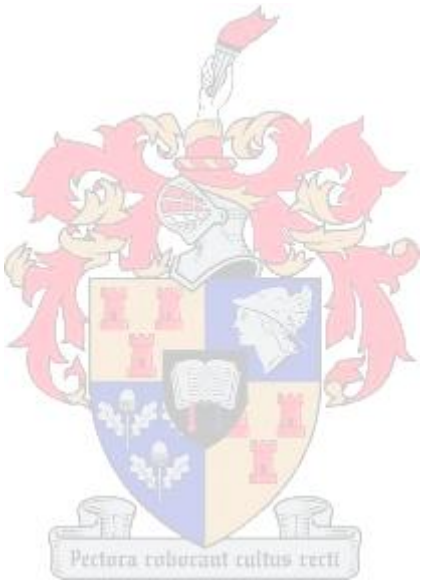
Comparison of machine learning models on different financial time series

Judene Simonis



Mini-dissertation presented in partial fulfilment of the requirements for the degree
MEng (Structured) (Industrial Engineering)
at the Faculty of Engineering at Stellenbosch University

Supervisor: Professor Andries Engelbrecht

December 2021

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 1, 2021

Abstract

The efficient market hypothesis implies that shrewd market predictions are not profitable because each asset remains correctly priced by the weighted intelligence of the market participants. Several companies have shown that the efficient market hypothesis is invalid. Consequently, a considerable amount of research has been conducted to understand the performance and behaviour exhibited by financial markets, as such insights would prove valuable in the quest to identify which products will provide a positive future return. Recent advancements in artificial intelligence have presented researchers with exciting opportunities to develop models for forecasting financial markets.

This dissertation investigated the capabilities of different machine learning models to forecast the future percentage change of various assets in financial markets. The financial time series (FTS) data employed are the S&P 500 index, the US 10 year bond yield, the USD/ZAR currency pair, gold futures and Bitcoin. Only the closing price data for each FTS was used. The different machine learning (ML) models that are investigated are linear regression, autoregressive integrated moving average, support vector regression (SVR), multilayer perceptron (MLP), recurrent neural network, long short term memory and gated recurrent unit. This dissertation uses an empirical procedure to facilitate the formatting, transformation, and modelling of the various FTS data sets on the ML models of interest. Two validation techniques are also investigated, namely single out-of-sample validation and walk-forward validation. The performance capabilities of the models are then evaluated with the mean square error (MSE) and accuracy metric. Within the context of FTS forecasting, the accuracy metric refers to the number of correct guesses about whether the price movement increased or decreased and the total number of guesses. An accuracy that is one percentage point above 50% is considered substantial when forecasting FTS, because a 1% edge on the market can result in a higher average return, which outperforms the market.

For the individual analysis of the single out-of-sample and walk-forward validation technique, the linear regression model was the best ML model for all FTS, because it is the most parsimonious model. The concept of a parsimonious model was disregarded when comparing and contrasting the two validation techniques. The ML models applying walk-forward validation performed the best in terms of MSE on the S&P 500 index and US 10 year bond yield. The SVR model obtained the highest accuracy of 52.94% on the S&P 500 index, and the MLP model obtained the highest accuracy of 51.26% on the US 10 year bond yield. The ML models applying single out-of-sample validation performed the best in terms of MSE on the USD/ZAR currency pair, gold futures and Bitcoin. The MLP model obtained the highest accuracy of 51.77% and 53.51% for the USD/ZAR currency pair and gold futures, respectively. The linear regression model obtained the highest accuracy of 55.04% for Bitcoin.

Acknowledgements

The author wishes to acknowledge the following people and institutions for their various contributions towards the completion of this work:

- My supervisor, Professor Andries Engelbrecht, for his insight, guidance and support in pursuit of this dissertation.
- My mentor and industry partner contact person, Zander Wessels for his insight, sharing his broad knowledge of financial market modelling and assistance during the dissertation.
- My industry partner contact person, Thomas Schlebusch, for initially providing the motivation and direction for my research.
- My family for their continuous moral support and unwavering belief in me throughout my studies.
- My friends, for the continued support, interest and encouragement throughout this year.
- My best friend and fiance, Derwalt Erasmus, for his support and guidance. Thank you for your curiosity, well thought out questions and engaging in conversations about the possibility of predicting the stock market and other financial markets with machine learning techniques.
- To the King of Kings — for His continual guidance, support and grace, as proclaimed in Hebrews 13:5 — “Never will I leave you; never will I forsake you.”

Table of Contents

List of Reserved Symbols

Symbol	Meaning
$\boldsymbol{a}$	Step size moved by gradient descent algorithm
b	Bias in neural network
β_0	Intercept for linear regression model
β_i	Quantifies the association between the variable and the response for linear regression
C	Cell state for long short term memory network
$\tilde{C}$	Vector of new candidate values for long short term memory network
d	Number of times differencing is performed
d_t	Delay layer in Elman and Jordan recurrent neural networks
ϵ	Error term
ε_t	residual error
h	Hidden layer for hidden unit j in neural network
i	Observation number in data set
I	Number of input units in a neural network
j	Number of attributes
J	Number of hidden units in a neural network
k	Iteration number
K	Number of output units in a neural network
l	Lower value of new normalised data range
L	Cost function in a neural network
L_e	Loss function in a neural network
m	Number of observations a mini-batch is comprised of when applying stochastic gradient descent
n	Total observations in data set
η	Learning rate
o_k	Output feature for target variable k in a neural network
p	Number of auto regression terms
q	Number of moving average terms
r_t	Reset gate in gated recurrent unit
u	Upper value of new normalised data range
U	Update gate in gated recurrent unit
μ	Average of changes between consecutive observations
v_{ji}	Weight of input and hidden unit in neural network
ν	Velocity variable from momentum algorithm
w_{kj}	Weight of hidden unit and output unit in neural network
$\boldsymbol{W}$	Weight matrix in a neural network
$\boldsymbol{x_i}$	Predictor measurement vector

x_i	Actual input value for observation i
x'	Normalised data point
$\boldsymbol{y_i}$	Response measurement vector
$y_{t(AR)}$	Variable predicted by the auto regression model
$y_{t(MA)}$	Variable predicted by the moving average model
$y*_t$	Variable predicted by the ARIMA model
y_i	Actual output value for observation i
y_{t-i}	Lagged values in auto regression term
$\hat{y}_i$	Predicted output value for observation i
γ_i	Regression coefficient for observation i
z_i	Input feature for observation i in a neural network
$\boldsymbol{\theta}$	Parameter matrix in a neural network
θ_i	Coefficient for lagged forecast error for observation i

List of Acronyms

Acronym	Description
$ARIMA$	Autoregressive integrated moving average
ACF	Autocorrelation function
$ADAM$	Adaptive movement estimation
ADF	Augmented Dickey-Fuller test
AI	Artificial intelligence
ANN	Artificial neural network
AR	Autoregressive model
DFD	Data flow diagram
FTS	Financial time series
$FTSEP$	Financial time series empirical procedure
GRU	Gated recurrent unit
I	Difference
$LSTM$	Long short term memory
MA	Moving average
MAE	Mean absolute error
$MAPE$	Mean absolute percentage error
ML	Machine learning
MLP	Multilayer perceptron
MSE	Mean square error
NLP	Natural language processing
$PACF$	Partial autocorrelation function
RBF	Radial basis function
$ReLU$	Rectified linear unit
$RMSProp$	Root-Mean-Squared Propagation
$RMSE$	Root mean square error
RNN	Recurrent neural network
SGD	Stochastic gradient descent
SVM	Support vector machine
SVR	Support vector regression

List of Figures

List of Tables

CHAPTER 1

Introduction

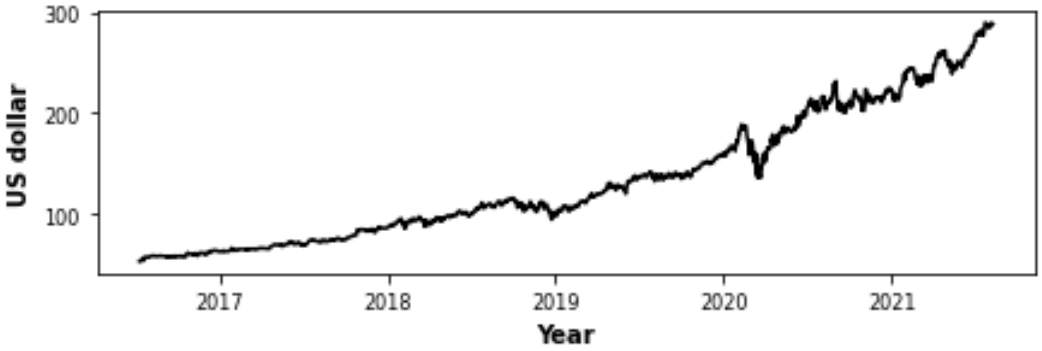
Contents

1.1 Background

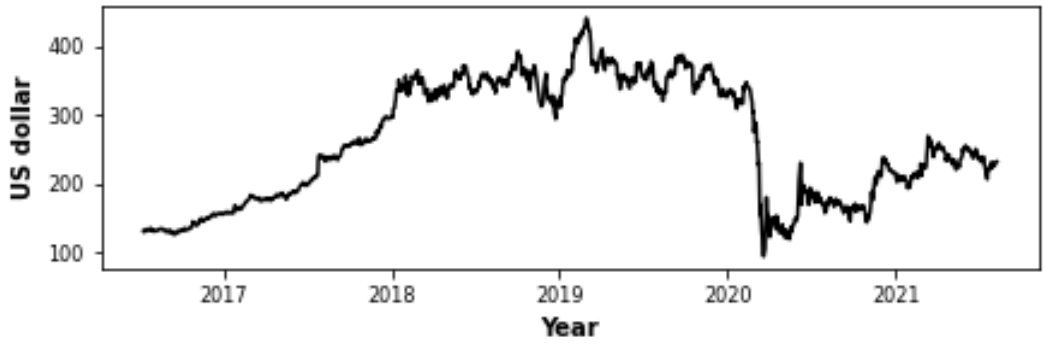
Financial markets play a significant role in the economy by allowing for the efficient allocation of resources, serving as an economic barometer, and providing individuals with the opportunity to increase their wealth [91, 119]. Modern economies rely heavily on the transactions occurring within financial markets. As a result, intensive research concerning the performance and behaviour exhibited by financial markets has been conducted over the past 50 years. Investors want to invest in an asset within a financial market with the expectation of future returns instead of a decreased monetary value. The criteria of having a future return have left investors pondering the decision of which company to invest in. The concept of predictive analysis, also known as forecasting, appeared in the financial field to address this impasse.

Forecasting of financial markets is regarded as one of the most challenging applications of forecasting, because financial data is an instance of a signal processing problem that is inherently noisy, non-linear, non-stationary, and deterministically chaotic [126]. The complexities associated with financial data resulted in some scepticism, which is reflected in the *efficient market hypothesis* (EMH) theory, developed by Fama [36]. According to the EMH theory, financial assets always trade at their fair market value, making it impossible for investors to purchase undervalued assets or to sell assets at inflated prices. Furthermore, price changes are independent of the past and unpredictable. However, numerous critics have rebutted the EMH theory with evidence that markets are not fully efficient, thereby making it possible to predict the future values of assets [51].

Consequently, prediction of financial market returns has become an important subject for individuals from academia and industry. Correct modelling of the financial market, to some extent, enables investors to determine which assets are better to invest in. For example, between 2017 and 2021 it was more profitable for an investor to invest in Microsoft than in Boeing, as illustrated in Figure 1.1. Microsoft provided a 399% return on capital, while Boeing provided a 103% return on capital, as illustrated in Figures 1.1 (a) and (b), respectively.



(a) *Microsoft share price [80]*



(b) *Boeing share price [14]*

FIGURE 1.1: *Comparison of the Microsoft and Boeing share prices for the period 2016 to 2021.*

In addition to identifying assets with a future positive return, analysts conduct many kinds of research to accomplish one goal: to beat the market. The term to beat the market is defined as the task of obtaining a rate of return that is consistently higher than the average return of the market while keeping the same level of risk [128]. An institution can only beat the market if the weighted return of all assets is higher than the return of the market. Various modelling techniques are utilised to try and model asset price movements which enable investors to beat the market. The techniques are focused on two areas of forecasting, namely *fundamental analysis* and *technical analysis* [85].

