that were used in this work are explained in section 3.4. Then, the model evaluation methodology is explained in section 3.5. And finally, the programming tools that were used to implement the experiments are listed in section 3.6.

3.1 Datasets

This work tries to measure the predictive capabilities of the prediction models on financial and non-financial datasets. In this work the term non-financial time series refers to not being a series of a financial asset price as defined in chapter 2.2. Therefore, means non-financial time series can still contain information like amount of sales of a commodity. Having two different types of time-series, the performance of the models predicting them can be compared. The financial datasets are introduced first. For every kind of the three financial markets discussed in section 2.1, a representative financial asset was chosen. Each financial dataset consisting of OHLCV time series. Subsequently, the non-financial datasets used are introduced. These consist of time series with a single feature value. Additionally, to analyze seasonality properties, an additive seasonal decomposition is applied to each time series [RWI90].

3.1.1 MSCI World price

As a representative time series for the stock market the MSCI World index was used. This index is one of the major global economy indices [Kue21], and it contains 1557 companies across 23 developed countries [Fer21a].

The time series was retrieved from Kaggle [Mar21], and it contains daily MSCI World OHLCV data from January 2012 until November 2017. As it is displayed in figure 3.1 the structure of the curve in linear scale and log scale look relatively similar. Additionally, there is a positive trend in the recent years as it is shown by its seasonal decomposition that is displayed in figure 3.2.

3.1.2 Euro price

Furthermore, for the forex market the time series of the euro price measured in USD was used. This currency pair was chosen because it is the most traded currency pair

Figure 3.1: MSCI World price in USD; linear scale on top and log scale on bottom; the D denotes when dividends were paid to the investors

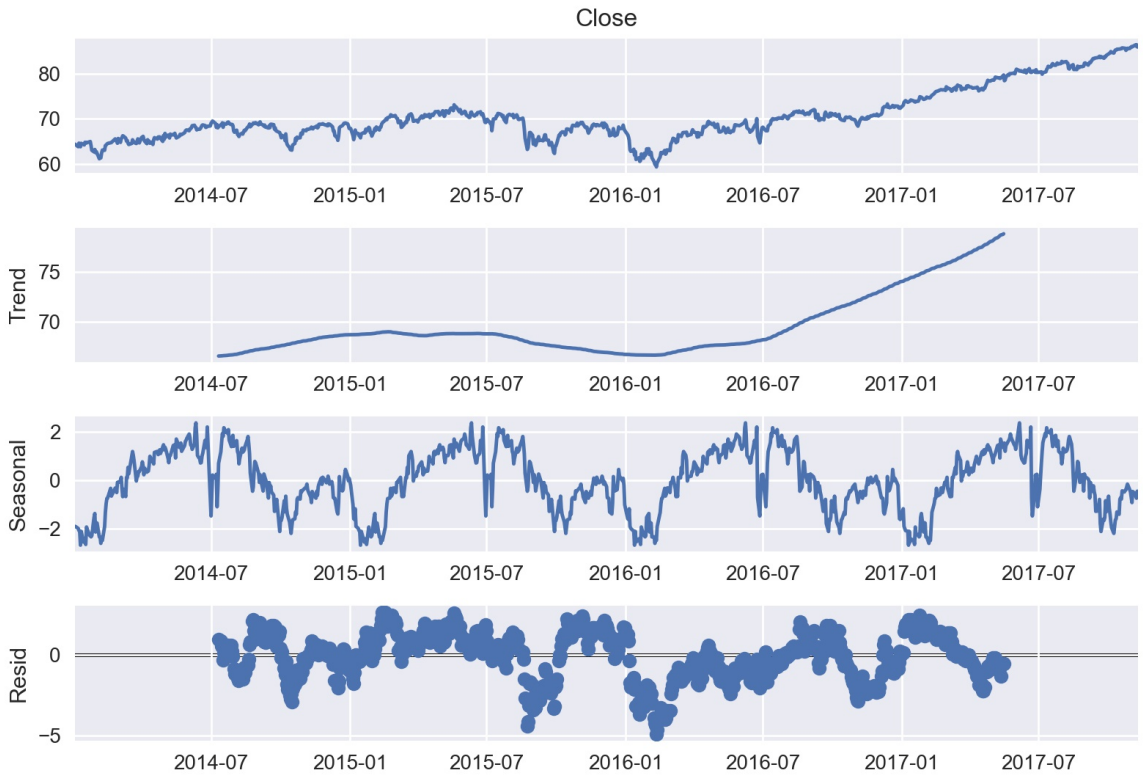


Figure 3.2: Seasonal decomposition of the MSCI World price in USD with residual points plotted on bottom

Figure 3.3: Euro price in USD; linear scale on top and log scale on bottom

in the world [IG21]. The time series of the price was tracked from 2010 to 2016, and it is displayed in figure 3.3. It was also retrieved from Kaggle [Jan21].

Just like the series of the MSCI World price, the linear and log scaled series look similar, see figure 3.3. The OHLCV was measured every 15min. In comparison to the other datasets, this is by far the largest time series with over 93000 time points, and therefore it took the most computing effort to be processed by the prediction models. Additionally, the seasonal decomposition of figure 3.4 shows a downward trend in 2014 and 2015.

3.1.3 Bitcoin price

The third financial time series used in this work is the price of Bitcoin. Bitcoin has always had the largest market capitalization of any cryptocurrency with more than a double of the market capitalization of the second biggest cryptocurrency, Ethereum (measured September 2021) [Che21b]. Therefore, the Bitcoin price is representative for the whole cryptocurrency market. Generally, when the Bitcoin price falls the price of all cryptocurrencies in the market falls as well.

The Bitcoin price was tracked at a daily frequency from its first day being publicly traded, in August 2011, and the time series ends in July 2020. This dataset is also retrieved from Kaggle [Raj21].

The series behaves quite differently to the MSCI World and euro prices. When

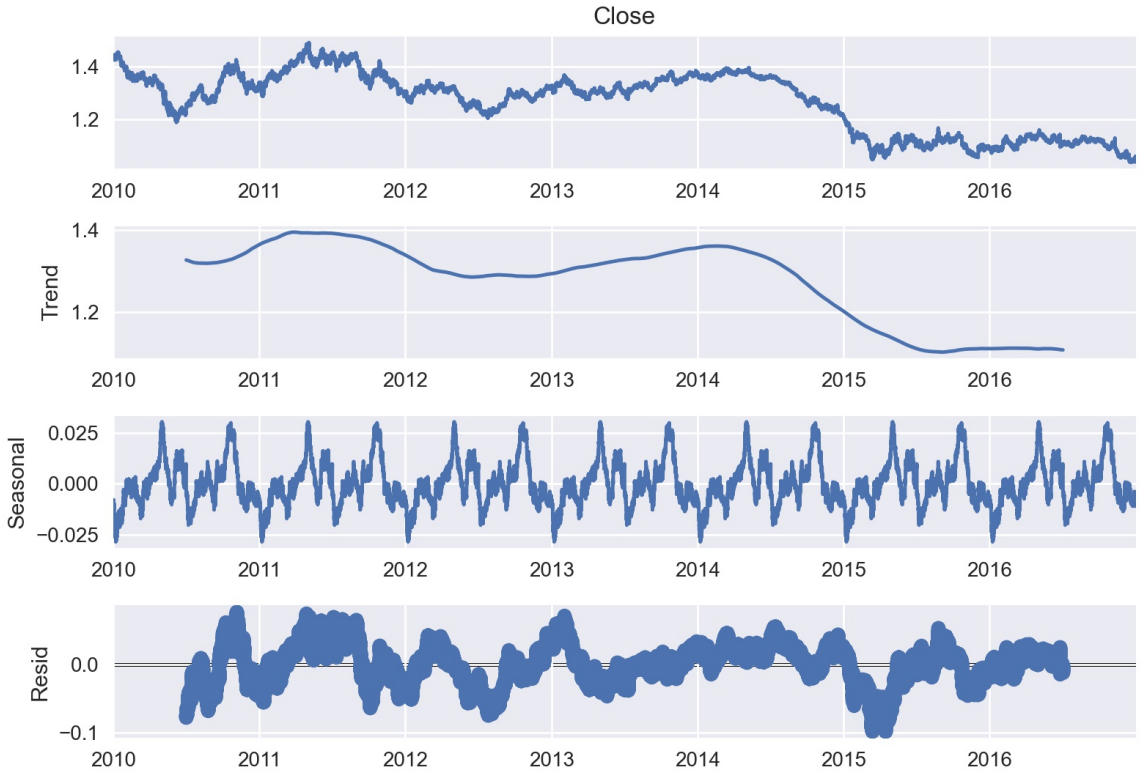


Figure 3.4: Seasonal decomposition of the euro price in USD with residual points plotted on bottom

Figure 3.5: Bitcoin price in USD; linear scale on top and log scale on bottom

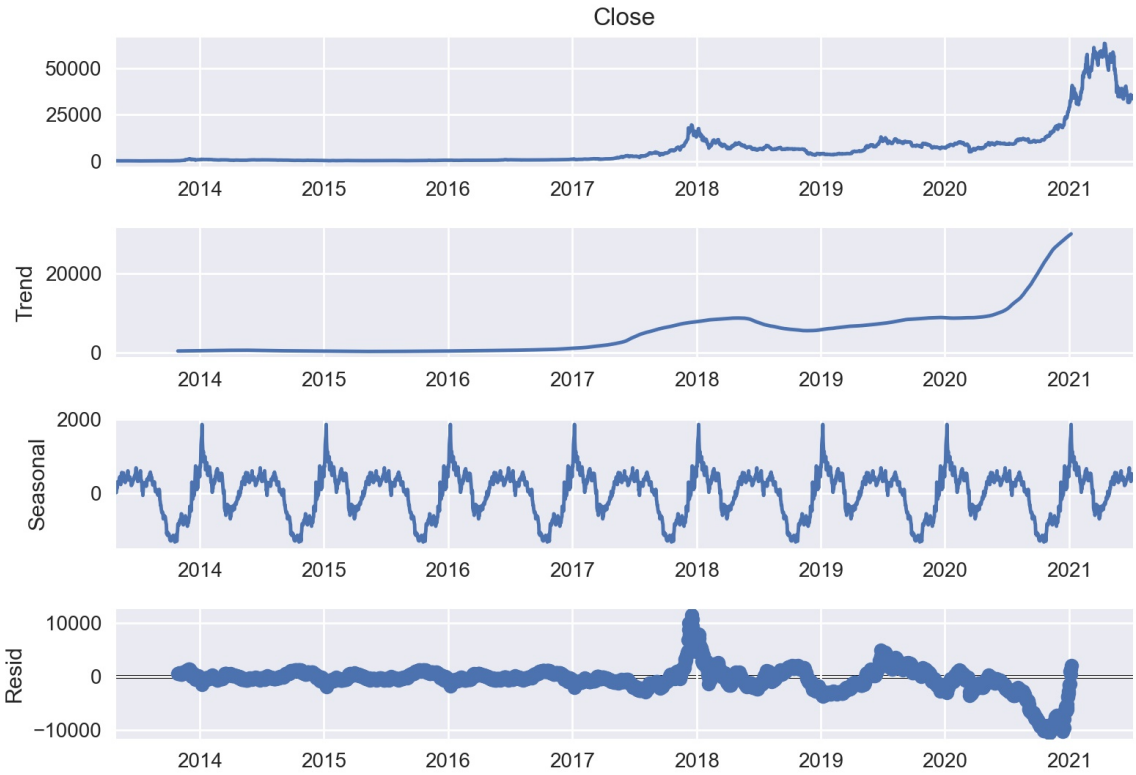


Figure 3.6: Seasonal decomposition of the Bitcoin price with residual points plotted on bottom

comparing the linear scaled curve to the log scaled curve in figure 3.5 it can be seen that the curves of the two scales look quite distinct. The Bitcoin growth rate resembles an exponential curve [Mut21] as it is also shown in the trend line of figure 3.6.

Additionally, the Bitcoin dataset used also contained information about the market capitalization for each sample. Therefore, this data was added additionally to the OHLCV information.