Two well-known companies, each employing a different technique, have shown it possible to beat the market, as illustrated in Figure 1.2. The market refers to the S&P 500 index, which had an average return of 12.43% from 1988 to 2021. The first company, Berkshire Hathaway managed by Warren Buffet and Charlie Munger, primarily employs fundamental analysis techniques and has obtained an average return of 18.09% over the same time period [61]. The Renaissance Technologies company, started by Jim Simmons, primarily employs technical analysis techniques and has obtained an average return of 40.18% over the same time period [21].

Fundamental analysis examines underlying forces like monetary and fiscal policy and economic indicators in an attempt to find the intrinsic value of an asset [111]. The intrinsic value provides an indication of the true value a company is worth. If an asset's price is lower than its intrinsic value, investors should buy it. On the contrary, if the value of an asset is higher than its intrinsic value, investors should sell it.

Technical analysis is based on the examination of past price movements to forecast future price movements of an asset [1]. In an attempt to forecast future price movements, technical analysis assumes that the price of an asset is a reflection of mass psychology ('the crowd') in action. Crowd psychology moves between greed, excessive optimism, and confidence on one hand and fear, pessimism and panic on the other [26]. Well known techniques that fall under technical analysis are the *moving average*, *autoregressive integrated moving average*, and more recently *artificial intelligence* (AI) techniques.

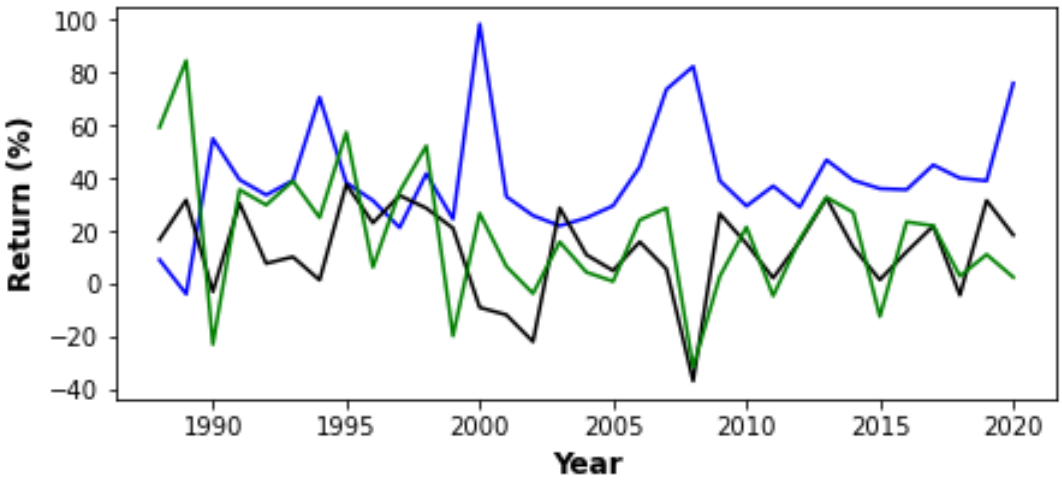


FIGURE 1.2: *Comparison of returns obtained by the S&P 500 index, Berkshire Hathaway and Renaissance Technologies from 1988-2021 [21, 61].*

Traditional statistical methods, like the moving average, are typically unsuccessful in predicting financial market data, because the data are uncertain, complex and non-linear [78]. As a result the focus shifted from linear to non-linear models which include *machine learning* (ML) techniques like *support vector machines* (SVM), *genetic algorithms* (GA) and *artificial neural networks* (ANN) [99]. The highly popularised *multilayer perceptron* (MLP) model has been successfully applied in the forecasting of financial assets [33, 104, 127]. However, MLP models are not without disadvantages: MLPs utilise computationally intensive training algorithms such as error back-propagation and become stuck in local minima [68]. Over the past few years, there have been successes achieved in forecasting financial data by employing ANNs with *deep neural architectures* [103]. More specifically, the successes were achieved with architectures that were originally designed and developed for domains such as *natural language processing* (NLP) and *image recognition*, rather than for financial data.

NLP models give a computer the ability to understand, analyse, manipulate, interpret and generate the human language (*i.e.* words). These language models are concerned with the relative order of words in a sentence, which allows them to preserve the context of a sentence from timestamp to timestamp [79]. By understanding and preserving the context of words in a sentence, NLP models are able to predict the next word in the sentence. The similarity between financial market data and sentences has led to the adoption and application of NLP models to financial market data sets. Financial market data related to time is known as *financial time series* (FTS) data, in the context of ML.

Given the vast quantity of statistical, ML and ANN models available, an important question arises with regard to which models obtain a superior performance on which type of FTS data set when forecasting future price movements. Different types of FTS data sets achieve different performance results when applied to the same ML model [18]. The difference in results is attributed to the different statistical properties exhibited by each FTS and the structure of the selected ML model. As a result, ML models that have a general capability to forecast the future price movement of any FTS can be investigated and identified. By investigating the models with superior performance on certain types of FTS data sets and identifying models with good general forecasting capabilities, investors can have a competitive advantage over their peers. In other words, they have the knowledge to identify assets with a positive future return which enables them to potentially beat the market.

Apart from the type of FTS data and ML models utilised, the data preprocessing tasks applied and the type of validation technique selected have an indirect influence on the performance of a ML model. By applying adequate data preprocessing tasks and validation techniques, investors are able to identify assets with a positive future return which enables them to potentially beat the market.

1.2 Project description

The continuous pursuit for higher yields in financial markets motivates researchers and practitioners to investigate ML models from other domains on FTS data sets, the general predictive capabilities of ML models and the types of validation techniques that can be employed to evaluate the performance of an ML model. Conducting this type of investigation enables one to increase the potential yield and beat the market.

Shallow neural networks, like MLPs, have been found to be successful in FTS forecasting over the past three decades [52]. However, over the past few years, deep neural network architectures started to emerge strongly as one of the best performing predictor classes within the ML field for FTS forecasting. More specifically, major successes were achieved with architectures originally designed and developed for the NLP domain rather than for FTS. Some of these include *long short term memory networks* (LSTMs) and *gated recurrent units* (GRUs) [103]. In addition, different types of FTS data sets influence the performance capability of a ML model. The primary aim of this dissertation is to compare and contrast the predictive capabilities of widely accepted ML models (statistical, ML and ANN models) for FTS forecasting when different types of FTS data sets are employed.

Currently, the single out-of-sample train-test split technique is the most widely applied validation technique in literature to verify how accurate a ML model is on an FTS data set. This technique is not wrong per se, but it has short comings [44]. This dissertation studies the effect of the single out-of-sample train-test split validation technique and a validation technique that is more suited for FTS forecasting. In turn, the predictive capabilities of the selected ML models are compared and contrasted based on the validation technique selected.

1.3 Project scope

In order to narrow down the scope of the problem considered in this dissertation, the following scope delimitations are adopted:

Financial time series data from different financial markets are employed in this dissertation. Different time series data is incorporated in order to avoid *selection bias* and to provide more insight into the predictive capabilities of each model in the context of different financial assets. The financial markets of interest in this dissertation are the stock market, bond market, foreign exchange market, and cryptocurrency market. For each financial market, the FTS of one asset is investigated. The assets under investigation are the S&P 500 index, the US 10 year bond yield, the USD/ZAR currency pair, gold futures and Bitcoin.

Input variables utilised to train and model an FTS are limited to the daily closing price, which is transformed into the daily close price *percentage change*, also known as the *daily return*. During modelling, daily returns for the previous two days are utilised to predict the current return.

Validation techniques employed in the analysis of the different ML models are limited to the single out-of-sample technique and the walk-forward technique, which is suited specifically for time series data.

Machine learning techniques employed in this dissertation are limited to the most pertinent and well-known statistical, ML and AI techniques for FTS forecasting, based on a thorough literature review. Only supervised learning techniques are employed to perform the comparison between models.

1.4 Objectives

The following objectives are pursued in this dissertation:

I To *conduct* a thorough survey of the literature pertaining to:

(i) The financial market in general, the different types of financial markets and associated assets, and the market forces responsible for price movements for each respective financial market,

(ii) time series and FTS data characteristics,

(iii) the investigation of guidelines in respect of the design, development, validation, and verification of models employed in data analysis and comparison, and

(iv) the development and evaluation of appropriate statistical, ML and ANN techniques capable of forecasting FTS data sets.

II To *design* an empirical procedure, based on the literature review of Objective I, for comparing the performance of different ML techniques and financial assets. This procedure should facilitate:

(i) The preprocessing and transformations of FTS data,

(ii) the development, evaluation and comparison of suitable ML techniques, and

(iii) the analysis and visualisation of model results with the objective of comparing them to each other in order to gain insights.

III To *obtain* and *prepare* different financial asset time series data sets in accordance with the empirical procedure outlined in Objective II.

IV To *evaluate* the predictive capabilities of the selected learning models described in Objective 1(iv) on the financial asset data prepared in Objective III, in fulfilment of Objective II.

V To *visualise* the model outputs of Objective IV, in fulfilment of Objective II.

VI To *compare* and *contrast* the performance evaluation metrics obtained for each ML model based on financial asset and validation technique.

VII To *recommend* sensible follow-up work pertaining to the work in this dissertation which may be pursued in the future.

1.5 Research methodology and dissertation organisation

Excluding this present introductory chapter, this dissertation contains a further six chapters which are structured as follows: The first and second chapter consists of a comprehensive literature review during which a thorough investigation of the topics stated in Objective I of Section 1.4 is conducted. More specifically, Chapter 2 is devoted to a review of the relevant literature related to financial markets, different financial assets within the financial markets, and the market forces responsible for the price movements in financial assets, in fulfilment of Objective I (i). Thereafter, the properties and characteristics of time series and FTS data sets are reviewed in pursuit of Objective I (ii) to better understand the type of data utilised in this dissertation. The focus then shifts in Chapter 3 to the literature pertaining to common ML paradigms, together with a description of the practical considerations associated with evaluating the performance of ML models within a supervised learning paradigm, in pursuit of Objective I (iii). Finally, a review of selected statistical, ML and ANN techniques capable of forecasting FTS data sets, in fulfilment of Objectives I (iv) is conducted.

Chapter 4 is concerned with the application of the literature detailed in Chapters 2 and 3 to propose an empirical procedure that can compare and contrast different ML models, as per Objective II. A high-level overview of the proposed framework is given, followed by a detailed description of its major components by means of data flow diagrams.

The following two chapters discusses the implementation of the empirical procedure outlined in Chapter 4. More specifically, in Chapter 5, Objective III is pursued by adequately preparing different financial asset TS data sets. In Chapter 6, the predictive capabilities of each ML model are evaluated and visualised in pursuit of Objectives IV and V. Furthermore, the results obtained by each ML model are appropriately assessed in pursuit of Objective VI. Thereafter, the performance evaluation metrics of each ML model are compared and contrasted based on the type of FTS and validation technique selected, in fulfilment of Objective VI.

The final part, Chapter 7, concludes with a summary, critical appraisal and contributions of this dissertation. In pursuit of Objective VII, possible improvement opportunities that may be pursued in the form of follow-up work are discussed.

CHAPTER 2

Financial time series

Contents

The aim of this chapter is to introduce the reader to financial markets in the context of forecasting future price movements of financial assets. The chapter opens with an investigation into the concept of financial markets, the different types of financial markets and the factors influencing the price movement of financial markets. Thereafter, considerations associated with the collection of financial data are discussed. Subsequently, a review of time series and FTS data is documented. The chapter closes with a summary of its contents.

2.1 Financial markets

A financial market is formally defined as a marketplace where activities related to the creation, trading and exchange of financial *assets* take place between *sellers* and *buyers* at a price determined by *market forces* [12, 34]. The assets are synonymous with products, while sellers and buyers refer to people, private and public companies that are collectively known as *investors*. In economics, the term market is defined as a meeting of buyers and sellers for a particular good or service and their aggregate business transactions in their own free will [119]. A financial market comprises multiple markets that can be distinguished by identifying the traded assets.

Financial markets are common to each country and play a prominent role in the economic growth and efficient allocation of resources within a country [12, 51]. For example, financial markets provide businesses and government entities with the opportunity to raise money for future business endeavours or expansions, they enable individuals to generate wealth, and they provide employment to thousands of individuals [91]. Therefore, prediction of the future movement of financial markets has become the top choice of computational intelligence for finance researchers from both industry and academia. However, an understanding of the different types of markets found within the financial market is required to successfully generate a forecast. In addition, one should understand how various *market forces* influence the price movement of each respective market. The different types of markets and their respective market forces are discussed in the remainder of this section.