3.1.4 Monotonic rising line

To analyze the models' performance on the most simple dataset possible a monotonic rising line was chosen, see figure 3.7. The line dataset contains around 2500 data points; this is the same amount as the Bitcoin price datasets. The data was generated using this function:

$$f(t) = t \tag{3.1}$$

3.1.5 Shampoo sales

The shampoo sales dataset is a standard dataset in machine learning and time series analysis [WMH98]. It contains only 36 samples of monthly shampoo sales, representing

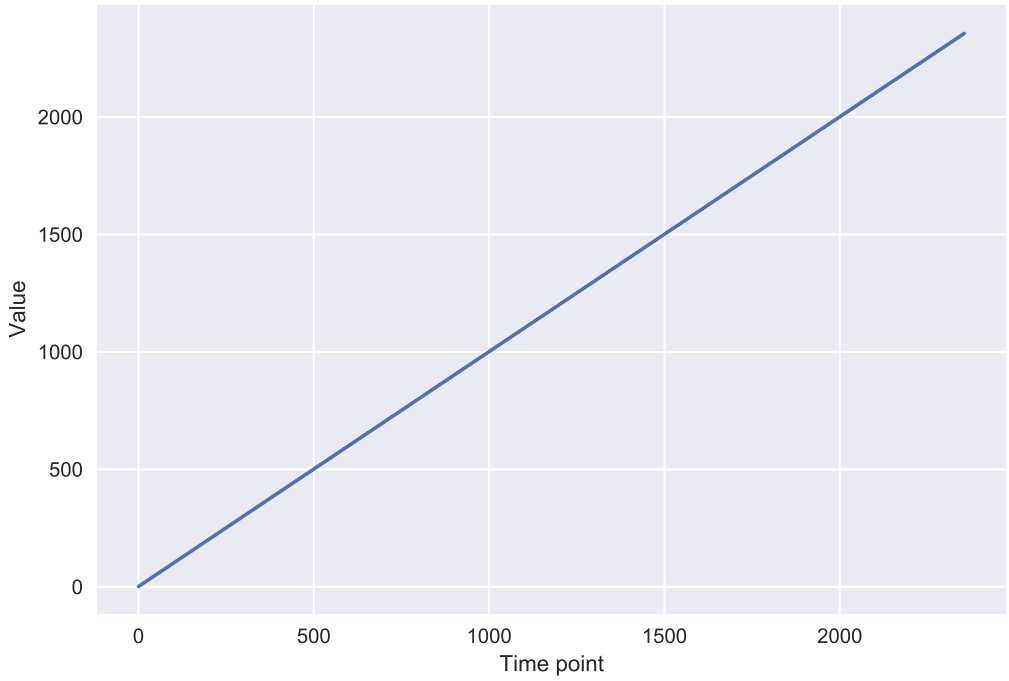


Figure 3.7: Monotonic rising line

its development over three years. Therefore, it is the smallest dataset used in this work. In figure 3.8 can be seen that the data contains a slightly increasing trend.

3.1.6 Alcoholic beverage sales

A second commodity sales dataset was added, because of the small sample size of the shampoo dataset. This third non-financial dataset refers to the sales count of beer, wine and distilled alcoholic beverages in the United States of America between 1992 and 2016 [U.S21].

The seasonal decomposition can be viewed in figure 3.9. There a trend comparable to the shampoo sales in figure 3.8 can be seen. Additionally, this dataset is the first dataset to show a clear seasonality characteristics that have a larger impact than the residual error.

3.2 Application of indicators

The financial time series were preprocessed with technical indicators to make the raw OHLCV price action more interpretable. Because there is a plethora of technical indicators to choose from, only a selection of the most promising indicators where used in this work.

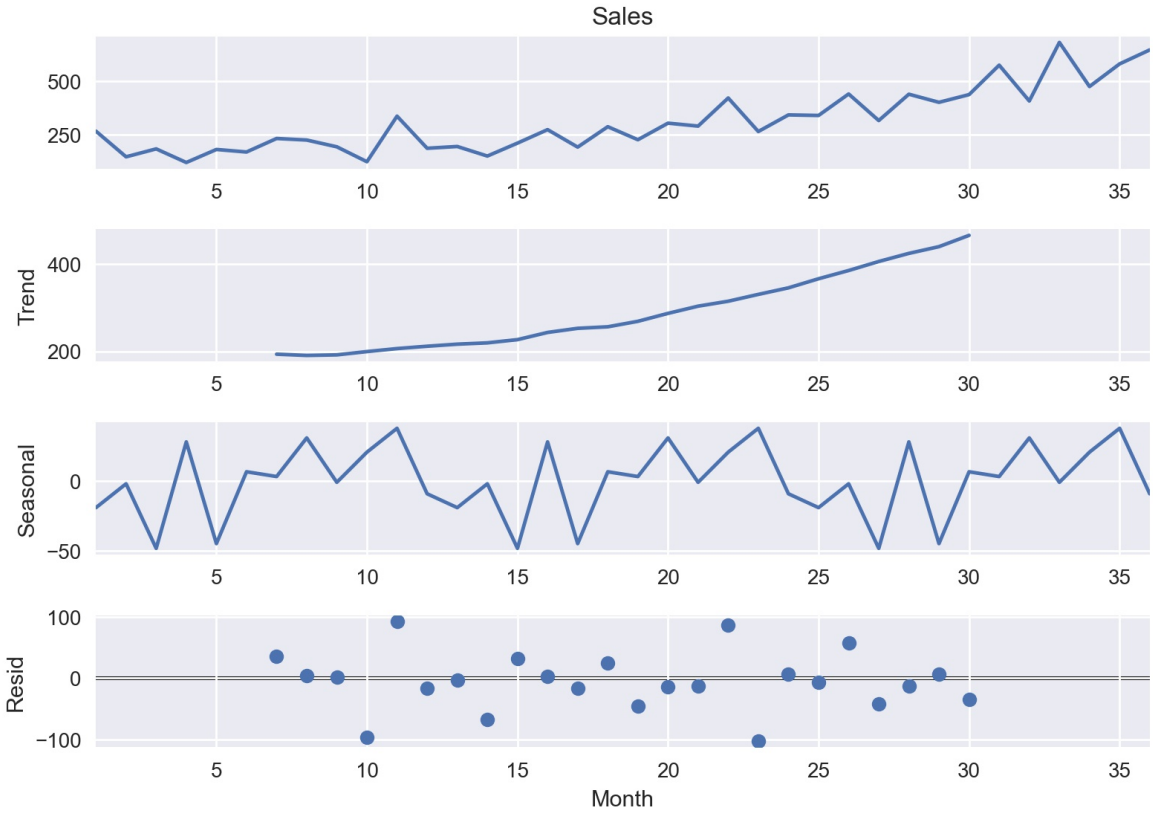


Figure 3.8: Shampoo sales over a three-year period with residual points plotted on bottom

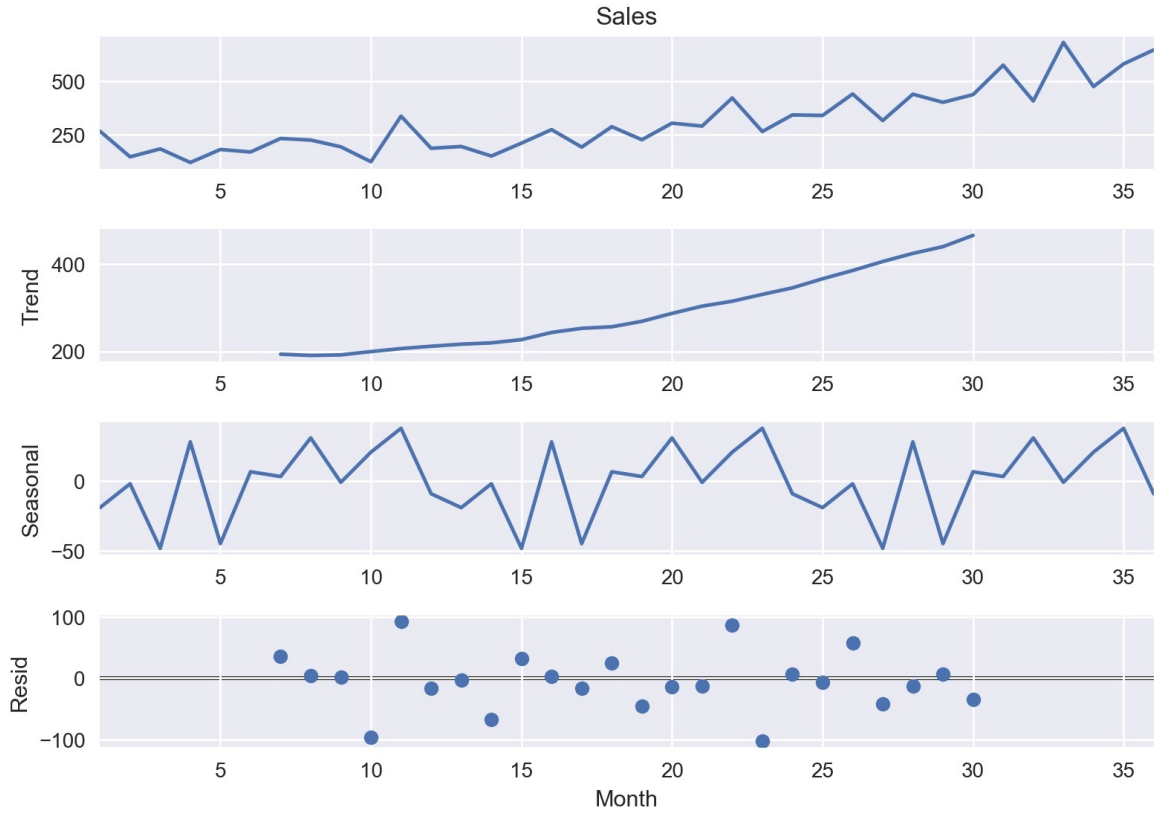


Figure 3.9: Beverage sales with residual points plotted on bottom

Indicators can be classified according to their capability of describing distinct features of a financial time series. These features are volume, volatility, trend and momentum [Pad21]. To have a diverse selection of indicators describing distinct features, a specialized indicator for each area was chosen. Each of the five indicators used in this work will be explained shortly in this section.

3.2.1 Volume Weighted Average Price

To describe the volume of a time series, the Volume Weighted Average Price (VWAP) overlay indicator is often used by algorithmic trading strategies [Mit21b]. The VWAP describes the average price weighted by volume. VWAP is calculated by adding up the total amount of traded security prices in a time frame T. In detail, VWAP is calculated by multiplying the price with the number of shares traded, then the result is divided by the total shares traded in the time frame T [Fer21c]:

$$VWAP_t = \frac{\sum_{t=0}^{T} Price_t * Volume_t}{\sum_{t=0}^{T} Volume_t} \tag{3.2}$$

VWAP serves as a reference point for prices for one day. As such, it is best suited for intraday analysis. Chart analysts can compare current prices with the VWAP values to determine the intraday trend. VWAP can also be used to determine relative values.

Figure 3.10: VWAP example [Tra21]

Prices below VWAP values are relatively low for T. By contrast, prices above VWAP values are relatively high for T [And21c]. An example of the VWAP is shown in figure 3.10.

3.2.2 Average True Range

To describe the volatility of the time series in this work, the Average True Range (ATR) indicator was used. This oscillator is used to develop a complete trading system, as well as be used for entry or exit signals as part of another strategy [Hal21]. ATR is based on the True Range (TR), and it is calculated with this formula [And21a]:

$$\begin{aligned} TR_t &= High_t - Low_t \\ ATR_t &= \frac{ATR_{t-1} * 13 + TR_t}{14} \end{aligned} \tag{3.3}$$

An example of the ATR is shown in figure 3.11.

3.2.3 Aroon

Describing the trend of a time series well is the most important task in time series prediction. Therefore, the data was processed with two of the most important trend indicators, the MACD and the Aroon oscillators [Pot21]. Aroon consists of $Aroon_{up}$ and $Aroon_{down}$ which measure the number of time steps since the last 25-period high

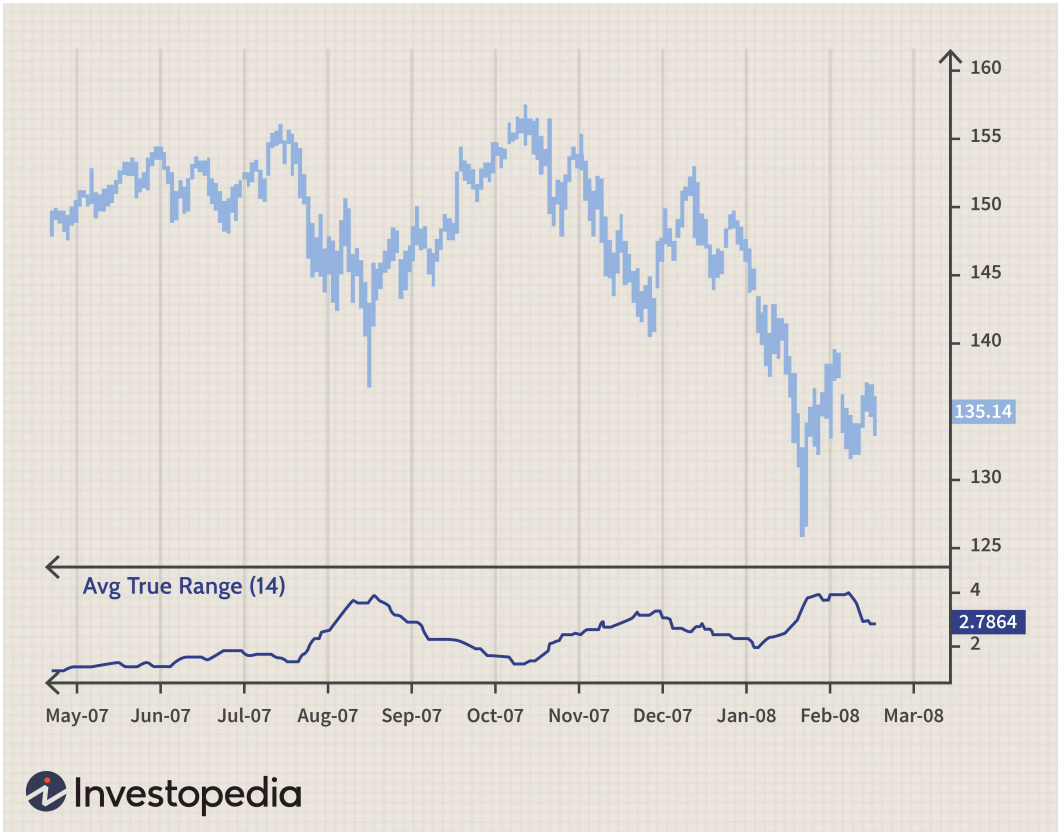


Figure 3.11: ATR example [Hal21]

and low. The calculation is made according to this formula [Mit21c]:

$$\begin{aligned} Aroon_{up} &= 100 * \frac{25 - max(High_{t-25..t})}{25} \\ Aroon_{down} &= 100 * \frac{25 - max(Low_{t-25..t})}{25} \\ Aroon &= Aroon_{up} - Aroon_{down} \end{aligned} \tag{3.4}$$

To identify how trends change in a financial time series the Aroon indicator is used. In detail, this indicator measures the time the last maximum and the last minimum in a time period $t = 25$. The concept behind the Aroon is that strong up-trends will regularly see new maxima, and strong down-trends will regularly see new minima. The indicator signals how regularly this is happening [Mit21a]. An example of this is shown in figure 3.12.

3.2.4 Moving Average Convergence Divergence

The MACD oscillator is calculated using 3 Exponential Moving Averages (EMAs). In detail, it consists of the MACD line and the signal line. The MACD line is calculated by subtracting a 26-period EMA from a 12-period EMA and the signal line is an EMA of the previous 9 periods of MACD. The MACD indicator then is calculated by

Figure 3.12: Aroon example graph with the yellow line representing $Aroon_{up}$ and the yellow line representing $Aroon_{down}$; resulting in the red line of the Aroon indicator

subtracting the signal line by the MACD signal, as it is shown in equation 3.5 [Fer21b].

The EMA is an exponentially weighted moving average and for a period N, and it is calculated based on a weighted multiplier α. The total calculation for MACD is reflected in these equations [Mav21][Vip21]:

$$\begin{aligned} \alpha_N &= \frac{2}{N+1} \\ EMA_{N,t}(Price) &= \alpha_N * Price_t + (1-\alpha_N) * EMA_{N,t-1} \\ MACD\_Signal_t &= EMA_{12,t}(Price) - EMA_{26,t}(Price) \\ Signal\_Line_t &= EMA_{9,t}(MACD\_Signal) \\ MACD\_Indicator_t &= MACD\_Signal_t - Signal\_Line_t \end{aligned} \tag{3.5}$$

When the MACD is positive it describes a bullish market. When the value is negative, it describes a bearish market. An example of a MACD curve is shown in figure 3.13.

3.2.5 Relative Strength Index

The last indicator that was used in this work is the RSI, a momentum indicator, oscillator, which is also one of the most important technical indicators [Pot21]. The RSI is based on the average gain and average loss over the last 14 time-points according

Figure 3.13: MACD example where the dark-blue line represents the EMA_{26} and the orange line represents the EMA_{12}; from those the light-blue MACD line is created and the columns on the bottom represent the MACD indicator [Fer21b]

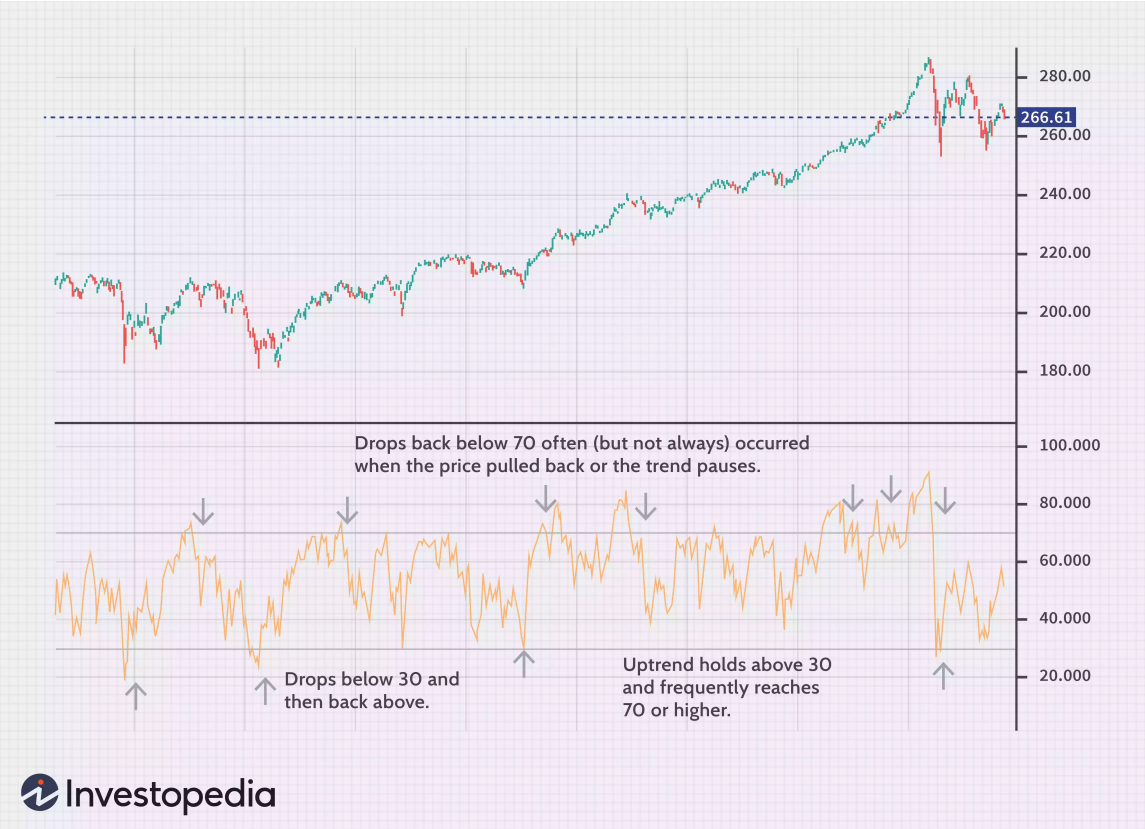


Figure 3.14: RSI example including some analysis when the indicator moves above 70 or below 30 [Pot21]

to these formulas [And21b]:

$$
\begin{aligned}
Gain_t &= Close_t - Open_t \\
Loss_t &= Open_t - Close_t \\
Average\_Gain_t &= \frac{Average\_Gain_{t-1} * 13 + Gain_t}{14} \\
Average\_Loss_t &= \frac{Average\_Loss_{t-1} * 13 + Loss_t}{14} \\
RS_t &= \frac{Average\_Gain_t}{Average\_Loss_t} \\
RSI_t &= 100 - \frac{100}{1 + RS_t}
\end{aligned}
\tag{3.6}
$$

Therefore, the indicator moves between 0 and 100 and describes recent price gains versus recent price losses of a security. In total, it helps to gauge momentum and trend strengths. In many trading strategies an RSI moving above a level of 70 or below a level of 30 is used for trading decisions [Pot21], as it is exemplified in figure 3.14.

3.3 Preprocessing

Because the raw data format needed further processing to be read successfully by the predictive models, certain amounts of preprocessing was done. In this section the preprocessing steps are explained. First, the scaling techniques applied to the data are shown in section 3.3.1. Secondly, the formatting of the time dimension of the data is explained in section 3.3.2.

3.3.1 Scaling

Because ML models such as neural networks are sensitive to the magnitude of the numbers they are processing it is recommended to apply feature scaling. Otherwise, it is possible for large inputs to slow down or prevent the learning and convergence of your network. The solution for this problem is scaling, which changes the absolute values but keeps the relative distances of a time series. Therefore, there is no data loss when it is applied and the data can be unscaled later to fully recreated the original data format.

There are two options how to scale a series that should be considered: normalization and standardization. Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1. This is useful for many ML tasks, because a fixed range of possible values is easier to process. But in financial time series prediction there are no fixed boundaries because unknown future values could yield a new absolute maximum or minimum. Such future extrema could never be predicted using normalization because they are outside the fixed range.

Therefore, standardization is a more flexible option for financial time series prediction. Standardization is calculated for each sample x_t from all the observed values $x_0..x_t$. It then transforms the distribution of values so that the mean $\mu_{0..t}$ is 0 and the standard deviation $\sigma_{0..t}$ is 1 [Bro17]. Two standardization techniques have been applied in this work that will be explained in this section: standard scaling and robust scaling.

Standard scaled values z_{standard} can be calculated for each sample x_t using this formula:

$$z_{\text{standard}}(t) = \frac{x_t - \mu_{0..t}}{\sigma_{0..t}} \tag{3.7}$$

But when large outliers are present the standard scaler cannot guarantee to scale the data into a balanced range around 0 [PVG+11].

Therefore, there is another approach to standardizing a time series in the presence of large outliers. This technique is called robust scaling and in its calculation the outliers are ignored. This can be achieved by using the median $\bar{x}_{0..t}$, the 25th percentile $x_{25}(x_{0..t})$ and 75th percentile $x_{75}(x_{0..t})$ of $x_{0..t}$. To calculate the scaled values $\bar{x}_{0..t}$ is subtracted from each value and then the result is divided by the interquartile range $x_{75}(x_{0,..t}) - x_{25}(x_{0..t})$. [Bro20] All in all, the robust scaled values z_{standard} are calculated

$$\begin{bmatrix} f_{1,t-1} & f_{2,t-1} & \cdots & f_{N,t-1} \\ f_{1,t-2} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ f_{1,t-T} & \cdots & \cdots & f_{N,t-T} \end{bmatrix}$$

Table 3.1: LSTM input sample

$$\begin{bmatrix} f_{1,t-1} & f_{2,t-1} & \cdots & f_{N,t-1} & f_{1,t-2} & \cdots & f_{N,t-T} \end{bmatrix}$$

Table 3.2: One-dimensional input sample for the Linear Regressor, XGBoost and MLP

with this formula:

$$z_{\text{robust}}(t) = \frac{x - \bar{x}_{0..t}}{x_{75}(x_{0..t}) - x_{25}(x_{0..t})} \tag{3.8}$$

Therefore, the robust scaler scales the data into a larger range that is more equally distributed around 0 compared to the standard scaler [PVG+11].

On of the scaling techniques had to be applied to the financial datasets, because the large values for the volume or market capitalization of the securities could break the ML models. Thus, both of the scaling techniques were applied to compare which one is better suited for financial time series analysis. This was done for the financial datasets only because on the non-financial datasets the scaling techniques did not need to be applied to create predictions.

3.3.2 Processing time dimension

In contrast to other areas of ML, time series analysis requires additional preprocessing of the time dimension before the predictive models can read a time series. The two different ways of processing the time dimension that were applied in this work are explained in this section.

The most straightforward way of processing the time dimension is to create a matrix for each time step t of a series. The first dimension of the matrix then corresponds to the feature dimension of the N features, f_1 to f_N, of the time series. Additionally, there is a second dimension of the matrix that represents the time dimension when using a window of T time steps from t to $t - T$. So for each feature f_n, then there are T values $f_{n,t-1}, f_{n,t-2}, ..., f_{n,t-T}$. This $F \times T$ matrix is illustrated in table 3.1. The LSTM models is specialized in processing data of multiple time steps.

The other ML models that were used in this work are not capable to process two-dimensional data but only one-dimensional data, therefore the time dimension of the data needed to be encoded in the feature dimension as well. Therefore, the rows of the $F \times T$ matrix displayed in table 3.1 were flattened to a single dimension. This creates a one-dimensional input vector as it can be seen in table 3.2.

3.4 Implemented prediction models

3.4.1 Prediction baselines

To create reliable prediction models, first strict baselines had to be defined. This is a good practice in ML in general to analyze if the ML models are able to create valuable predictions or not. Therefore, three naive predictors were chosen as baselines in this work.