2.1.1 Types of financial markets

A financial market comprises numerous individual markets that are either local, regional or international, and varying in size. The individual markets are interconnected with one another and therefore influence each other. For example, if market A suffers, market B will have a positive return and market C will have a negative return for the day. The most popular financial markets are the *capital* market, *money* market, *over-the-counter* market, *foreign exchange* market, *commodity* market, *derivatives* market, and *cryptocurrency* market [12]. The concept of a financial market and the popular markets it is comprised of is illustrated in Figure 2.1.

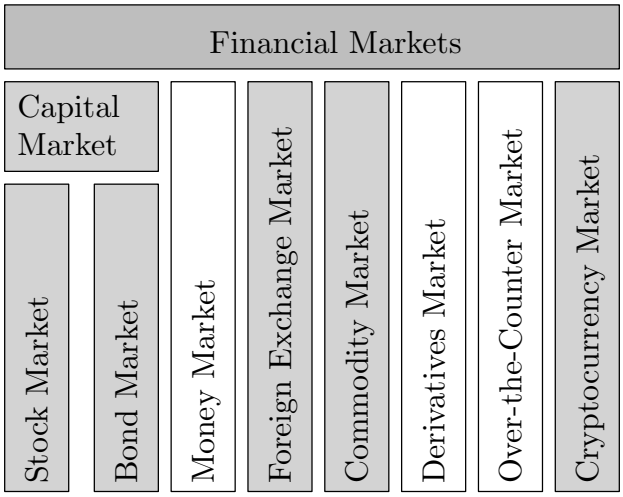


FIGURE 2.1: *Popular types of financial markets [12].*

Individual markets are characterised based on the properties of the financial assets traded and the needs of the different market participants. The markets of particular interest in this dissertation are the capital, foreign exchange, commodity and cryptocurrency markets. These markets are defined as follows:

Capital market: The capital market is comprised of two markets, namely the *stock* market and the *bond* market, as illustrated in Figure 2.1. The stock market refers to a market where regular activities of issuing, buying and selling of *shares* of publicly-held companies, also known as *stocks*, take place [12]. A share is a financial instrument that represents fractional ownership in a company and gives a buyer a proportionate claim to the company's assets and earnings.

The stock market is comprised of multiple stocks. An *index* of the market is a smaller sample of the stock market that is representative of the whole. Essentially, an index is a collective grouping of stocks by industry or market capitalisation. The purpose of an index is to provide investors with a consolidated view of how the market or industry sector is performing [51]. The consolidated view helps investors to identify sectors within the market that are under-or over- performing and consequently identify possible investment opportunities. Upon identification of the underperforming industry index, investors may either invest funds in a specific stock or an *index fund*. An index fund is a financial asset designed to *track* an index. An index fund has the same properties as a share (*i.e.* it may be bought and sold), with the exception of providing exposure to an entire industry instead of one company within the chosen industry.

The bond market, also known as the *debt market*, is the collective name given to all trades and issues of debt securities [51]. This market provides an opportunity for governments and corporates to raise capital to pay down debt or fund new projects and investments. In a bond market, investors

buy bonds from governments and corporates which will pay the monetary value of the bond with interest on the agreed maturity date. Before reaching maturity, bonds are freely traded in the bond market. Currently, government issued bonds are by far the most liquid and active bond market [12].

Foreign exchange market: The foreign exchange market, also known as *Forex*, is a global marketplace that determines the exchange rate for currencies around the world [96]. Forex does not set the absolute value of a currency but rather its relative value by setting the market price of one currency if paid for with another [12]. Consequently, currencies are traded in pairs (*e.g.* USD/ZAR, CHF/EUR or EUR/ZAR). For example, on 1 May 2021, 1 US dollar was worth 14 South African Rands, 1 Swiss Franc was worth 0.91 Euros, and 1 Euro was worth 17 South African Rands. Participants in the Forex can buy, sell, exchange and speculate on the relative exchange rates of various currency pairs.

Commodity market: The commodity market is a marketplace where the trading of raw materials take place. Commodities are split into two broad categories, namely soft and hard commodities. Soft commodities refer to livestock or agricultural products such as pork, cattle, sugar, corn and coffee. Hard commodities refer to natural resources that are mined or extracted such as gold, silver, and oil [12].

Cryptocurrency market: Cryptocurrency is a digital asset and a form of decentralised currency that employs *blockchain technology*. Blockchain is defined as a peer-to-peer decentralized distributed ledger technology that makes the records of any digital asset transparent and unchangeable [20]. This technology can work without involving any third-party intermediary. Activities related to cryptocurrencies include *mining*, *trading* on a *crypto exchange*, and using it as a form of clandestine payment for products or services [38]. Cryptocurrency mining is the process of mining coins by solving complex numerical puzzles. Crypto exchanges, also known as *digital currency exchanges*, operate similarly to the financial markets outlined above, in that they allow for the buying and selling of assets [5]. Trading occurs on popular crypto exchanges such as Binance, Bitstamp and Coinbase [67].

2.1.2 Influence of market forces on financial assets

A market force is defined as a factor that has an ability to affect change in a market and occurs either naturally or as a result of government intervention [119]. Market forces influence all financial assets. However, some forces have a larger impact than others. Market forces vary based on the type of financial market and the asset itself, and therefore, numerous assets with associated market forces exist within each financial market. This dissertation, however, will only document the market forces of one chosen asset from each of the financial markets of interest. The financial markets of interest are shaded in grey in Figure 2.1.

The assets under investigation have been identified as either the most liquid, active, popular, or well known within each financial market. The assets of interest are the *S&P 500 index*, the *US 10 year bond yield*, the *USD/ZAR* currency pair, *gold futures* and *Bitcoin*.

The S&P 500 index is one of the most widely followed and often quoted indexes in the world [74]. The index is comprised of the 500 largest companies by market capitalisation in the United States and is an asset in the stock market.

The US 10 year bond yield is an asset traded in the bond market and refers to the 10 year US treasury bond yield. The treasury yield serves as an indicator of broader investor confidence and is often utilised as a proxy for financial matters such as mortgage rates [51]. Most analysts employ the 10

year yield as the "risk free" rate when valuing the markets or individual stock [60]. The US 10-YR Yield will be referred to as the bond yield in the remainder of this dissertation.

The USD/ZAR currency pair in the Forex market is of particular interest from a South African perspective, because the US dollar is the most traded and largest reserve currency in the world [96].

Gold is among the most valued commodities found in the commodity market. Gold has a history of utility in currency and jewellery as well as being a favoured safe haven asset [30, 83]. The gold futures will be referred to as gold in the remainder of this dissertation.

Bitcoin is the first blockchain-based cryptocurrency that was launched in 2009. Bitcoin is the most popular and valuable cryptocurrency in the cryptocurrency market, with an approximate market capitalisation of $927 billion [5].

The most influential market forces on each financial asset are documented in Table 2.1. The table documents the data format and how regularly the data for each asset is available and collected. The data format can be quantitative, denoted by A, or qualitative, denoted by B. The regularity of the data refers to how often new data is available for the specific force and, in this case, may be either in days, months, years or not at regular intervals (NARI).

Certain market forces that do not occur daily have a substantial impact on the price movement of an asset [34]. The forces include, but are not limited to, interest rates, inflation rates, monetary and fiscal policy, political stability and economic performance [119]. The impact of irregular occurring market forces documented for each asset in Table 2.1 is described as follows: For the S&P 500 index, monetary policies set by the Federal Reserve affect the cost of borrowing and, in turn, spending and investment by businesses and consumers. Consequently, a good (bad) monetary policy may have a positive (negative) impact on the S&P 500 index value. A positive economic performance, indicated by economic growth and high employment, will result in stocks increasing in value [8, 51]. As a result, the S&P 500 index will also increase in value. In addition, financial crises, natural disasters, elections, and government policies have a substantial positive or negative effect on the S&P 500 index.

The bond yield will experience a positive increase in yield if a low interest rate and a high inflation rate is observed [32, 12]. If the US government's fiscal policy is to increasingly borrow money to service their debt, the yield increases. However, if the government's monetary policy is to reduce the supply of short term securities in open market operations, the bond yield decreases.

From the perspective of South Africa, the price movement of the USD/ZAR currency pair in the Forex market will depreciate in value if South African inflation rates are higher than US inflation rates (*i.e.* the Rand becomes weaker). Inflation and interest rates are negatively correlated, and therefore a high inflation rate leads to a low interest rate, which in turn will further depreciate the value of the Rand [9, 29]. An increase in public and government debts will result in a depreciation of the Rand. The current account of a country reflects the balance between the value of imports and exports. If there are more exports than imports, South Africa will experience a current account surplus resulting in the appreciation of the Rand. Lastly, a stable political state in South Africa with little or no risk for political turmoil or an unstable political state in the USA tends to lead to an appreciation in the value of the Rand.

Gold in the commodity market will experience an increase in value if inflation rates are high, interest rates are low, or there is economic uncertainty [35, 97]. In the cryptocurrency market, Bitcoin may experience an increase in value if user adoption increases. Therefore, the popularity of the currency can impact the price movement of Bitcoin. Bitcoin reward halving refers to the practise where the reward for Bitcoin mining halves when 210 000 blocks are generated [5]. On average, this phenomenon occurs every four years. Regulations or responses from regulatory bodies in prominent countries can cause the value of Bitcoin to increase or decrease.

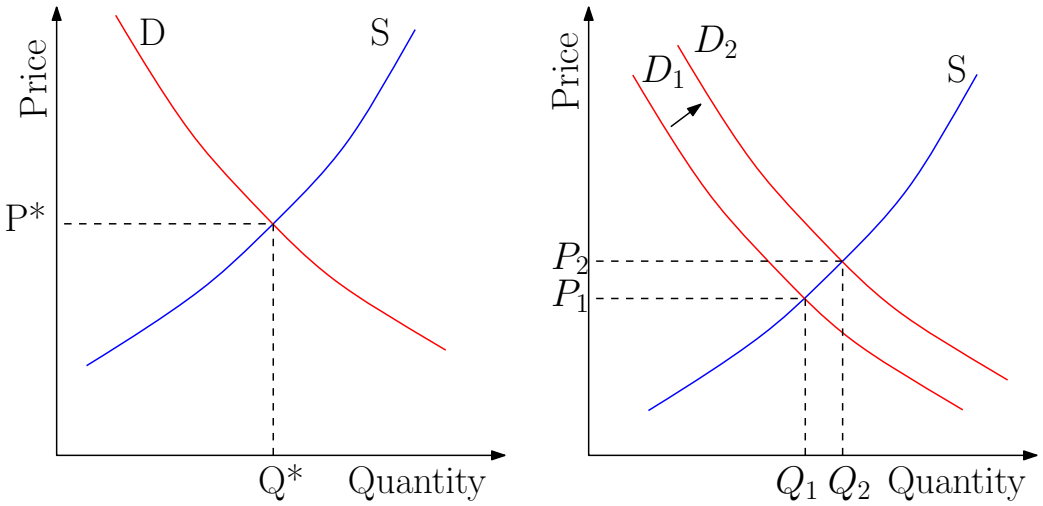
TABLE 2.1: *Major forces that influence price movements in financial assets [3, 8, 35, 60, 114]. A and B denotes quantitative and qualitative data types, and NARI indicates that the data is not collected at regular intervals.*

Market forces that influence financial products					
Financial Market	**Product**	**Indicator**	**Forces**	**Format of data**	**Regularity**
Stock Market	S&P 500 index	Supply and demand	Monetary policy	A & B	NARI
			Economic performance	A & B	NARI
			Currency valuations	A	Daily
			Commodity prices	A	Daily
			Financial crises, natural disasters, elections and government policies	B	NARI
Bond Market	US 10-YR Yield	Supply and demand	Interest rate	A	NARI*
			Inflation rate	A	Monthly
			Fiscal policy	A & B	NARI
			Monetary policy	A & B	NARI
Forex Market	USD/ZAR	Supply and demand	Interest rate	A	NARI
			Inflation rate	A	Monthly
			Public debts	A	NARI
			Current account	A	Quarterly
			Political stability	B	NARI
Commodity Market	Gold	Supply and demand	Interest rate	A	NARI
			Inflation rate	A	Monthly
			Uncertainty	B	NARI
			Currency Movement USD/EUR	A	Daily
			Oil price	A	Daily
			Silver price	A	Daily
Cryptocurrency Market	Bitcoin	Supply and demand	User adoption	A & B	NARI
			Bitcoin reward halving	A & B	Every 4 years
			Cryptocurrency regulations	A & B	NARI

Table 2.1 indicates that the assets are influenced by a majority of market forces that do not occur daily or at regular intervals. Irregular occuring data makes the successfull forecasting of the future price movement of financial assets over a short period (*i.e.* days or weeks) considerably more difficult [46]. However, the outlined market forces ultimately influence the *supply and demand* for each respective asset. Supply and demand is a fundamental economic principle and a big driver of price movement within the financial market [12]. Supply and demand data is available daily, and consequently forecasting the future price movement of financial assets over a short period successfully is considerably easier to some extent. The price movements of financial assets are recorded in certain time frequencies and are therefore referred to as *time series* (TS) data.