The first baseline is the previous value. Therefore, it takes the previous time point's value $Close_{t-1}$ and uses this value as a prediction for the current day. Thus, the error of this baseline for a time series of length T can be described with this formula:

$$Baseline_{\text{prev}} = \frac{\sum_{t=0}^{T} MAPE(Close_{t-1}, Close_t)}{T} \tag{3.9}$$

The second baseline predictor takes all the previously observed values and calculates their mean value. This mean is then used to predict the current day, as it is summarized in this formula:

$$Baseline_{\text{mean}} = \frac{\sum_{t=0}^{T} MAPE(Mean_{0..t-1}, Close_t)}{T} \tag{3.10}$$

The third baseline predictor is applied just as the mean baseline but instead of the mean the median of the time series was calculated. Therefore, this formula was applied:

$$Baseline_{\text{median}} = \frac{\sum_{t=0}^{T} MAPE(Median_{0..t-1}, Close_t)}{T} \tag{3.11}$$

3.4.2 Linear Regressor

The most simple machine learning model used in this work is the linear regressor, as it was introduced in section 2.5.1. For that, the default model of ordinary least squares regression of scikit-learn was used in this work[15].

3.4.3 XGBoost Regressor

In this work, three different configurations of the XGBoost model, according to the three boosters introduced in section 2.5.2, were tested. The default GBTree booster, the Dart booster with a dropout rate of 10% and the GBLinear booster were tested. The number of trees in the ensemble has been set to 1000, as this represents a default value for time series problems [Bro21][Mul21]. Further parameters were set to the default parameters of the XGBoost library [CG16].

Furthermore, the model was trained with early stopping. For that the error on the validation set was observed after every training epoch, i.e. after a training on every

[15]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

sample in the training set. When the validation error then did not decrease for 50 rounds, the training was ended and the best performing model on the validation set was selected as resulting model of the training session.

3.4.4 Multilayer perceptron

The MLPs that were used in this work were designed in a modular fashion. The input to the MLP was transmitted into a variable amount of densely connected feed-forward neural network blocks. This layer contained a variable amount of neurons with a Rectified Linear Unit (ReLU) activation function, which has the following formula [Bah21]:

$$RELU(x) = max(0, x) \tag{3.12}$$

To the output of the densely connected layers a dropout rate of 15% was applied. Furthermore, an optional batch-normalization step could be applied, i.e. the batch would be normalized before being passed to the next layer.

After these densely connected blocks, the output was mapped to a single output neuron to create the prediction value. In this step a linear activation function was used. All of these parameters were chosen because they are a common choice for neural network model configurations [Dab21].

The weights of the layers were initialized according to a Glorot uniform initializer [GB10]. Which comprises the drawing of samples from a uniform distribution within a range $[-l, l]$, with the limit l calculated as

$$l = \sqrt{\frac{6}{n_i + n_o}}, \tag{3.13}$$

and with n_i being the number of input units of the weight matrix and n_o being the number of output units of this matrix.

The MSE was used as the metric to calculate the loss for the gradient-descent of the ANNs in this work. Furthermore, the Adam optimizer has been used with the default hyperparameter setup of Keras[16]. As being used for the XGBoost training, explained in section 3.4.3, early stopping has been used for the MLP model as well. Because the training of neural networks generally takes more time than the training of boosting tree models, instead of 50 early stopping rounds, a decreased number of 10 early stopping rounds were made. Similar to the XGBoost, the best performing model on the validation set was then chosen as the training result.

[16] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

3.4.5 Long Short-Term Memory network

The LSTM models were created analogously to the MLP models described in the previous section. The only difference was, that of instead of having blocks with densely connected layers, the LSTM models had blocks with LSTM layers.

3.5 Evaluation methodology

Now that the implemented models have been covered, this section will focus on the methodology used to evaluate and compare the models. In the following two subsections, the creation of the test set, in subsection 3.5.1, and the training and validation set, in subsection 3.5.2, will be described.

3.5.1 Test set

Before starting to apply the models, the dataset was split into two parts: the first part comprises the first 80% of the data and the second part the most recent 10% of the data.

The first part is used for training and validation. Training is the process of fitting the parameters according to gradient descent explained in section 2.5.5. In contrast, validation is an evaluation phase after a certain number of training steps, to track the training performance during training time. The evaluation frequency in Keras is set to one epoch, i.e. the process of training once on every sample of the training subset.

And the second part is taken aside and will be used as a test set after selecting the most promising models and parameter configurations. This test set will only be used once in the final evaluation of this work. Thus, it is ensured that the final performance evaluation is as close as possible to the performance when using the model as a real-world prediction service, predicting data points it has never seen before.

3.5.2 Training and validation set

In general, there are different ways of splitting the larger part of the dataset into a training and a validation set. A common way is to give the validation set the same size as the test set. In this case, one would reserve another 10% for the validation set resulting in an 80-10-10 split. That means the model is trained for 80% of the total time steps and after that it is validated on the following 10% of the data. This is how the training and validation sets for the experiments were split.

3.6 Programming tools

In this section, the software versions of the programming tools used in this work are listed. This is important for reproducibility because the tools might have influenced

	Laptop	Local server	Kaggle notebooks
Operating system	macOS 11.4	Ubuntu 20.04	Ubuntu 18.04
Python	3.9.6	3.9.6	3.7.10
Keras[25]	2.4.3	2.5.0.dev2021032900	2.4.3
pandas[26]	1.1.5	1.3.0	1.2.4
scikit-learn[27]	0.24.2	0.24.2	0.23.2
TensorFlow[28]	2.5.0	2.5.0 (built from source)	2.4.1

Table 3.3: Software versions

the results of this work in unexpected ways. Additionally, the performance and results of the tools might change when using other versions.

The experiments were deducted on three different machines which will be listed in this section. The first machine is a development laptop running macOS[17]. It was used to develop the experiments and to deploy small calculations. The second machine is an Ubuntu[18] server that was used to deploy longer calculations that were accelerated by GPU usage. The third type of machine used is the cloud offering of Kaggle. This cloud service was used to access high-performance GPUs and Tensor Processing Units[19].

All machines used the same code built in Python 3. This is the most recent version of Python[20], a dynamically typed and interpreted programming language, which is the most popular programming languages used for ML solutions [SPE21]. Additionally, all machines used the Python libraries NumPy[21] version 1.19.5, the technical analysis library ta[22] version 0.7.0, XGBoost[23] version 1.4.2, and Graphviz[24] version 0.17. In the cases where multiple software versions were used by the different machines, the versions are listed in table 3.3.

[17]https://www.apple.com/macos/big-sur
[18]https://ubuntu.com
[19]https://www.kaggle.com/docs/tpu
[20]https://www.python.org
[21]https://numpy.org
[22]https://github.com/bukosabino/ta
[23]https://xgboost.readthedocs.io/en/latest
[24]https://graphviz.org
[25]https://keras.io
[26]https://pandas.pydata.org
[27]https://scikit-learn.org/stable
[28]https://www.tensorflow.org

4 Experiments

In this chapter the experiments conducted in this work are described. The approach taken is based on a theoretical framework that describes four central steps of deploying ML models [GBC16]:

1. An error metric and a goal value you want to achieve with your models have to be determined.

2. A working model has to be established as soon as possible.

3. The properties and the performance of the model were analyzed with according to the error metric. The property of the model to generalize for new data, as it should, have to be modeled, and defects in your data or algorithms have to be detected.

4. The last step is the parameter search. Repeatedly incremental changes were made, such as using multiple kinds of data, adjusting the data formats and adjusting hyperparameters, based on the specific findings from the experiments.

5. At last, after doing all the experimentation, an independent test score on the test set was calculated.

Using this approach, it was decided to choose the MAPE as the main metric. After that, the first step was the analysis of the performance of the benchmark models. Thus, a target value for the research models could be estimated. This was done in the first subsection, where a first analysis of the models was done. Additionally, it was then checked how the initial configurations of the models performed compared to the baselines.

After the first analysis, several experiments to analyze the performance of the models under different conditions were made. At first, in section 4.2, it was observed how the different data scaling methods of this work affected the model's performance. In section 4.3 it was observed how the performance changed with the application of indicators in the data preprocessing.

After the analysis of the data preprocessing techniques, in the next sections it is described how changing the models affects their performance. Therefore, in the sections 4.4 and 4.5 different sets of hyperparameters for the different models were applied. So it was checked in what way the performance of the models could be improved.

In section 4.6 the amount of time steps that were fed into the models was analyzed as well. And in the final section, section 4.7, the best configurations that could be found in the previous experiments were then evaluated regarding their performance on the test sets. To create more compact tables, abbreviations for the MSCI World (MXWO), Linear Regression (LR) and Previous value (Prev) were used.

	LR	XGB	MLP	LSTM	Prev	Mean	Median
MXWO	386.85	3877.58	489.10	520.84	**325.70**	3968.14	3181.30
Euro	**28.09**	35.86	71.20	399.63	28.10	1908.62	1215.53
Bitcoin	590.23	664.99	822.38	762.03	**534.71**	4270.87	3186.01
Line	**0.00**	518.43	7.19	3.45	4.47	281.51	281.51
Shampoo	2063.15	1625.89	1588.18	**1550.85**	2448.07	2127.36	2378.12
Beverages	**788.43**	1239.43	813.31	924.32	1260.22	924.49	917.13

Table 4.1: First evaluation measured with MAPE; XGB represents the XGBoost model; bold values refer to the top score on a dataset

4.1 First evaluation

To begin the analysis, the models were analyzed with a simple default parameter setup. For the XGBoost the GBTree booster was chosen, because it is the default booster of Python's XGBoost library [CG16]. For the neural networks simply one neural network block, as explained in section 3.4.4, was used. Additionally, a number of 200 neurons was chosen, and no batch normalization was done. This ANN configuration was chosen because it was suggested by Kaggle tutorials [Dab21]. Additionally, the amount of time steps to process for each sample was set to 10 to have a simple default value. The datasets for the first analysis were scaled using the standard scaling procedure and no indicators were applied.

The results of the first experiments can be seen in table 4.1. The displayed values were measured with the MAPE metric, and therefore, the values represent a percentage value how good the models could predict the data, as explained in section 2.4.3.

The first thing that was done analyzing the data was to check how the baseline models performed on the datasets. It can be seen in table 4.1 that the previous value benchmark provided the smallest MAPE for the financial datasets and the monotonic rising line. In comparison, the mean and median provided a better benchmark for the sales time series of shampoo and alcoholic beverages. The difference between the error of the previous value and the other baselines on financial datasets, as well as the line, is very large. Whereas, the difference of results of the baselines for the sales datasets, of shampoo and beverages is comparably small. Because the previous value benchmark seems to be harder to beat and because the focus of this work lies on the financial datasets, the previous value was chosen as the main benchmark for the experiments.

Analyzing the model's performance it can be seen that only the linear regressor for the euro price was able to beat the previous value baseline on financial datasets, but it only beats it at a very small margin of 0.01%. Especially the XGBoost seemed to have difficulties with the MSCI World dataset; its result is comparable to the mean and median baseline results on this dataset.

In contrast, the models generally were able to predict the line as well as the sales datasets successfully. All four models could beat the baselines on the shampoo dataset

	LR	XGBoost	MLP	LSTM
MXWO$_s$	118.77	1190.54	150.17	159.91
MXWO$_r$	118.52	1119.19	213.55	153.67
Euro$_s$	**99.96**	127.62	253.38	1422.17
Euro$_r$	**99.96**	124.23	435.35	2016.96
Bitcoin$_s$	110.38	124.36	153.80	142.51
Bitcoin$_r$	149.03	115.34	885.42	188.58

Table 4.2: Scaling comparison measured relatively to the previous value benchmark MAPE; the subscript s refers to standard scaled data and r refers to robust scaled data; bold values refer to the top score on a dataset

with the LSTM having the best value. The high prediction errors of the shampoo dataset, around 2000% on average, is probably due to its small sample size. Also for the beverage sales dataset all models were able to beat the previous value benchmark, but only the linear regressor and the MLP were able to beat all three benchmarks with the linear regressor having the lowest error. Regarding the monotonic rising line, only the LSTM and the linear regressor could beat the previous value benchmark with the linear regressor finding the perfect solution $y(x) = x$, having 0% error.

4.2 Scaling methods analysis

After the first evaluation the scaling techniques explained in section 3.3.1 were compared. Therefore, the results created with the standard scaled datasets in section 4.1 were compared to the models predicting the robust scaled datasets. The results of this comparison are summarized in the table 4.2. Because the previous value is used as the main baseline for the evaluation, the MAPE results of the models were divided by the results of the previous value for the respective datasets. Therefore, the values in the table represent how large the prediction error of the models was compared to the previous value baseline, e.g. 100% being just as good as the baseline.

The scaling comparison displayed in table 4.2 shows a diverse result. For the neural networks the standard scaling yields a significantly better result on each dataset. Instead, the linear regressor shows different preferences depending on the exact dataset. On the euro dataset the linear regressor performs equally good with both scaling methods. On the MSCI World dataset it performs slightly better with the robust scaler, whereas, on the Bitcoin dataset it performs better with the standard scaler by a larger margin. In contrast, the XGBoost predictions were better when this model was used with the robust scaled data compared to the standard scaled data. The reason for that might be that the XGBoost is less easily distracted by outliers in the dataset than the other models or that it needs a more stable distribution of the values. These results have then been considered in the experiments in the upcoming sections of this chapter.

	LR	XGBoost	MLP	LSTM
MXWO$_s$	118.77	1190.54	150.17	159.91
MXWO$_i$	126.77	1338.93	305.77	119.86
Euro$_s$	**99.96**	127.62	253.38	1422.17
Euro$_i$	**99.89**	123.66	260.01	1306.58
Bitcoin$_s$	110.38	124.36	153.80	142.51
Bitcoin$_i$	118.18	194.40	198.21	343.48

Table 4.3: Indicator evaluation measured relatively to the previous value benchmark MAPE; the subscript s refers to the data without indicators and i refers the data preprocessed with indicators; bold values refer to the top score on a dataset

4.3 Indicator preprocessing analysis

In this section the application of the indicators described in section 3.2 is evaluated. For this, the results of predicting the standard scaled data was compared to the results predicting the data preprocessed with indicators before applying standard scaling. The results of this experiment are presented in the table 4.3.

The results of the indicator evaluation displayed in table 4.3 do not show a clear picture. The only model with a clear preference is the MLP model that generates worse predictions with the indicator processed data. Every other model performs better with the indicators on some datasets and on other datasets they perform better without indicators. Therefore, for the further experiments both types of datasets were applied.

4.4 XGBoost analysis

For hyperparameter tuning of the XGBoost model, three different configurations of boosters, GBTree, Dart and GBLinear as explained in section 3.4.3, were compared. This comparison is displayed in table 4.4. Based on the mixed results for this model in the experiments of section 4.2 the evaluation was made for both, the datasets robustly scaled and standard scaled. Additionally, the evaluation was made for the financial data processed with indicators and the data processed without indicators. Therefore, 4 different combinations of preprocessing was evaluated for each dataset.

Evaluating the results of table 4.4 the first thing that can be seen is that the GBLinear booster performs very differently from the tree based boosters. For the most cases this booster clearly generates the best predictions. But in the case of the euro and Bitcoin datasets used without indicators this booster produces very large errors. The only occurrence where a tree booster, the GBTree, performed better than the GBLinear was for the shampoo dataset. Because of the good performance of the GBLinear, especially on standard scaled data with indicators, this setup was preferred in further experiments.

Further, the XGBoost was used to make a feature analysis of the input data of

	$\mathbf{XGBoost_{GBTree}}$	$\mathbf{XGBoost_{Dart}}$	$\mathbf{XGBoost_{GBLinear}}$
$\mathbf{MXWO_s}$	1205.49	1312.94	242.00
$\mathbf{MXWO_r}$	1119.19	1160.31	247.34
$\mathbf{MXWO_{i,s}}$	1329.43	1388.44	137.06
$\mathbf{MXWO_{i,r}}$	1410.01	1507.57	139.43
$\mathbf{Euro_s}$	127.62	1056.41	∞
$\mathbf{Euro_r}$	124.23	815.27	∞
$\mathbf{Euro_{i,s}}$	123.70	1065.27	120.14
$\mathbf{Euro_{i,r}}$	124.19	120.55	122.67
$\mathbf{Bitcoin_s}$	123.79	387.33	7285.87
$\mathbf{Bitcoin_r}$	115.34	388.29	∞
$\mathbf{Bitcoin_{i,s}}$	217.44	643.04	112.22
$\mathbf{Bitcoin_{i,r}}$	190.89	791.45	112.72
Line	11597.99	67649.89	**0.00**
Shampoo	**66.42**	115.22	80.49
Beverages	96.40	221.19	**63.34**

Table 4.4: XGBoost tuning measured relatively to the previous value benchmark MAPE; the subscript s refers to the standard scaled data, r refers to robust scaled data and i refers the data preprocessed with indicators; bold values refer to the top score on a dataset and extremely large values have been replaced with ∞

which some interesting findings will be discussed here. As it can be seen in figure 4.1, or figure A.1 in the appendix for a display of all features, the $Close_{t-1}$ seems to be the most relevant feature for the Bitcoin dataset. This is the value that the previous value benchmark that the models are trying to beat is based upon. The second most important feature is the $Volume_{t-1}$. This ranking is reasonable because these two features from time step $t-1$ are close values on the timescale to the $Close_t$ that they are trying to predict. Additionally, the two features are more independent of each other than other features. E.g. the $Close_{t-1}$ and $Open_{t-1}$ features describe both the price property, but $Volume_{t-1}$ describes the volume, a different property.

Nevertheless it was observed, that the XGBoost algorithm seemed to be biased to gibe the features at the beginning of the data vector a higher importance, even though this only happens when the influence of two features provided exactly the same prediction outcome. But *Close* being the first and $Volume$ being the second feature in the vectors did have an influence the importance rating.

Lastly, $Close_{t-10}$ was rated to be the third most important feature and after that the $Volume_{t-2..t-10}$ was ranked. The feature importance is similarly distributed for the MSCI World dataset. An example decision tree created by the XGBoost with the GBTree booster is displayed in the figures A.4 and A.5 in the appendix.

In contrast, for the euro dataset the feature importance distribution ranks the $Close_{t-10}$ as the most important feature, as it can be seen in figure 4.2 and for a full distribution in the figure A.2 in the appendix. The reason for that might be the

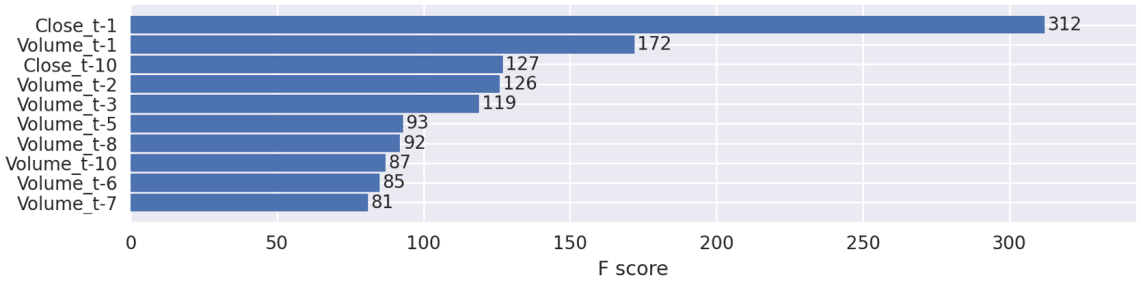


Figure 4.1: 10 most important GBTree features of Bitcoin data

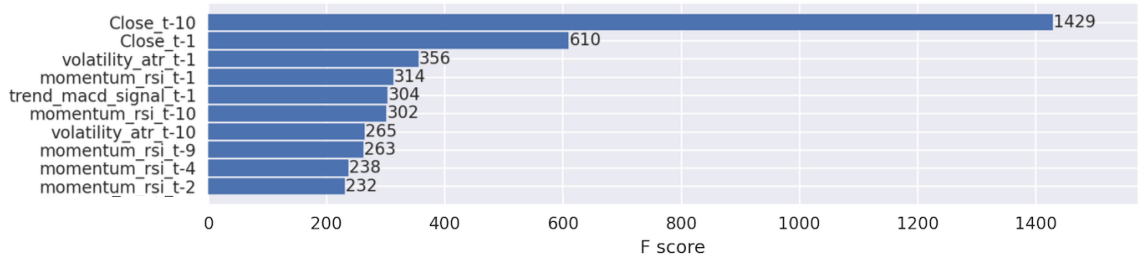


Figure 4.2: 10 most important GBTree features of euro data

more granular time intervals of the dataset of 15 minutes compared to the one-day time interval of the other financial datasets. The second interesting observation that can be seen in figure 4.2 is that the *Close* features were actually ranged higher than the indicators. So it seems that the indicators were not the most helpful information source for the model. But the reason for this ranking could also be because the *Close* feature was provided as the first feature in the matrix as it was described earlier.

Furthermore, the feature analysis of the Dart booster looked different from the GBTree booster. As it can be seen in figure 4.3 and for a full distribution in the figure A.3 in the appendix, the dart booster focused much more on a single feature.

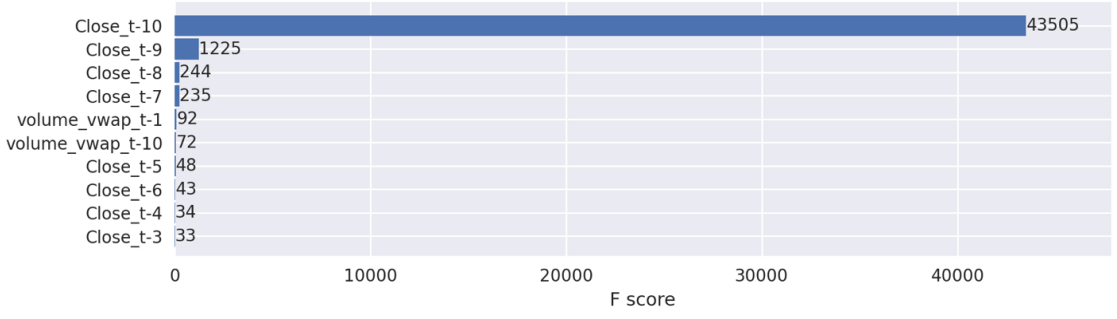


Figure 4.3: 10 most important Dart features of euro data

	$\mathbf{MLP_1}$	$\mathbf{MLP_2}$	$\mathbf{MLP_{1000}}$	$\mathbf{LSTM_1}$	$\mathbf{LSTM_2}$	$\mathbf{LSTM_{1000}}$
$\mathbf{MXWO_s}$	150.17	169.31	2432.72	159.91	220.46	765.30
$\mathbf{MXWO_i}$	305.77	257.63	1629.72	119.86	162.94	1810.34
$\mathbf{Euro_s}$	253.38	628.29	3272.56	1422.17	3983.56	4176.09
$\mathbf{Euro_i}$	260.01	787.23	4219.67	1306.58	3807.97	4352.15
$\mathbf{Bitcoin_s}$	153.80	162.12	425.77	142.51	145.00	426.07
$\mathbf{Bitcoin_i}$	198.21	234.71	1421.67	343.48	862.60	1516.65
Line	160.85	1333.11	109828.41	**77.18**	12629.31	109207.38
Shampoo	64.87	71.32	321.66	63.35	**55.38**	343.16
Beverages	64.54	**65.44**	297.85	73.35	79.14	789.52

Table 4.5: Neural network tuning measured relatively to the previous value benchmark MAPE; the subscript s refers to the standard scaled data and i refers the data preprocessed with indicators; bold values refer to the top score on a dataset

4.5 Neural network analysis

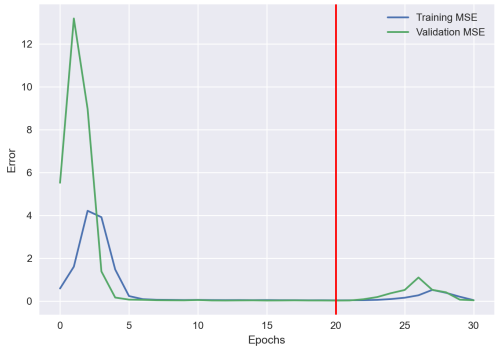
After the hyperparameter tuning of the XGBoost has been made, the hyperparameters of the neural networks were analyzed as well. For this, the amount of layer-blocks of the model as well as the amount of neurons have been adapted. The main findings of these experiments are presented in table 4.5, figure 4.4 and there is further information in the appendix.

In table 4.5 different neural network configurations are listed. The index 1 and 2 refers to the one and two layer neural network configurations with the default configuration of 200 neurons. The results show that almost all the one-layer models performed better on the financial datasets, as well as the monotonic rising line data, while the two-layer models seemed to perform good on the sales datasets. Therefore, for further experiments of this work, one-layered neural networks were preferred.

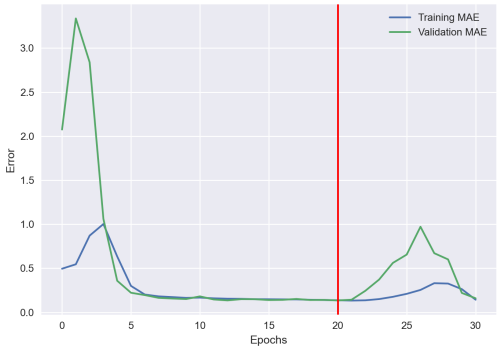
Furthermore, also different configurations of amounts of neurons for the layer were tested. In table 4.5 the index 1000 relates to a neural network with 1000 neurons in the single main layer. As it can be seen, the 200 neuron configuration (MLP_1) performed better than the 1000 neuron configuration. A more in depth analysis of number of neurons can be seen in table A.1 in the appendix. From these further evaluations it was concluded that for the best performing number of neurons the default parameter 200 for the MLP model was chosen. Instead, the LSTM model seemed to perform best with a number of neurons of 400 as the table A.1 suggests.

Batch normalization for the neural networks was tested as well, but it did make the prediction results significantly worse. Therefore, batch normalization was omitted in further experiments.