Principle of supply and demand

The principle of supply and demand is influenced by two factors, namely the price of the product denoted by P^* and the quantity that is available denoted by Q^*. The supply and demand curve, illustrated in Figure 2.2, shows the relationship between supply (provided by the seller) and demand (provided by the buyer). If a buyer and seller agree on a price for a product, a price equilibrium is reached and a transaction occurs. This concept is illustrated in Figure 2.2(a). When an equilibrium is reached, the supply curve (blue line) intersects with the demand curve (red line). In this case, a buyer purchases Q^* products at a price P^* from a seller.



(a) *Market equilibrium is reached.*

(b) *Over demand in the market.*

(c) *Over supply in the market.*

FIGURE 2.2: *Supply and demand curves [12].*

An increase in demand from Q_1 to Q_2 shifts the demand curve to the right, as illustrated in Figure 2.2(b). A situation has been created where there are more buyers than sellers and as a result, the price increases from P_1 to P_2 to address the shortage of available products. An increase in supply from Q_1 to Q_2 shifts the supply curve to the right, as illustrated in Figure 2.2(c). In this case, a situation has been created where there is an oversupply of products and a shortage of buyers, and as a result, the price decreases from P_2 to P_1.

2.2 Data Collection

In this section, the market forces and indicators outlined in Table 2.1 are discussed in terms of *where* the data of interest is collected for forecasting purposes. Data concerned with market forces like monetary and fiscal policy, interest and inflation rates, economic performance, public debts and current accounts are primarily found on economic websites like MarketWatch. Market force data that are only qualitative like political stability, uncertainty and natural disasters, to name a few, are recorded in the form of broadcasted news, articles or interviews.

The market force data discussed thus far in this section, however, has a low and differing *granularity*, making it somewhat difficult to employ with higher granularity data. Granularity refers to the level of detail of the data within a data structure [94]. A *high* granularity of data refers to data that is extremely detailed in terms of how it is collected. On the contrary, *low* granularity data refers to data that is not collected regularly. For example, data collected every second, minute or day is of high granularity, while data collected monthly or yearly is of a low granularity.

In the context of Table 2.1, high granularity data refers to market forces that are, for example, available daily like currency valuations or commodity prices. Although the task of combining low and high granularity data for forecasting is complex, it is possible [95]. However, for this dissertation, the market forces for each product (which are primarily of a low granularity) will not be employed.

The supply and demand data generated daily for each respective product is of a high granularity and will therefore be utilised as the main data source for this dissertation. The open price, close price, high price, low price and volume data are typical indicators employed when collecting supply and demand data. Supply and demand data is readily available on numerous financial websites such as Investing.com, MSN money or Google Finance. Only the close price data is employed for the modelling and forecasting of future asset prices in this dissertation. Although this is a very simplistic approach, the objective of this dissertation is to create a level field where different types of models with their respective capabilities, properties, advantages and disadvantages can be compared.

2.3 Financial time series

In mathematics, a TS is defined as a chronologically ordered collection of measurements or observations taken at regular intervals, for instance, hours, days, months or years [71]. Examples of TS data include daily heart rate monitoring, monthly unemployment rates, or yearly global temperatures, to name but a few. TS data related specifically to the sequence of prices of some financial asset over a specific period are collectively known as FTS data [117]. This type of TS data is the focus of this dissertation. FTS data is a specialised form of TS data, and therefore has the same data characteristics as TS data. However, in comparison with other types of data, TS data has several characteristics that make their analysis and prediction different.

TS data is different from other data types in that it exhibits one or more of the following characteristics: a *trend*, *seasonality*, *cyclical* behaviour, and/or *irregularity* [46]. In the context of TS data, a trend is defined as a gradual upward or downward shift in the level of the series. A trend can increase or decrease over time. Seasonality refers to the regular and predictable changes that recur in regular calendar intervals such as months or fiscal years. Seasonality is always of a fixed and known period. For example, the winter clothing sales increase in the winter compared to other seasons and ice-creams sales increase in the summer when the weather is warm. Cyclic behaviour refers to a repetitive, non-fixed and unpredictable pattern in the TS data points. The duration of these fluctuations, known as the *period*, is of at least two years. Irregularity, also known as *white noise*, is the residual TS after the trend, seasonality, and cyclical characteristics have been removed. It is a random variation that any other factor cannot explain.

A TS may be deconstructed to depict the individual characteristics, as illustrated in Figure 2.3. In this figure, the TS data set under investigation is depicted in the rectangular block labelled 'Observed'. The decomposed TS indicates that the observed TS contains three of the possible four characteristics: trend, seasonality, and white noise.

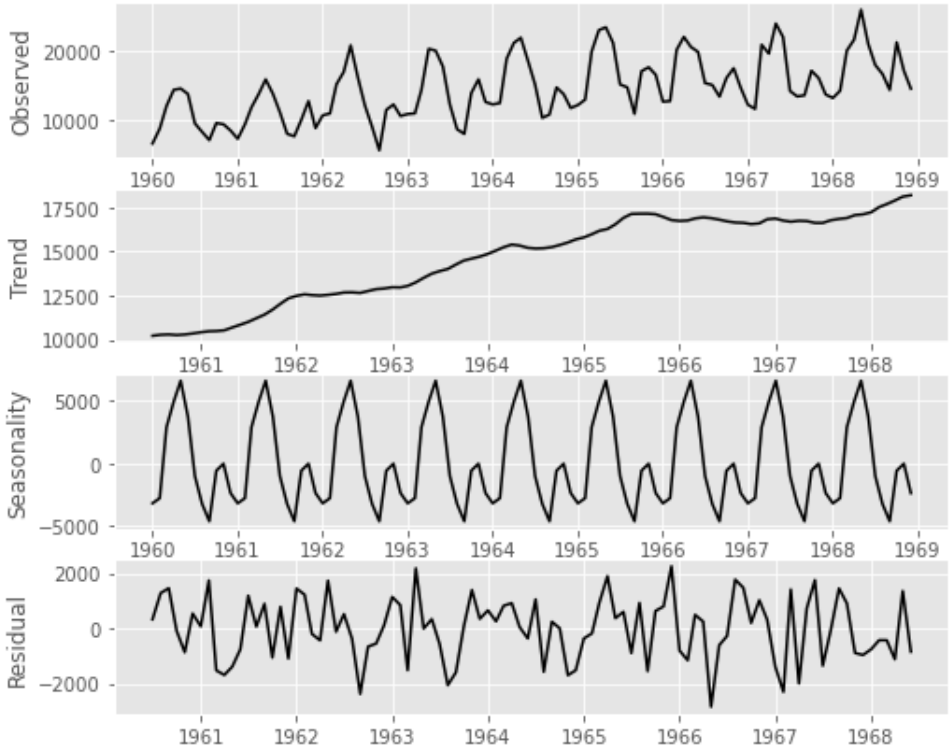


FIGURE 2.3: *Deconstructed time series [18].*

The rectangular block labelled 'Trend' indicates that the TS trend is positively increasing from 1948 to 1958. The third block labelled 'Seasonality' shows that the TS follows the same pattern on an annual basis. There is a sharp increase at the beginning of each year, a sharp decrease in the middle of the year followed by a slight increase. Finally, the block labelled 'Random' indicates short-term fluctuations in the TS that are not predictable.

2.4 Chapter summary

The aim of this chapter was to introduce the reader to financial markets in the context of forecasting future price movements of financial assets. The chapter opened with an investigation into the concept of financial markets and why they are necessary. Furthermore, the most popular types of financial markets were outlined. The markets of particular interest in this dissertation, namely the capital, foreign exchange, commodity, and cryptocurrency, were reviewed. Subsequently, a financial asset for each respective financial market was chosen based on liquidity and popularity. The assets chosen were the S&P 500 index, bond yield, USD/ZAR currency pair, gold and Bitcoin. The market forces that have a dominant influence on the price movement for each respective asset were investigated in the next subsection.

Thereafter considerations associated with the collection of financial data were discussed. In the last section, the focus shifted to TS and FTS data. More specifically, the properties and characteristics of typical TS data sets were documented and visualised.

CHAPTER 3

Machine learning

Contents

Arthur Samuel, a pioneer in the field of AI, was the first to coin the term 'machine learning' upon the publication of his program, which could successfully play the game of Checkers by 'learning' from its past experiences playing the game. He defined ML as '*the field of study that gives computers the ability to learn, and even to improve themselves, without being explicitly programmed*' [84]. According to the widely adopted definition by Tom Mitchell [82], "[a] computer program is said to *learn* from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."

In the case of Arthur Samuel's Checkers playing program, a computer program learns to *play* Checkers (T), and may improve its performance as measured by its *ability to win* (P) through experience acquired by *playing games against itself* (E) [82]. Alternatively, within the realm of TS forecasting one may construct a program that learns to *forecast the future values of a given TS data set* (T) by analysing and identifying patterns from *past instances within the TS data set* (E) with the aim of accurately forecasting the *future values in the TS* (P).

An overview of the various types of ML paradigms is given in this chapter. An outline of the typical training procedures and evaluation metrics employed in this field follows. Subsequently, selected ML algorithms deemed relevant for understanding the remainder of the material in this dissertation are described. The notion of neural network algorithms is introduced and reviewed separately due to its additional complexity.

3.1 Types of machine learning

ML algorithms are broadly classified into four paradigms namely, *supervised*, *unsupervised*, *reinforcement* learning, and *evolutionary* learning algorithms.

The supervised learning paradigm refers to the situation in which, for each observation $i \in \{1, ..., n\}$ of *predictor measurement(s)* $\boldsymbol{x_i}$, there is an associated response measurement $\boldsymbol{y_i}$ [55, 101]. In this paradigm, a model is fit to ensure that the response is related to the predictors, with the objective of better understanding the inference (relationship between the response and the predictors) and accurately predicting the response for future observations. The utilised data is known as *labelled data*, because each data point has a label corresponding to the desired output. Given a weather data set with inputs temperature, humidity, and wind speed, each record within the data set is labelled as either sunny or cloudy. The algorithm learns by predicting a label (sunny or cloudy) for each training observation, comparing the predictions to the ground truth labels and adjusting the parameters in the algorithm accordingly. The labels (sunny or cloudy) of additional unlabelled data can then be predicted by the model.

In contrast, the unsupervised learning paradigm refers to the situation in which, for each observation $i \in \{1, ..., n\}$ of vector measurements $\boldsymbol{x_i}$ there is no associated response $\boldsymbol{y_i}$. The data is referred to as *unlabelled data*, because a response variable is not present. The aim of the analysis is to identify underlying structures within the data. *Clustering* is an example of this analysis, where similar data points are grouped together based on their features [48].

Situations in which labels are available for some, but not all, of the observations are addressed by learning algorithms referred to as *semi-supervised* methods. To improve the performance of purely supervised methods [90], these methods are tailored to employ both unlabelled and labelled data. Intuitively, information with regard to the distribution or structure of the data set is extracted from all available observations in an unsupervised manner and the relationship between the inputs and the class labels (outputs) is investigated by means of a supervised learning component [57].

The reinforcement learning paradigm is modelled closely on the learning process of animals. Reinforcement learning is comprised of three important components: The *agent* (decision maker), the *environment* (everything the agent interacts with), and *actions* (what the agent can do) [56]. The agent chooses an action to perform in response to the current state of its environment. Once the action is performed, the agent receives feedback as to how good this decision was in the form of a *reward*, allowing it to adjust its strategy, or *policy*, accordingly [57]. The process is repeated until an acceptable policy has been learned by the agent. Stated differently, the agent is tasked to determine, through a trial-and-error process, the correct set of actions it should implement to maximise (or minimise) some notion of cumulative reward (or punishment). Typical applications of reinforcement learning include control systems, robotics, and gaming.

Each of the above mentioned paradigms has challenges that should be handled in order to efficiently utilise ML techniques. AI search and optimisation algorithms often solve the challenges, which form part of the evolutionary learning paradigm. Common challenges that are addressed include training classifiers and predictors to maximum accuracy, parameter tuning, the minimisation of multi-objective problems, under-and over-fitting, and feature selection [81]. Optimisation algorithms typically employed in this paradigm are as follow: *particle swarm* optimisation [45], the *Harris hawks* optimiser [115], the *grey wolf* optimiser [112], and evolutionary search [88].

3.2 Model training and evaluation

The source of experience for a ML algorithm takes the form of a data set, as indicated previously. These data is split into three subsets, namely a *training*, a *validation*, and *test* data set [55]. Each data set serves a different purpose, as shown in Figure 3.1. The training data are employed to *train* the model according to the chosen ML algorithm in order to determine good *model parameters*. In addition to parameters, most ML algorithms also have a set of *hyperparameters* whose values are randomly initialised before training begins [57]. The validation data set is employed in an attempt to maximise the measured learning performance through iteratively adjusting the values of hyperparameters for a chosen ML algorithm. Lastly, the test data is employed to evaluate the final performance of the model [108].

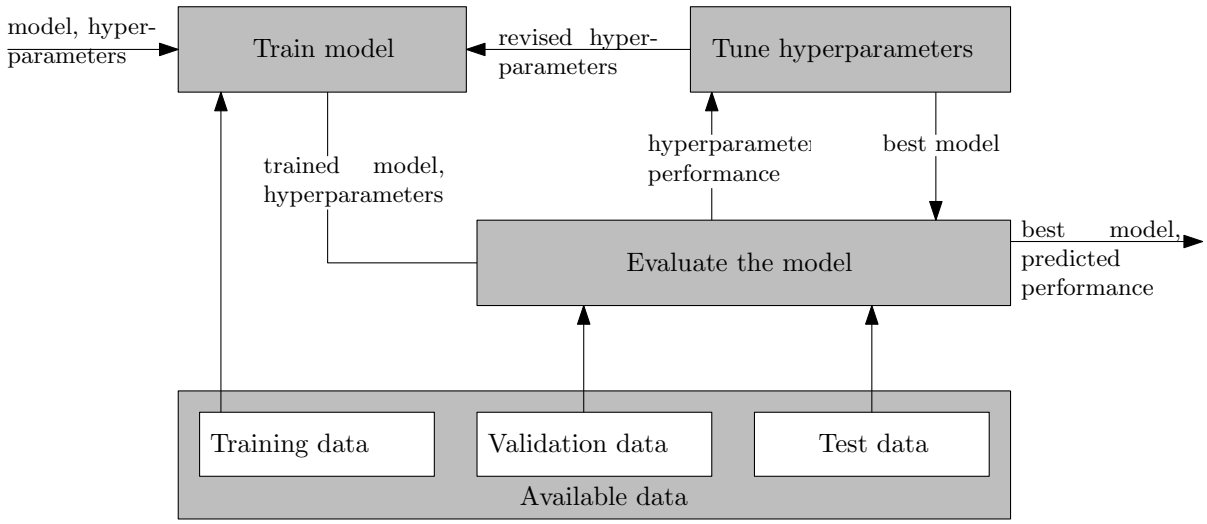


FIGURE 3.1: *The role played by the training, validation and testing data sets [57].*

The process of *hyperparameter tuning* and the generalisability of a ML model with respect to the validation and test data is explained in the remainder of this section. Subsequently, several key metrics regarding the performance evaluation of ML models are introduced. The focus in these sections, and the remainder of this chapter, is on the supervised learning paradigm.

3.2.1 Hyperparameter tuning

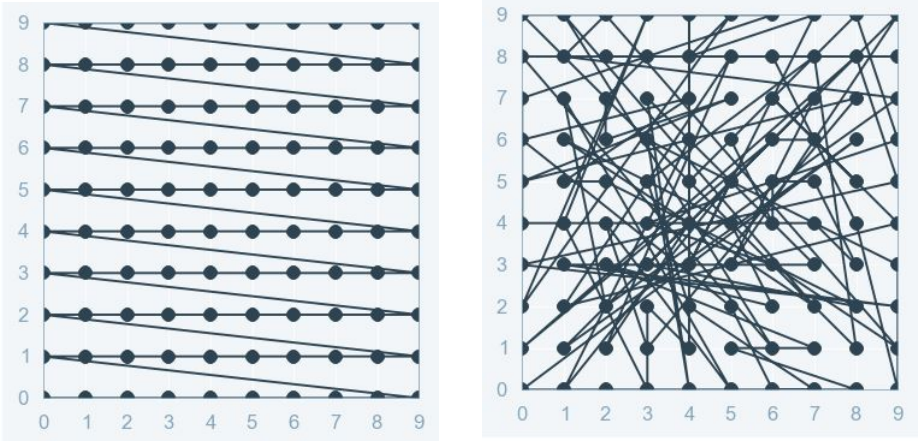
The choice of hyperparameter values can have an influence on the configuration of a model and consequently the performance of a ML model after evaluation. Sophisticated optimisation methods for hyperparameter tuning include *metaheuristics*. *Metaheuristics* are high-level problem-independent procedures that guide an underlying, more problem specific heuristic in an effort to increase performance [57, 110]. A *heuristic* is an approach used to solve a problem by trial and error when an exact algorithmic approach is impractical [124].

However, the most common approaches to hyperparameter tuning are *manual search*, *grid search*, *random search* [28], *Bayesian optimisation* [109] and the *F-race algorithm* [116].

Manual search: This approach entails the manual selection of hyperparameters which are employed to train the model in respect of the training data [55]. The performance of the model is then evaluated with the validation data set, and the hyperparameters are adjusted to increase model performance,

based on generally accepted heuristic guidelines and the experience of the modeller. The process is repeated until a satisfactory level of performance is achieved.

Grid search: This approach follows a *uniform searching pattern* by considering and evaluating every combination of hyperparameters. The search pattern followed to find the optimal configuration is in the form of a grid, as illustrated for a two parameter case in Figure 3.2 (a). Once all combinations are evaluated, the model with the best performance in respect of the validation data set is considered the best performing model. Although easy to implement, grid search suffers from the curse of dimensionality, meaning that it becomes increasingly computationally expensive as the number of hyperparameters increases [11]. Consequently, analysts prefer to employ the random search method if the data set under investigation has a high dimensionality.



(a) *The uniform searching pattern followed by grid search*

(b) *The non-uniform searching pattern followed by random search*

FIGURE 3.2: *Methods to optimise hyperparameters of ML algorithms [11].*

Random search: This approach considers and evaluates random combinations of hyperparameter values in order to find the best solution for the model, as illustrated in Figure 3.2 (b). Random search is more computationally efficient compared to grid search, and the chances of finding the optimal configuration of hyperparameters in a shorter period is comparatively higher. However, a primary drawback of the random search method is that it typically yields a high variance during computation since the selection of parameters is entirely random. To mitigate the effects of this drawback, several random search iterations should ideally be implemented to ensure that good hyperparameters are found.

Bayesian optimisation: This approach is a sequential model-based optimisation technique that considers the performance of configured hyperparameters from previous iterations to decide on the next hyperparameter value candidates. Instead of performing multiple *independent* experiments (as done in grid search and random search), Bayesian optimisation advocates for the use of intelligence to pick the next configuration of hyperparameters which improves the model performance [122]. The process is repeated until the calculated performance converges to an optimum. By selecting the hyperparameters in an *informed manner*, Bayesian optimisation has been found to be more computationally efficient than both grid search and random search.

F-race algotihm: This algorithm is a special case of the *iterated racing* algorithm that utilises Friedman's non-parametric two-way analysis of variance by ranks [75]. The F-race algorithm samples new configurations according to a particular distribution and then selects the best configuration

of the newly sampled ones by means of *racing*. Finally, the sampling distributions are updated in order to bias the sampling towards the best configurations. The approach is repeated until a termination criterion is met.

3.2.2 Generalisability and the bias-variance trade-off

The simplest model evaluation technique is the *single out-of-sample test*. To evaluate a model with this technique, the data set is split into a training, validation and testing set as illustrated in Figure 3.3. The training and validation set are utilised to determine the optimal parameter settings of a ML model. However, the evaluation of a model becomes more biased as hyperparameters are tuned to perform well on the validation set. Consequently, to ensure an unbiased performance evaluation, the model's performance is evaluated with respect to the independent test set [108]. Generally, the ratio split for the data set in the training validation and test set is 70:20:10. However, this ratio can be adjusted according to preference. Within the context of TS forecasting, this technique is problematic, because only a fraction of the data is employed to generate an out-of-sample validation and the ML model is implicitly biased towards the most recent period [44].

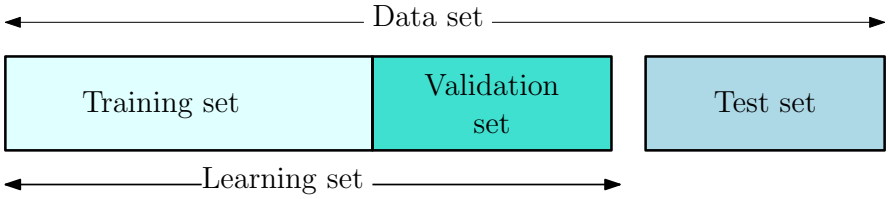


FIGURE 3.3: *single out-of-sample evaluation technique.*

One of the most widely applied procedures for model evaluation is *k-fold cross-validation*. In *k-fold cross validation* the training data are partitioned into k subsets, or *folds*, and k training iterations are performed. However, in the case of TS forecasting, k-fold cross validation is non-trivial, because it does not preserve the temporal dependency between observations [44].

Hyndman introduced a more sophisticated version of cross-validation which solves the issues related to TS forecasting [98]. The approach introduced by Hyndman is similar to the validation approach developed by Pardo. Pardo introduced a validation technique specifically for TS, known as *walk-forward validation* [76]. The walk-forward validation approach simulates models in a "walk-forward" sequence, periodically retraining the model to incorporate all data available at that point in time [64]. The training set therefore only consists of observations that occurred *prior* to the observations that form the validation set [58]. Walk-forward validation is considered a robust approach if not the best in TS modelling. The technique provides a realistic view of how a model performs over time and leverages the available data to a higher degree. The walk-forward technique enables ML models to re-parametrise periodically as new data is received and the time changes. The regular shifting of parameters enables the ML model to become robust and a clearer picture of how the entire modelling approach performs through various time cycles is obtained [93].

The cross-validation techniques introduced by Hyndman for TS evaluation are *split cross-validation* and *blocked cross-validation* [98]. The validation technique introduced by Pardo is simply referred to as walk-forward validation. All three techniques divide the training set into three folds at each iteration with the condition that the validation and testing data sets are always ahead of the training data set, as illustrated in Figure 3.4. In each figure, the horizontal axis is indicative of the data set size over time t, and the vertical axis represents the iteration k.