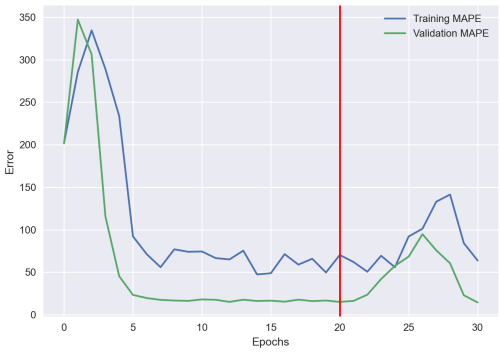
Additionally, the training curves of the neural networks were observed to check for possible overfitting of the data and how the training behavior of the models was. As it can be seen in figure 4.4 the training of the MLP saturated with the lowest MAPE



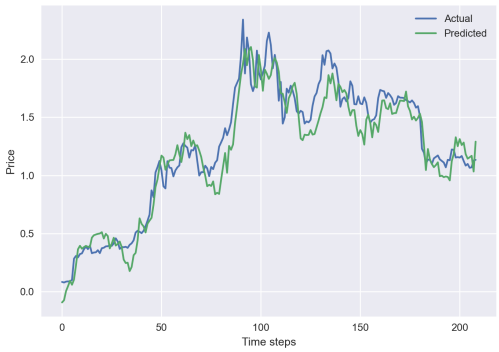
(a) MSE during training



(b) MAE during training



(c) MAPE during training



(d) Prediction of the validation curve

Figure 4.4: Neural network training graphs of an example training of an MLP on Bitcoin standard scaled data with indicators; the red lines correspond to the training epoch where the best model was found

validation error after about 20 epochs on average. Sometimes the training behavior went up for the first couple of training epochs, as in figure 4.4, and after the first couple of epochs the error curve fell quickly and then saturated. The same behavior could be observed for the LSTM model. The neural network training took most of the time around a maximum of 15 minutes per run. For the large euro dataset the neural networks trained for multiple hours, instead.

4.6 Lag size analysis

The last parameter that was modified is the lag size parameter for the prediction models. The default value of 10 time steps into the past that the model received with each sample was extended to a lag size of 30 time steps. This corresponds to approximately one month of data in the Bitcoin and MSCI World datasets and 7.5 hours in the euro dataset. The shampoo dataset could not be predicted with this lag size because the training set only consisted of 24 samples. The comparison was made with all models, while the XGBoost used the GBLinear booster and the LSTM used a number of 400 neurons.

	$\mathbf{LR_{10}}$	$\mathbf{LR_{30}}$	$\mathbf{XGBoost_{10}}$	$\mathbf{XGBoost_{30}}$
$\mathbf{MXWO_s}$	118.77	135.13	239.00	128.33
$\mathbf{MXWO_i}$	126.77	216.19	138.04	180.34
$\mathbf{Euro_s}$	**99.96**	100.04	∞	172.56
$\mathbf{Euro_i}$	**99.89**	99.96	120.10	124.33
$\mathbf{Bitcoin_s}$	110.38	123.21	7319.79	120.15
$\mathbf{Bitcoin_i}$	118.18	134.26	119.67	128.19
Line	**0.00**	**0.00**	**0.00**	**0.00**
Beverages	62.56	28.18	64.63	**22.40**
	$\mathbf{MLP_{10}}$	$\mathbf{MLP_{30}}$	$\mathbf{LSTM_{10}}$	$\mathbf{LSTM_{30}}$
$\mathbf{MXWO_s}$	150.17	279.42	159.91	130.04
$\mathbf{MXWO_i}$	305.77	420.40	119.86	143.82
$\mathbf{Euro_s}$	253.38	414.66	1422.17	1351.17
$\mathbf{Euro_i}$	260.01	434.61	1306.58	1154.68
$\mathbf{Bitcoin_s}$	153.80	204.86	142.51	135.90
$\mathbf{Bitcoin_i}$	198.21	432.47	343.48	1217.70
Line	160.85	1773.60	77.18	97.32
Beverages	64.54	54.53	73.35	296.91

Table 4.6: Lag size analysis measured relatively to the previous value benchmark MAPE; the subscript 10 refers to a lag size of 10 and the subscript 30 refers to a lag size of 30; bold values refer to the top score on a dataset and extremely large values have been replaced with ∞

It can be seen in the table 4.6 that the linear regressor and the MLP showed a clear preference for a lag size of 10. In contrast, the XGBoost and the LSTM were quite unclear and sometimes performed better with a lag size of 10 and sometimes with a lag size of 30. Because of the preference of some models of the lag size of 10, this parameter was chosen for the final test evaluation in the next section.

4.7 Test set evaluation

After the prediction setup has been tuned as good as possible, as it was explained in the previous sections of this chapter, the final evaluation on the unseen test set, described in section 3.5, was made. The validation set was added to the training set to have even more training examples and the tests were executed on the datasets processed with and without indicators and scaled with the standard scaler. Additionally, a lag size of 10 time steps were chosen. The XGBoost was set up to use the GBLinear booster and the neural networks used only one block of artificial neurons, as described in section 3.4.4. The MLP was configured to use 200 artificial neurons, while the LSTM was tuned to use 400 artificial neurons. And the batch normalization was turned off for the neural networks.

First the benchmarks were recalculated, as shown in table 4.7. The MAPE results look similar to the benchmark calculation on the validation set in table 4.1; most values stayed approximately the same. The only value that changed significantly is the MSCI

	Prev	Mean	Median
MXWO	**171.35**	2803.76	1370.30
Euro	**24.09**	3756.71	1250.35
Bitcoin	**588.26**	6270.68	4721.82
Line	**4.02**	278.80	279.00
Shampoo	2395.47	**2031.90**	2270.07
Beverages	1397.02	1065.23	**1034.64**

Table 4.7: Benchmarks on the test set measured with MAPE; bold values refer to the top score on a dataset

	LR	XGBoost	MLP	LSTM
MXWO$_s$	106.70	114.10	176.25	207.69
MXWO$_i$	113.02	166.57	257.16	126.96
Euro$_s$	**99.96**	196.06	512.87	1974.30
Euro$_i$	**99.88**	132.25	320.05	1680.45
Bitcoin$_s$	120.05	105.18	132.32	139.72
Bitcoin$_i$	118.26	106.17	173.09	308.66
Line	**1.99**	3.23	117.16	667.41
Shampoo	55.13	58.64	24.30	**19.40**
Beverages	71.06	70.75	**70.14**	74.81

Table 4.8: Model performance on the test set measured relatively to the previous value benchmark MAPE; the subscript s refers to the standard scaled data and i refers the data preprocessed with indicators; bold values refer to the top score on a dataset and extremely large values have been replaced with ∞

World benchmark, it nearly halved from 325.70% to 171.35% of error. The reason for that can be explained by the decreased volatility of the index after 2017, as it can be seen in figure 3.1.

When analyzing the results of the test displayed in table 4.8, it can be seen that the results in comparison to the first evaluation, shown in table 4.1 could be only slightly improved. While the performance of the XGBoost model could be drastically improved due to the application of the GBLinear booster, it still could not beat the previous value benchmark on the financial datasets. On the financial datasets, only the linear regressor predicting the euro dataset could beat the benchmark of the previous value. The linear regressor performance on the euro dataset could be slightly improved by the application of indicators. And the performance of the neural networks stayed comparably the same on the financial datasets, while improving drastically on the shampoo and beverage sales datasets.

5 Conclusion

5.1 Discussion of results

After the experiments of the previous chapter have been evaluated, in this section the results of this work are discussed. Doing that, the four research questions of section 1.2 are discussed:

Q1: Is it possible to build an algorithm to predict financial time series using ML techniques?

Only one model, the linear regressor, could create a prediction on one dataset, the euro price, that actually outperformed the previous value benchmark; this was done by a slight margin of 0.12%. But there are some remarks that make this finding questionable: on the one hand it should first be confirmed with further analysis that this outperforming of the baseline will actually continue in other market situations. On the other hand due to the tiny advantage over the previous value benchmark it is questionable if it might be possible create a profitable trading strategy using this model. As a last remark it has to be added, that the linear regressor was planned to be used as a simple benchmark model and there was no hyperparameter tuning possible for the linear regressor. The performance of the linear regressor could only be improved with the processing of the datasets with indicators.

Q2: How are different ML prediction models differing in their prediction quality?

In contrast, the other ML models were expected to perform better than the linear regressor, but they could not outperform the previous value benchmark for a single financial dataset. This finding actually seems counterintuitive, but it might be the case that the more complex models also were more likely to try to fit patterns in the data that did not matter for the price development. This could be the case because there is a lot of randomness in financial data, as financial time series are a random walk for the most part [TG04]. Therefore, more complex models seem to be prone to fit random patterns; what in turn produces even worse results as a simple previous value benchmark.

All in all, it can be concluded that the EMH of financial markets is almost correct. For the most part, financial markets are completely random and unpredictable, but there is a still a comparably small possibility to create predictions, as it could be done by a small margin by one combination of dataset and model.

Q3: How are the prediction models behaving on different datasets?

In the final test all models were able to outperform the baselines on non-financial datasets with the only exception of the neural networks predicting the monotonic line

dataset. Thus, it was concluded that there was no implementation error, that would be the reason for the bad prediction quality for the financial time series. Even the best model, the linear regressor could only predict one of the three datasets, the euro price, better than the baseline.

Furthermore, the linear models could provide the best predictions for the monotonic rising line dataset. In contrast, the neural networks could predict the sales datasets the best, see table 4.8.

Q4: In what ways is it possible to improve the model's prediction quality?
It was possible to increase the predictions of the XGBoost by using the GBLinear booster over the default tree boosting methods. It could also be the case that this booster simplifies the capabilities of the model and therefore avoids the effect described in the answer to question Q1, as simpler models seem to have an easier time when dealing with near-random data. Additionally, the performance of the LSTM could be improved by adding an increased the number of neurons. But it has to be added, that even though improvements were made, the tuned models still could not outperform the previous value benchmark. Therefore, it is questionable how decisive these improvements actually were.

Another more important improvement that could be made is the processing of the data with indicators. Even though the data processed with indicators was not always easier to be predicted than the default OHLCV data, the performance of linear regressor on the euro price dataset could be further improved with this technique. Without indicator processing the linear regressor outperformed the previous value benchmark only by 0.04% and after the application of indicators this gap could be tripled to 0.12%.

5.2 Areas of further research

Future research could tune and test more hyperparameters values and hyperparameter types of the XGBoost and the neural networks. Also, more models could be developed in future research. The number of runs that needed to be executed for a single parameter configuration was already quit high because of the large amount of datasets. Therefore, it is suggested that with more computational resources and professional high-performance computing it would be possible to deploy more and longer experiments. Additionally, the resulting values would be more reliable, if instead of relying on single runs, multiple runs of the same configuration can have been executed and the average results could be calculated.

Furthermore, it could be observed in more detail how much the distribution of the feature analysis is based on the feature order. When this is understood in more depth it could be analyzed how well the features of the indicators actually performed in comparison to data not processed with indicators. Additionally, it could be observed

in more depth which indicators perform best for this kind of tasks and test multiple indicator combinations.

Moreover, future research could focus on enriching the data with more information. This way, it might be possible to improve prediction quality of the models. One approach how this could be done is using news datasets or crawl data from the Twitter API[29]. Additionally, this data could be further classified using sentiment analysis. Furthermore, fundamental financial data like quarterly reports of companies could be processed and added to the dataset. And even more real world data could be added to the dataset, like political information or calendar data. Using real world data would be helpful to detect paradigm change points in data to improve the interpretability. This would be especially helpful on datasets like the Bitcoin price because its performance has been heavily correlated with political regulations and tweets of celebrities like Elon Musk. Besides, it could also be observed how combining the datasets of multiple time series would affect the prediction performance.

Another idea is to apply further preprocessing to increase time resolution of the financial time series when there is a lot of activity in the market and decrease the time resolution when there is less activity in the market. The reason why this might work is that the time dimension of financial time series is not really linear, instead there are times when the markets are moving faster and times when they are moving slower [DP18].

[29]https://developer.twitter.com/en/docs/twitter-api

Bibliography

[Ama21] Kimberly Amadeo. Make Financial Markets Work for You. https://www.thebalance.com/an-introduction-to-the-financial-markets-3306233, August 2021.

[And21a] Chip Anderson. Average True Range (ATR) [ChartSchool]. https://school.stockcharts.com/doku.php?id=technical_indicators:average_true_range_atr, September 2021.

[And21b] Chip Anderson. Relative Strength Index (RSI) [ChartSchool]. https://school.stockcharts.com/doku.php?id=technical_indicators:relative_strength_index_rsi, September 2021.

[And21c] Chip Anderson. Volume Weighted Average Price (VWAP) [ChartSchool]. https://school.stockcharts.com/doku.php?id=technical_indicators:vwap_intraday, September 2021.

[Bah21] Pragati Baheti. 12 Types of Neural Networks Activation Functions: How to Choose? https://www.v7labs.com/blog/neural-networks-activation-functions, September 2021.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, 2006.