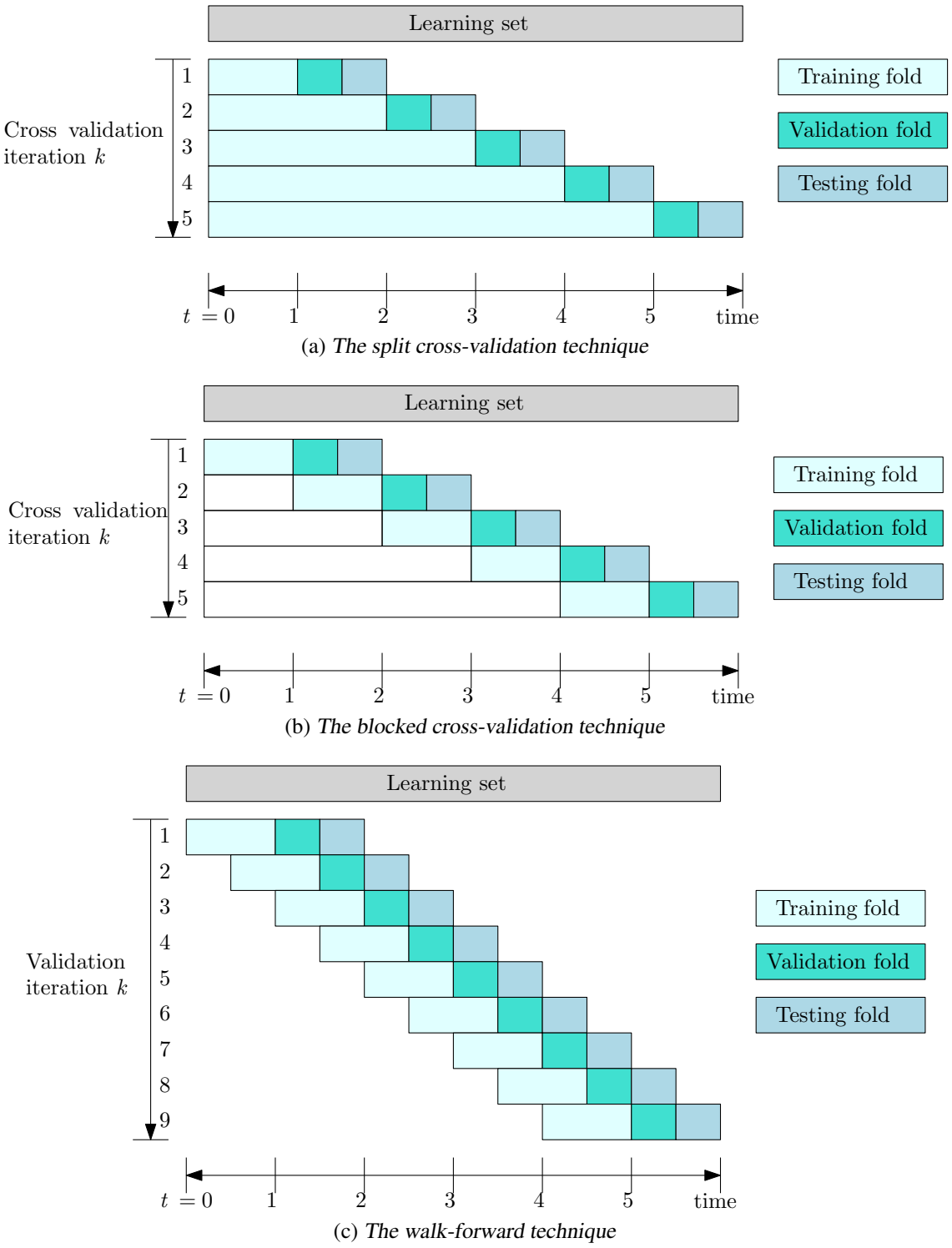


FIGURE 3.4: *Walk-forward validation techniques.*

For the split cross-validation technique the size of the training data set increases with each iteration k and the validation and test data sets moves one period ahead, as illustrated in Figure 3.4 (a). The process is repeated until all the iterations (in this case 5) have been completed. The performance of the ML model is then computed by averaging over the k validation set performances. A disadvantage of this technique is that a large data set will cause the ML model to become biased towards the most recent period. A large data set will therefore experience the same problem as the single out-of-sample validation technique.

In contrast to split cross-validation, the blocked cross-validation technique does not increase the size of the training data set with each iteration *k*, as illustrated in Figure 3.4 (b). Instead, the technique requires the training set size to be constant, while the validation and testing data sets are still one period ahead. Consequently, as the number of iterations *k* increases over time, the training data set employed in $k-1$ is disregarded.

Pardo's walk-forward validation technique is illustrated in Figure 3.4 (c). This technique is very similar to the blocked cross-validation technique, because the training set size is the same. The significant difference between the two techniques is the time periods for the training, validation and testing sets. As mentioned earlier, blocked cross-validation disregards the training data utilised in iteration $k-1$ for iteration k. Therefore, the training set in each iteration is completely new and has never been utilised as a training data set before. With respect to the validation data set, a half a time period gap is observed between consecutive iterations. The same phenomenon is observed for the testing data set.

On the contrary, an overlap is observed between the training data sets of consecutive iterations for the walk-forward validation technique. In other words, iteration k employs half the training data employed in iteration $k-1$. Furthermore, no time gap is observed between the validation and testing data sets, respectively. In reusing training data and forecasting every period for the testing data set, the walk-forward technique can validate the ML model nine times (*i.e.* $k=9$) for the same data set size. More iterations ensure that the model is robust.

The bias-variance trade-off

In ML, one of the most important concepts to consider is the notion of *underfitting* and *overfitting*. Suppose that a model performs exceptionally well in respect of the training data set, but poorly in respect of the validation data set. In this case, it is likely overfitting the data, as depicted by a model with complexity "C" in Figure 3.5. Overfitting occurs when the model is strictly fitted to the data by adding model complexity (*e.g.* instead of using a linear function, a higher-order function is used). The additional complexity leads to a high variance and low bias, which attempts to account for random variation in addition to expected variation in the data [113]. Instead of identifying the underlying trend, the model is simply memorising the observations in the training set.

As a consequence the model has a poor generalisation ability when presented with the validation data set. A simpler model that does not fit the training data perfectly results in a lower variance and higher bias [22]. Such a model performs poorly in respect of the training data set, but possesses the ability to predict future values more accurately. If, however, the chosen model is too simple, it underfits the data and consequently is unable to adequately explain the variance in the data, as depicted in the model with a complexity "A" in Figure 3.5 [57, 106]. To rectify the problem of underfitting and overfitting the data, a trade-off between bias and variance must be made, which allows the model to better identify the underlying trend in the training set, as depicted by "B" in Figure 3.5. The ideal model would therefore have a low bias and low variance.

The observed model performance gives an indication of a the generalisability of a model. However, the utilised data cannot be viewed strictly as unseen data since it was employed to adjust the values of hyperparameters during model development. Therefore, to measure model performance more accurately, the performance is evaluated in respect of the test data after training has been completed [57, 106]. This type of evaluation is done in the hope that it resembles the performance of the model upon deployment.

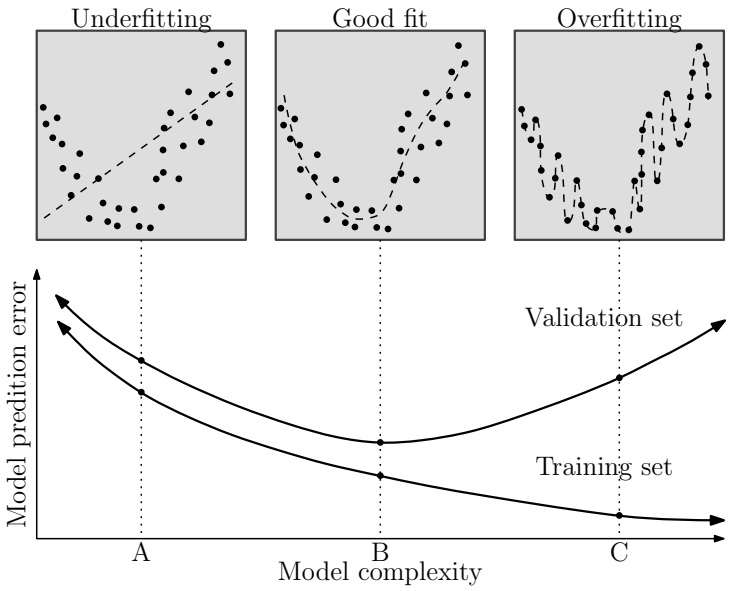


FIGURE 3.5: *The trade-off between an overfitting and underfitting a model [120].*

3.2.3 Evaluation metrics

Performance evaluation of a model is an essential part of any ML investigation. Different error metrics have different strengths and weaknesses and are therefore combined to provide an overall performance measure of the model. The *residual* and *forecast* error serve as the basis for all evaluation metrics. The residual error is more commonly known as the *training* error, while the forecast error is also known as the *testing error*. However, the formulas for both residual and forecasting errors are the same, and therefore they are collectively referred to as *errors* [15]. The error of a TS model is formally defined as the difference between the actual value y_i and the predicted value $\hat{y}_i$.

The most important evaluation metrics employed to measure prediction errors of TS forecasting models are as follows:

Mean absolute error: The *mean absolute error* (MAE) is a metric that is employed to measure the arithmetic average of *deviations* between predictions and actual values [53]. The positive and negative deviations from the actual values are calculated as absolute values and therefore taken equally into account. The absolute error is calculated for every observation *i*, after which the sum of the absolute errors is divided by the total number of observations *n*. The MAE is mathematically expressed as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.1}$$

The MAE is scale-dependant, meaning that the calculated forecast errors are on the same scale as the data. For example, an MAE of 5 units indicates that the predictions deviate from the actual values by 5 units on average. Furthermore, the MAE is extremely sensitive to outliers, and consequently, the MAE is typically employed in combination with additional metrics [50].

Mean absolute percentage error: The *mean absolute percentage error* (MAPE) calculates the *mean percentage* deviation between predictions and actual values. The MAPE is very similar to the MAE, with the exception of dividing the error with the actual value at observation *i* and multiplying the sum of the absolute values with 100. The MAPE is mathematically expressed as

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{3.2}$$

The MAPE is *scale-independent*, making it easier to interpret because the error is observed in a percentage format. However, when applying MAPE it should be taken into account that small denominators lead to zero division or a blown-up value. The percentage accuracy of a model is obtained by subtracting 100 from the MAPE value.

Mean square error: The *mean square error* (MSE) is a metric that calculates the average squares of the difference between the predicted and actual values. Since the square of errors is taken, larger errors have more weight on the score, making the model very sensitive to outliers [15]. The MSE is mathematically expressed as

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.3}$$

The MSE is a popular scale-dependent evaluation metric employed in combination with other measures like MAE [53]. It is difficult to comprehend the meaning of an MSE of 55.24, and therefore the root sum of the errors is calculated in order to transform the metric in the same unit as the forecast variable.

Root mean square error: The *root mean square error* (RMSE) metric addresses the scale dependent problem identified in the MSE metric by adding a root to the MSE. Essentially, it reverts the unit of measurement to the original scale. The RMSE is mathematically expressed as

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{3.4}$$

The RMSE is a good measure of how accurately the model predicts the response and it has the same unit as the predictions. In addition, the RMSE is robust to outliers.

The errors discussed above do not provide an absolute evaluation by themselves [39] in terms of FTS forecasting. For example, MSE as a loss function forces the models to learn a 'moving average' of sorts. *i.e.* the best bet is a line through the middle. Thus for this dissertation, accuracy is considered. The accuracy metric is more reliable as it can be used with other optimisation techniques to solve a portfolio problem. The accuracy is simply the ratio between the number of correct guesses about whether the price movement increased or decreased and the total number of guesses. The accuracy formula is expressed as

$$Accuracy = \frac{\text{Number of right guesses}}{\text{Total number of guesses}} \tag{3.5}$$

3.3 Relevant machine learning algorithms

There are many statistical and ML algorithms in literature, each of which is applicable to a variety of tasks. The ability of an algorithm to successfully fit the data and to generate accurate results depends on how the data is structured during data preprocessing. The purely statistical algorithms that are important in the context of this dissertation assume that the TS data is *stationary*. ML models, on the other hand, do not make the assumption of stationarity. Algorithms that are important in the context of this dissertation are described in this section.

3.3.1 Linear regression

Linear regression is a well-known statistical learning method utilised to predict the value of quantitative variables and forms the basis of many other learning methods [55]. The underlying assumption of the linear regression method is that there is an approximate linear relationship between the input variables and the output variable [57]. There are two types of linear regression models within literature, namely *simple linear regression* and *multiple linear regression*.

Simple linear regression is considered as a straightforward approach for predicting a quantitative response y on the basis of a single predictor variable x. The linear relationship is written as

$$y = \beta_0 + \beta_1 x + \epsilon \tag{3.6}$$

where, β_0 and β_1 are two unknown constants namely the *intercept* and *slope*. Collectively, the constants are known as the model *coefficients* or *parameters*. In practise, these coefficients are unknown and must be estimated using the training data provided. The parameter ϵ is a small error term. The data is written as *n* observation pairs $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, each of which consists of a measurement of x and a measurement of y. The objective is to obtain coefficient estimates β_0 and β_1 in order to *fit* the model to the data in such a way that each prediction, $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$, is as close to the *true* value y_i as possible [57]. The most popular method for ensuring *closeness* is *least squares regression*.