[BKM08] Zvi Bodie, Alex Kane, and Alan Marcus. *Investments 8th Edition*. McGraw-Hill/Irwin, Boston, 8th edition edition, June 2008.

[BMK+17] Filippo Maria Bianchi, Enrico Maiorino, Michael C. Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting. *arXiv:1705.04378 [cs]*, 2017.

[Bro17] Jason Brownlee. How to Scale Data for Long Short-Term Memory Networks in Python. https://machinelearningmastery.com/how-to-scale-data-for-long-short-term-memory-networks-in-python/, July 2017.

[Bro20] Jason Brownlee. How to Scale Data With Outliers for Machine Learning. https://machinelearningmastery.com/robust-scaler-transforms-for-machine-learning/, May 2020.

[Bro21] Jason Brownlee. XGBoost for Regression. https://machinelearningmastery.com/xgboost-for-regression/, March 2021.

[CG16] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[Che21a] James Chen. Technical Indicator Definition. https://www.investopedia.com/terms/t/technicalindicator.asp, September 2021.

[Che21b] Brandon Chez. Cryptocurrency Prices, Charts And Market Capitalizations | CoinMarketCap. https://coinmarketcap.com/, May 2021.

[Cox21] Nick Cox. Difference between regression analysis and curve fitting. https://stats.stackexchange.com/questions/151452/difference-between-regression-analysis-and-curve-fitting, September 2021.

[CZ05] Jakša Cvitanic and Fernando Zapatero. Introduction to the Economics and mathematics of financial markets, 2005.

[Dab21] Caner Dabakoglu. Time Series Forecasting - ARIMA, LSTM, Prophet. https://kaggle.com/cdabakoglu/time-series-forecasting-arima-lstm-prophet, September 2021.

[DMGGR15] Arnaud De Myttenaere, Boris Golden, Bénédicte Le Grand, and Fabrice Rossi. Using the Mean Absolute Percentage Error for Regression Models. *arXiv:1506.04176 [cs, stat]*, June 2015.

[Dom21] Patti Domm. A stunning fall and a recovery: How the stock market has evolved one year since Covid hit. https://www.cnbc.com/2021/03/12/a-stunning-fall-and-a-recovery-how-the-stock-market-has-evolved-one-year-since-covid-hit.html, March 2021.

[DP18] Marcos Lopez De Prado. *Advances in Financial Machine Learning*. John Wiley & Sons, 2018.

[Fam70] Eugene F. Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, 25(2):383–417, 1970.

[Fer21a] Henry A. Fernandez. MSCI World Index. https://www.msci.com/World, September 2021.

[Fer21b] Jason Fernando. Moving Average Convergence Divergence (MACD). https://www.investopedia.com/terms/m/macd.asp, September 2021.

[Fer21c] Jason Fernando. Volume-Weighted Average Price (VWAP). https://www.investopedia.com/terms/v/vwap.asp, September 2021.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, March 2010.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[Gup17] Vikas Gupta. Understanding Feedforward Neural Networks | LearnOpenCV. https://learnopencv.com/understanding-feedforward-neural-networks/, October 2017.

[GVS+16] Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. Contextual LSTM (CLSTM) models for Large scale NLP tasks. *arXiv:1602.06291 [cs]*, May 2016.

[HA18] Rob J.. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts: Melbourne, Australia., 2018.

[Hal21] Mary Hall. Enter Profitable Territory With Average True Range. https://www.investopedia.com/articles/trading/08/atr.asp, September 2021.

[Hay21a] Adam Hayes. Financial Markets. https://www.investopedia.com/terms/f/financial-market.asp, August 2021.

[Hay21b] Adam Hayes. Heteroskedasticity. https://www.investopedia.com/terms/h/heteroskedasticity.asp, August 2021.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015.

[IG21] IG. Top 10 most traded currency pairs. https://www.ig.com/en/trading-strategies/top-10-most-traded-currency-pairs-191206, September 2021.

[Jan21] Michal Januszewski. EURUSD - 15m - 2010-2016. https://kaggle.com/meehau/EURUSD, September 2021.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017.

[Ken21] Will Kenton. Zero-Sum Game. https://www.investopedia.com/terms/z/zero-sumgame.asp, August 2021.

[Kue21] Justin Kuepper. Major World Stock Market Indexes. https://www.thebalance.com/major-world-stock-market-indexes-4148491, September 2021.

[Mar21] Boris Marjanovic. Huge Stock Market Dataset. https://kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs, September 2021.

[Mav21] Jack B. Maverick. How Is the Exponential Moving Average (EMA) Formula Calculated? https://www.investopedia.com/ask/answers/122314/what-exponential-moving-average-ema-formula-and-how-ema-calculated.asp, September 2021.

[Mit21a] Cory Mitchell. Aroon Indicator Definition and Uses. https://www.investopedia.com/terms/a/aroon.asp, September 2021.

[Mit21b] Cory Mitchell. Trading With VWAP and MVWAP. https://www.investopedia.com/articles/trading/11/trading-with-vwap-mvwap.asp, September 2021.

[Mit21c] Cory Mitchell. What Is the Aroon Oscillator? https://www.investopedia.com/terms/a/aroonoscillator.asp, September 2021.

[Mul21] Rob Mulla. [Tutorial] Time Series forecasting with XGBoost. https://kaggle.com/robikscube/tutorial-time-series-forecasting-with-xgboost, September 2021.

[Mur99] John J. Murphy. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. Prentice Hall Press, January 1999.

[Mut21] Lisa C. Muth. How to read a log scale: The chart that can't start at zero. https://blog.datawrapper.de/weeklychart-logscale2/, September 2021.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[Pad21] Darío López Padial. Technical Analysis Library in Python, September 2021.

[Pot21] Charles Potters. Top 7 Technical Analysis Tools. https://www.investopedia.com/top-7-technical-analysis-tools-4773275, September 2021.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Pyk21] Kurtis Pykes. Fighting Overfitting with L1 or L2 Regularization - Which One Is Better? https://neptune.ai/blog/fighting-overfitting-with-l1-or-l2-regularization, April 2021.

[Raj21] Sudalai Rajkumar. Cryptocurrency Historical Prices. https://kaggle.com/sudalairajkumar/cryptocurrencypricehistory, September 2021.

[Reg20] Sam Rega. How Robinhood and Covid opened the floodgates for 13 million amateur stock traders. https://www.cnbc.com/2020/10/07/how-robinhood-and-covid-introduced-millions-to-the-stock-market.html, October 2020.

[Rob21] Harry V. Roberts. Statistical versus Clinical Prediction of the Stock Market – ScienceOpen. https://www.scienceopen.com/document?vid=fd7b0c49-1fad-4fa5-be61-e55bbcfc99d5, August 2021.

[Ros21] Sean Ross. How do momentum and trends differ? https://www.investopedia.com/ask/answers/121614/what-are-main-differences-between-momentum-and-trend.asp, August 2021.

[Rud17] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*, June 2017.

[RWI90] Cleveland Robert, C William, and Terpenning Irma. STL: A seasonal-trend decomposition procedure based on loess. *Journal of official statistics*, 6(1):3–73, 1990.

[SCS16] Adam Summerville, Michael Cook, and Ben Steenhuisen. Draft-Analysis of the Ancients: Predicting Draft Picks in DotA 2 using Machine Learning. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, September 2016.

[SLJ+14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *arXiv:1409.4842 [cs]*, September 2014.

[SPE21] SPEC INDIA. Top 10 Best Programming Languages For Machine Learning In 2021. https://www.spec-india.com/blog/programming-languages-for-machine-learning, September 2021.

[SPGK21] Arnab Sen, Brian Pu, Si Gao, and Elias Krauklis. Impact of Retail Options Trading. https://www.docdroid.net/5gM68EW/barclays-us-equity-derivatives-strategy-impact-of-retail-options-trading-pdf, August 2021.

[SW10] Claude Sammut and Geoffrey I. Webb, editors. *Encyclopedia of Machine Learning*. Springer US, Boston, MA, 2010.

[TG04] Allan Timmermann and Clive W. J. Granger. Efficient market hypothesis and forecasting. *International Journal of Forecasting*, 20(1):15–27, January 2004.

[Tim21] Time Zone Converter. Forex Market Hours - Forex Market Time Converter. https://forex.timezoneconverter.com/, August 2021.

[Tra21] TradingView. Volume Weighted Average Price (VWAP). https://www.tradingview.com/support/solutions/43000502018-volume-weighted-average-price-vwap/, September 2021.

[U.S21] U.S. Census Bureau. Merchant Wholesalers, Except Manufacturers' Sales Branches and Offices: Nondurable Goods: Beer, Wine, and Distilled Alcoholic Beverages Sales. https://fred.stlouisfed.org/series/S4248SM144NCEN, September 2021. Last updated: 2021-08-06 09:01:02-05.

[VEB+17] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv:1708.04782 [cs]*, August 2017.

[Vip21] Tim Vipond. Exponentially Weighted Moving Average (EWMA). https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/exponentially-weighted-moving-average-ewma/, September 2021.

[Vog19] Markus Vogl. Financial Time Series Analysis using Wavelets & Neural Networks. https://www.youtube.com/watch?v=ffZsObHUoFA, August 2019.

[Vog21] Markus Vogl. Data Science für algorithmische Finanzmarkt- & Zeitreihenanalyse - YouTube. https://www.youtube.com/playlist?list=PLFXw4NpfUWMi4enJ2_jtKjSwHPhnNbd2G, August 2021.

[WMH98] Steven Wheelwright, Spyros Makridakis, and Rob J Hyndman. *Forecasting: Methods and Applications*. John Wiley & Sons, 1998.

[Woo19] Thomas Wood. F-Score. https://deepai.org/machine-learning-glossary-and-terms/f-score, May 2019.

[WP97] Terry J Watsham and Keith Parramore. *Quantitative Methods in Finance*. Cengage Learning EMEA, 1997.

Appendix

	$\mathbf{MLP_{50}}$	$\mathbf{MLP_{100}}$	$\mathbf{MLP_{400}}$	
$\mathbf{MXWO_s}$	184.12	574.15	243.97	
$\mathbf{MXWO_i}$	505.84	195.31	198.32	
$\mathbf{EURUSD_s}$	327.76	398.79	191.64	
$\mathbf{EURUSD_i}$	406.47	381.86	307.11	
$\mathbf{Bitcoin_s}$	195.70	139.33	153.34	
$\mathbf{Bitcoin_i}$	452.24	291.24	223.65	
Line	350.78	184.12	517.67	
Shampoo	79.30	59.51	57.09	
Beverages	73.37	65.11	59.00	
	$\mathbf{LSTM_{50}}$	$\mathbf{LSTM_{100}}$	$\mathbf{LSTM_{400}}$	$\mathbf{LSTM_{500}}$
$\mathbf{MXWO_s}$	160.98	187.28	138.11	143.59
$\mathbf{MXWO_i}$	330.70	179.54	133.20	240.95
$\mathbf{EURUSD_s}$	1451.81	1369.68	1406.48	1369.36
$\mathbf{EURUSD_i}$	1378.12	1288.37	1284.13	1420.95
$\mathbf{Bitcoin_s}$	143.87	129.92	126.96	148.82
$\mathbf{Bitcoin_i}$	192.31	685.80	747.43	465.27
Line	**14.99**	578.30	326.85	382.55
Shampoo	**32.32**	58.25	59.19	55.19
Beverages	73.21	70.58	75.61	73.28

Table A.1: Additional neural network experimentation measured relatively to the previous value benchmark MAPE; the subscript refers to the number of neurons in the main layer of a neural network; bold values refer to the top score on a dataset