However, most problems have more than one predictor variable that dictates the model outcome. To accommodate multiple predictors, the simple linear regression model is extended to form a multiple linear regression model, which takes the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_i x_i + \epsilon,$$

where x_i represents the i^{th} predictor and β_i quantifies the association between the variable and the response. If all other predictors are held constant, each predictor β_i is interpreted as the average effect on y for a unit increase in x_i [57]. The coefficients for multiple linear regression are estimated similarly to the simple linear regression case. Popular measures of accuracy are the MSE and RMSE, which were discussed in Section 3.2.3.

3.3.2 Autoregressive integrated moving average

Many TS behave as though they have no fixed mean. Even so, they exhibit homogeneity because apart from local level and trend, one part of the series behaves much like any other part. Models that exhibit such homogenous *non-stationary* behaviour are obtained by assuming that some *difference* of the process is stationary. These models are referred to as autoregressive integrated moving average (ARIMA) processes [117].

The ARIMA process, proposed by Box and Jenkins, is a prominent statistical model used for analysing and forecasting non-stationary TS [16]. The ARIMA model assumes that past observations can alone be utilised when forecasting by taking the observed trends, cycles and errors of TS into account. It is a simplistic model comprised of three components, namely an *autoregressive model* (AR), a *moving average model* (MA), and an order of difference (I). The ARIMA model has three parameters p, d and q, which are obtained from the components (one parameter from each). The model is therefore presented as ARIMA(p, q, d). Before considering the complete ARIMA model, each component is examined separately.

Auto regression: Regression is a statistical method that attempts to determine the strength of a relationship between one dependant variable and a series of independent variables [16]. An AR model is nothing more than a regression model which assumes that preceding values are partially auto-correlated with the current value, and therefore a prediction for the current value is generated by using the previous values of the data point itself [77]. An AR model is expressed as

$$y_{t(AR)} = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \varepsilon_t. \tag{3.7}$$

where, $y_{t(AR)}$ is the variable predicted by the AR model, μ is the average of the changes between consecutive observations, γ_i are the lagged values (also known as regression coefficients) and y_{t-i} are the lagged values of y_t. Furthermore, p is a parameter referencing the number of AR terms or the lagged values y_{t-i}, and ε_t is the white noise, also called the residual error. An AR model is similar to a multiple linear regression model in the sense that it employs a combination of predictors to forecast the variable of interest, with the exception of employing lagged values of y_t as predictors [92].

Moving average: Rather than using the prior values of the forecast variable in a regression, as was done by the AR model, a MA model employs past forecast errors to predict a future data point [63]. The MA model is expressed as

$$y_{t(MA)} = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}. \tag{3.8}$$

where, $y_{t(MA)}$ is the predicted variable by the MA model and θ_i are the coefficients of the lagged forecast errors. The parameter q denotes the number of lagged forecast errors. Each generated y_t value is a weighted moving average of the past q forecast errors.

Integration: The AR and MA models require stationary TS data with no trend and a constant variance and mean. This requirement is problematic considering that most TS data sets have a trend or associated seasonality, known as *temporal dependence* [16]. The characteristics of a TS naturally make it non-stationary. Hooker and Cave-Brown proposed a technique known as differencing to remove the trend within a TS, thereby transforming a non-stationary TS to a stationary TS [77]. Differencing calculates the derivative of a series by replacing the actual observed values with the difference between consecutive observations. The number of times a TS is differenced to make it stationary is denoted by the parameter d.

In order to determine the number of times a series should be differenced, a statistical test is performed on the given TS to determine whether it is stationary or non-stationary. The most commonly utilised statistical test is known as the *augmented Dickey-Fuller* (ADF) test [23].

Augmented Dickey-Fuller test

Fundamentally, the ADF test is a statistical significance test, meaning that hypothesis testing is involved with a resultant test statistic and p-value. The ADF test belongs to a category of tests called *Unit Root tests*. A unit root is a characteristic of a TS that makes it non-stationary. Therefore, the presence of a unit root, indicated by $\alpha = 1$, is indicative of a non-stationary TS [86].

The Dickey-Fuller test has a null hypothesis requiring $\alpha = 1$. If the null hypothesis is not rejected, the series is non-stationary. As the name suggests, the ADF test is an 'augmented' version of the Dickey-Fuller test. The ADF test expands on the Dickey-Fuller test to include high order regressive processes,

meaning that the number of differencing terms are increased. The ADF test has the same null hypothesis as the Dickey-Fuller test.

The ARIMA model

After applying differencing to the series, the ARIMA model combines Equations 3.7 and 3.8 to form the ARIMA(p, q, d) equation, which is expressed as

$$y_t^* = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i}^* + \varepsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}. \tag{3.9}$$

where y_t^* is the predicted variable by the ARIMA model. Once the TS is stationary the *autocorrelation function* (ACF) and *partial autocorrelation function* (PACF) are applied in order to determine the values of p and q [16, 63]. The ACF and PACF graphs are plotted in Figure 3.6. Both correlation functions are measures of association between current and past TS values. The ACF refers to the correlation between a TS y_t and its lagged values y_{t-i}. Stated simply, the ACF indicates how the present value of a TS is correlated with the past i values. If y_t is correlated with y_{t-1}, then y_{t-1} and y_{t-2} are also correlated according to the ACF. The presence of such correlation is attributed to either the fact that new information contained in y_{t-2} could be used to forecast the value of y_t, or that both time periods are connected to y_{t-1}. The uncertainty is solved with a PACF. A PACF explains the partial correlation present in the series and the lags of itself. Stated simply, a PACF measures the correlation between y_t and y_{t-i} after the removal of the effects of the lags $1, 2, 3, ..., i-1$.

The dashed lines in Figure 3.6 delimit the significant correlations, where the values above the line are significant. The spike in the first lag in both the PACF and ACF plots in Figures 3.6(a) and (b) indicate that y_t is strongly correlated with y_{t-1}. The lag beyond which the PACF cuts off indicates the number of AR terms, and similarly the lag beyond which the ACF cuts off indicates the number of MA terms. From this figure, the number of AR terms are $p = 0$, and the number of MA terms are $q = 1$.

The ARIMA model is especially successful in short term TS forecasting [77]. Furthermore, it is a strictly statistical approach that only requires the prior data of the TS to generalize the forecast. In only requiring the prior data of the TS, the model is able to increase the forecast accuracy while keeping the number of parameters to a minimum [92].

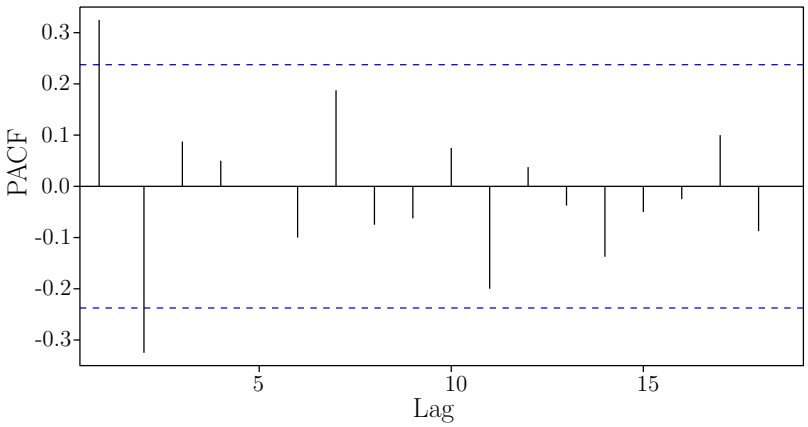
The major drawbacks of a ARIMA model are that it is 'backward-looking' in the sense that long term forecasts eventually go to straight line predictions, and it struggles to predict series with turning points [16]. In the context of predicting financial TS data, ARIMA models have definite limitations in counterfeiting properties, namely asymmetries, sudden outbreaks at irregular time intervals, and periods of high and low volatility.

3.3.3 Support vector machines

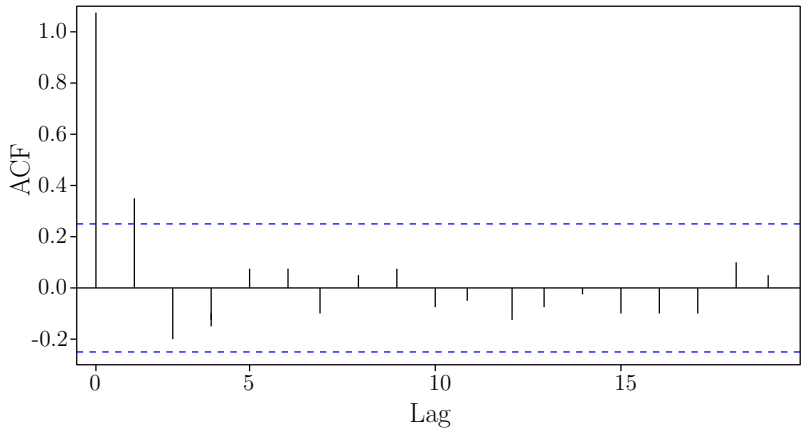
Support vector machines (SVMs) [54] are a family of supervised learning models that have been shown to perform well in a variety of scenarios, including classification and regression problems. SVMs are robust to outliers and they have an excellent generalisation capability [55].

The objective of SVM models is to find a *hyperplane* separating data points of two different classes by as wide a margin as possible [55]. A hyperplane is defined as a flat affine subspace of dimension $j-1$, where *j* is representative of the number of attributes within the training data.

Consider the two-dimensional example in Figure 3.7. The data points belong to one of two classes, represented by pink and blue dots, respectively. The hyperplane in the figure is represented as a solid black line that separates the two classes, while the black dashed lines represent the *margin* by which the classes are separated [57]. The data points which lie on the dashed lines are known as *support vectors*.



(a) *The PACF plot gives the number of MA terms q*



(b) *The ACF plot gives the number of AR terms p*

FIGURE 3.6: *Correlation functions plots.*

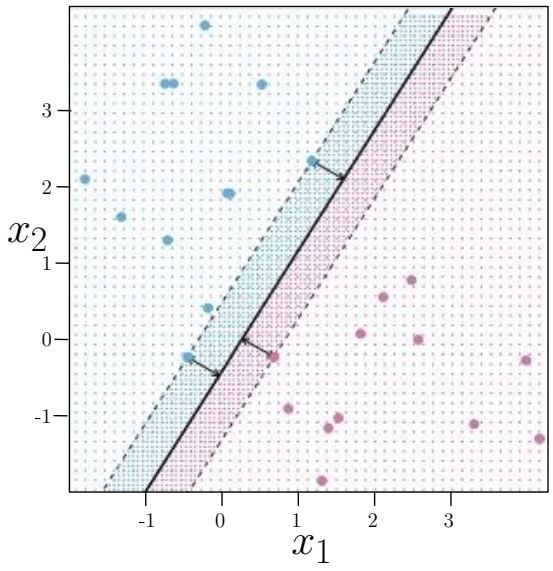
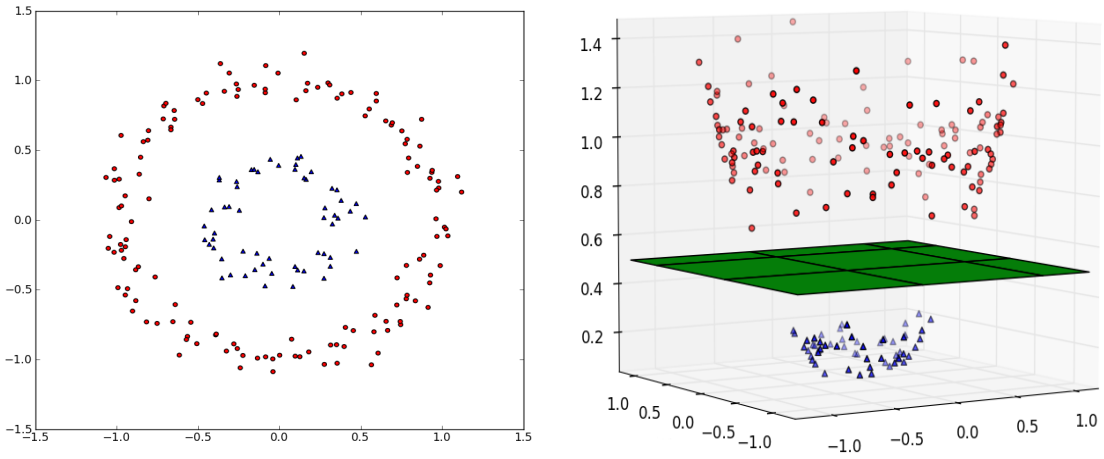


FIGURE 3.7: *Visualisation of the SVM algorithm.*

For the classes to be distinctly separated, any linear line between the dashed margin lines could have been chosen as the separating hyperplane. However, the chosen hyperplane depicted in the figure was chosen such that the margin, or the distance between the points closest to the support vectors in each class, is maximised [57]. By maximising the distance between the hyperplane and the support vectors, the SVM is less sensitive to noise and is, therefore, able to generalise better. On the contrary, if the hyperplane is too close to the support vectors, the SVM will be very sensitive to noise and not have a good generalisation ability.