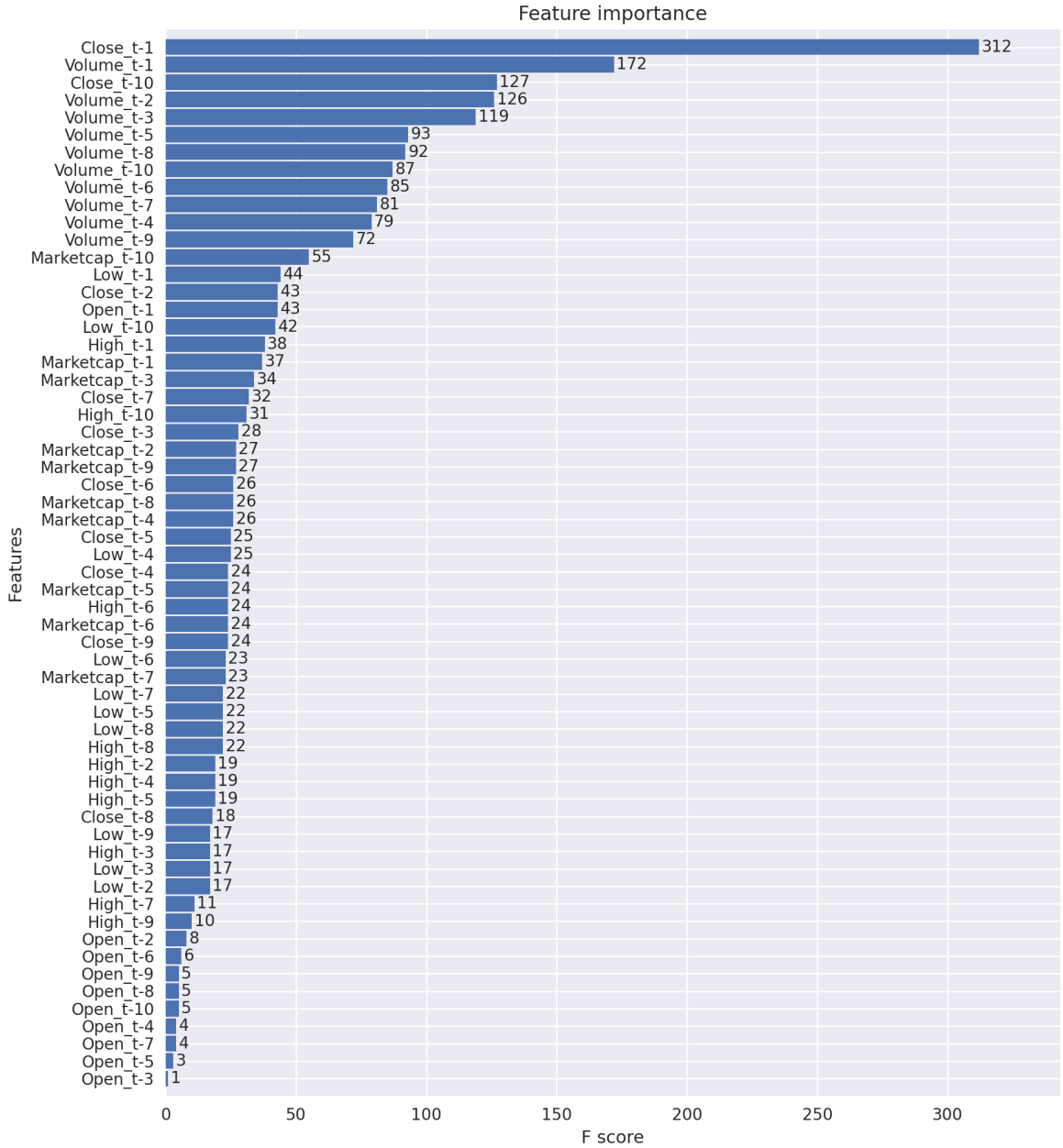


Figure A.1: Full GBTree feature analysis of Bitcoin data

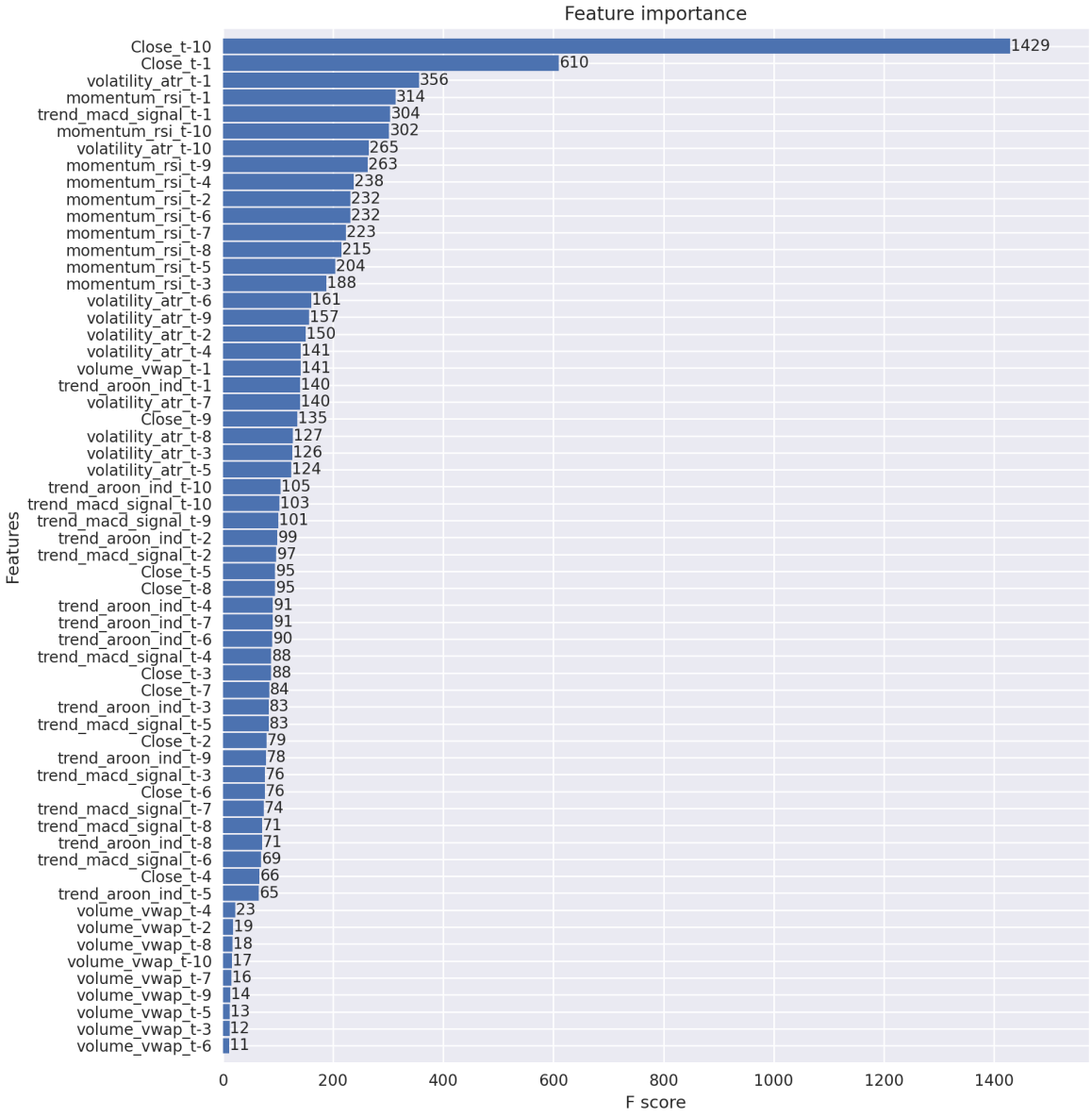


Figure A.2: Full GBTree feature analysis of euro data

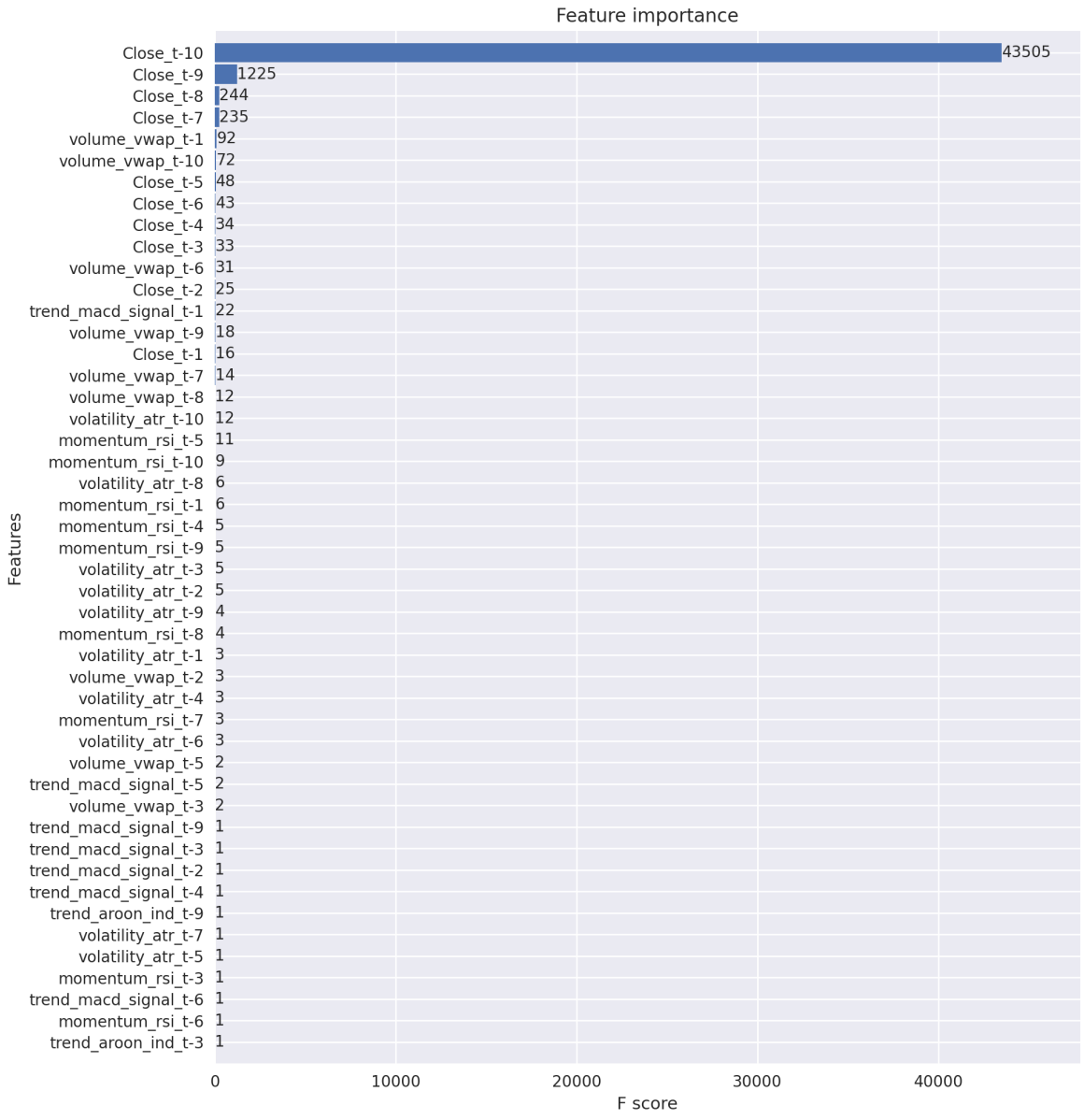


Figure A.3: Full Dart feature analysis of euro data

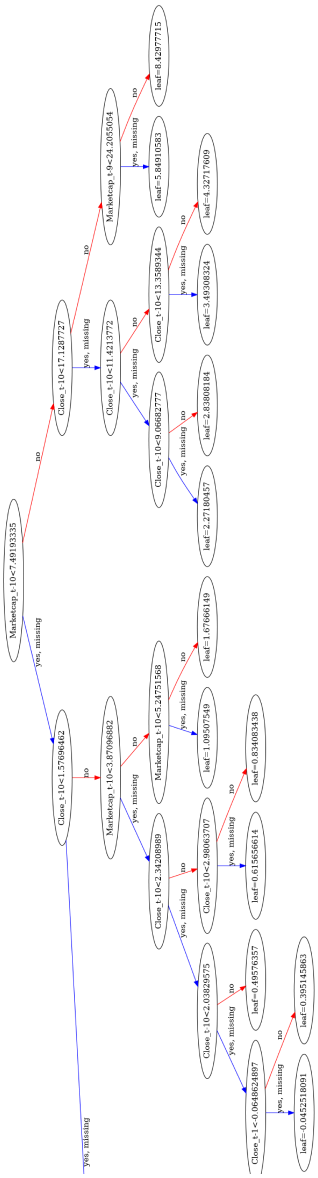


Figure A.4: Decision tree example part 1; the right half; created by XGBoost using the GBTree booster for Bitcoin data

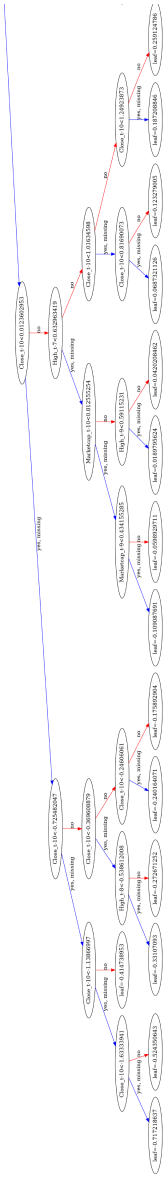


Figure A.5: Decision tree example part 2; the left half; created by XGBoost using the GBTree booster for Bitcoin data

Acknowledgments

Vielen Dank für Alles an Prof. Dr. Kühnberger, Ludwig Schallner, Johannes, Hanna, Maria und Walter.

Confidentiality clause

This work may only be made available to the first and second reviewers and authorized members of the board of examiners. Any publication and duplication of this Master's Thesis, even in part, requires the expressed permission of the author Tillmann Brodbeck and IBM Deutschland GmbH[30].

[30]https://www.ibm.com/de-de/about

Declaration of authorship

I hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

City, Date

Signature