If, however, it is found that the data is not separable by a linear hyperplane, the so-called *kernel trick* is employed, as is illustrated in Figure 3.8 [55]. The blue and red dots depicted in the two-dimensional plane in Figure 3.8 (a) cannot be separated. However, if the points are plotted to a higher dimensional plane (in this case three-dimensions), then the categories can be separated by a hyperplane as illustrated in Figure 3.8 (b) [55].



(a) *Linearly inseparable set of observations in a two-dimensional plane*

(b) *Transformation of inseparable observations to a three-dimensional plane*

FIGURE 3.8: *A demonstration of the kernel trick [55].*

In general, the separation of two categories is not as clear-cut as illustrated in Figure 3.7. A non-negative *tuning parameter* C is introduced when observations from both categories overlap. *Slack variables* allow an individual observation to be on the wrong side of the hyperplane or margin by a certain distance, governed by C [55]. The hyperparameter C controls the bias-variance trade-off in the SVM model — a small value of C results in a high bias, but a lower variance.

The intended use of SVMs is binary classification. However, the algorithm has been successfully extended to accommodate several classes by employing the *one-versus-one* approach [54]. A further extension of SVMs is to predict a quantitative instead of a qualitative response [13]. The extension is referred to as *support vector regression* (SVR). Both SVR and linear regression models employ a loss function. However, instead of seeking coefficients to minimise the sum of squared residuals (done by linear regression), SVR considers the *epsilon-insensitive loss function* [107]. The goal of SVR is to find a function $f(\boldsymbol{x})$ that deviates from y_x (the actual output) by a value no greater than ϵ (a parameter value) for each $\boldsymbol{x}$ in the training data set while keeping the curve as flat as possible [55, 106]. Therefore, only error terms larger than ϵ are considered in the calculated loss function which is to be minimised. The function $f(\boldsymbol{x})$ in SVR is similar to the hyperplane separating classes in support vector classification.

3.4 Neural networks

This section provides an introduction to neural networks and deep learning. Subsequently, optimisation algorithms used during the training of a neural network are investigated and documented. Furthermore, metrics to evaluate the learning process during the training of a neural network and methods that improve the learning process are discussed. The section ends with important considerations regarding the design choices of neural networks.

3.4.1 Introduction to neural networks and deep learning

An artificial neural network (ANN) [4] is a model that is fitted over data to appropriate the functional mapping, $f : \mathbb{R}^I \to \mathbb{R}^K$, where I is the number of inputs and K is the number of outputs. The fitting is accomplished through the application of an optimisation algorithm to minimise a given loss function.

The neurons in an ANN are loosely modelled on those within the human brain, which process signals from surrounding neurons and pass on signals if certain thresholds are exceeded (non-linear activation function) [37, 57]. A neuron in an ANN is a nesting of functions of the form $g(f(\boldsymbol{x}))$, where $f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + b$ is a linear *score function* and $g(s)$ is an activation function. The $\boldsymbol{W}$ and b terms in the scoring function refer to the *weight matrix* and *bias*, respectively [42]. The weights and biases are parameters which are summarised in a parameter matrix $\boldsymbol{\theta}$. Typically, neural networks are represented in the form of a structured computational graph, with *compute nodes* organised in *layers* and connected by edges in a directional manner, as illustrated in Figure 3.9 [57, 71].

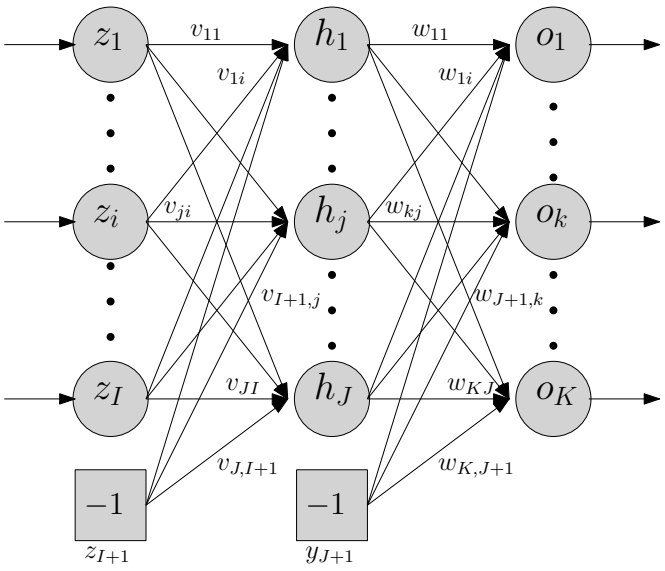


FIGURE 3.9: *Architecture of a complete feed forward neural network.*

The ANN illustrated in Figure 3.9 is referred to as a *feedforward* neural network, on account of the edges of the graph flowing in a single direction, from the input layer to the output layer. The first vertically organised column is referred to as the *input layer*, the last column is referred to as the *output layer*, and the remaining layers in the middle are known as the *hidden layers*. A feedforward neural network can have more than one hidden layer with hidden units h_j. However, Figure 3.9 has only one hidden layer for illustration purposes. Furthermore, the input layer has one neuron for each of the input features z_i, while the number of neurons in the output layer corresponds to the number of target variables o_K. Subscripts

I, J and K refer to the total number of input, hidden and output units, respectively. The subscripts i, j and k refer to the i^{th}, j^{th} and k^{th} input, hidden and output units, respectively. The notation v_{ji} refers to the input weight and hidden unit weight, while w_{kj} refers to the weight of the hidden unit and the output unit.

Deep learning refers to an ANN where the network is comprised of several hidden layers [18]. Through the nesting of non-linear functions, deep learning can model arbitrarily complex relationships to solve a variety of problems. In addition, given enough hidden layers, neural networks are able to model any sufficiently smooth function to any desired level of accuracy. Neural networks are therefore considered as *universal approximators* [42]. Deep learning shifts the responsibility of *feature extraction* from the modeller to the model. Figure 3.10 visualises the weight vectors learnt by a neural network for a facial recognition task. In the first layers, the network extracts low-level concepts, like colour contrasts and corners, which are then iteratively combined to create meaningful higher-level concepts for solving the facial recognition problem.

(a) *Low-level features* (b) *Mid-level features* (c) *High-level features*

FIGURE 3.10: *Features extracted by a neural network for a facial recognition task [89].*

3.4.2 Training neural networks

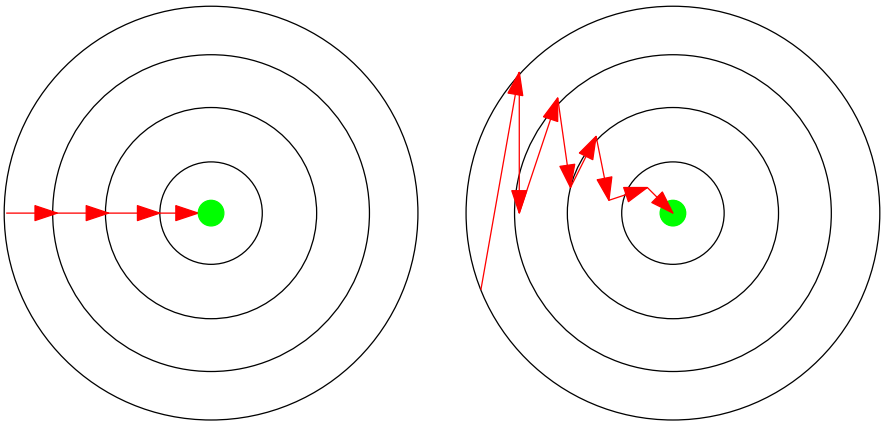
As mentioned previously, each layer of an ANN has a parameter matrix $\boldsymbol{\theta}$ comprised of the weights and biases. The parameter values are determined by minimising the cost function, $L = \sum_{e=1}^{n} L_e\left(\boldsymbol{\theta}, \boldsymbol{x}_e, \boldsymbol{y}_e\right)$. The error for a particular observation is denoted by the loss function L_e, and n refers to the number of training observations.

Approximate methods are utilised during the optimisation of the cost function, because the function of the network is typically non-convex [42]. *Gradient-based optimisers* are the most common approaches employed to minimise the cost function. The remainder of this section describes the gradient-based optimisation algorithms typically employed during the training of neural networks.

Gradient descent

Gradient descent is a first-order iterative algorithm also known as *the method of steepest descent* [42]. An arbitrary point $\boldsymbol{x}_0$, chosen as the starting point of the gradient descent algorithm, moves a certain distance $\boldsymbol{a}$, known as the *step size*, in the direction defined by the negative gradient vector at $\boldsymbol{x}_0$ with the objective of reaching the minimum point of the function. The step size is scaled by the *learning rate* η. Therefore, the learning rate is a multiplicative factor applied to the gradient.

The process of moving towards the minimum point of the function is repeated until a stopping criterion is met. For example, if the gradient norm is below a given tolerance value, the process is terminated. Gradient descent easily finds a global optimum for convex problems. However, gradient descent struggles for non-convex problems to find a global optimum [42]. The gradient descent algorithm is illustrated in Figure 3.11 (a). The red path denotes the gradient update steps taken by the gradient descent algorithm in search of the global minimum (green dot).



(a) *Gradient descent* (b) *Stochastic gradient descent with momentum*

FIGURE 3.11: *Comparison between gradient descent algorithms.*

Stochastic gradient descent

The *stochastic gradient descent* (SGD) [4] algorithm is an extension of the gradient descent algorithm. With the development of deep learning, the insight of SGD is that the gradient is an expectation, which is approximately estimated using a small set of samples [42]. In other words, the gradient of the loss function is estimated by utilising *mini-batches* each comprised of m observations from the training data.

The term stochastic refers to the fact that different pieces of the data (subsets) are employed during different iterations. The algorithmic procedure is the same as for gradient descent, with the exception of averaging the gradient over m batched samples [37].

Both gradient descent and SGD converges very slowly if the learning rate is not tuned accurately, and therefore alternative methods known as *adaptive learning algorithms* are employed to accelerate the convergence of learning by calculating gradients more efficiently [18].

Stochastic gradient descent with momentum

Momentum is designed to accelerate learning, especially in the case of small but consistent gradients, high curvature, or noisy gradients [42]. The momentum algorithm accumulates an exponentially decaying moving average of past gradients and continues to move in their direction as illustrated in Figure 3.11(c). Essentially, momentum serves to smooth the search trajectory.

The momentum algorithm introduces a variable ν, known as *velocity*, which adds the previous search directions (*i.e.* previous gradients) together in order to smooth the search trajectory. Formally, ν refers to the direction and speed at which the parameters move through parameter space. The velocity along a certain direction increases if the gradient vector repeatedly points in the same direction. However, if the gradient vector points in opposite directions the velocity is cancelled out.

Root-Mean-Squared Propagation

The *root-mean-squared propagation* (RMS Prop) [4] algorithm changes the gradient accumulation into an exponentially weighted moving average in order to perform better on non-convex problems. Instead of increasing velocity in promising directions, the RMS Prop algorithm scales down directions of uncertainty. Empirically, RMS Prop has been shown to be an effective optimisation algorithm for neural networks, and consequently it is currently one of the go-to optimisation methods routinely employed [42].

Adaptive movement estimation

The *adaptive movement estimation* (ADAM) algorithm is a combination of momentum and RMS Prop with a few important distinctions [62]. First, the momentum in ADAM is directly incorporated as an estimate of the first-order moment (with exponential weighting) of the gradient. Secondly, ADAM includes bias corrections to the estimates of both the first-order and second-order moments to account for their initialisation at the origin. ADAM is generally regarded as the most effective optimisation scheme for neural networks [42].

3.4.3 Monitoring the learning process

To evaluate whether the learning process is effective several metrics can be tracked during model training. Metrics include, the value of the cost function (average loss) and the training and validation accuracy of the network during training as a function of the number of epochs (training iterations) [42, 57].

The loss function profile is indicative of whether the parameter optimisation process is working (loss should be decreasing) and whether a suitable learning rate is applied. If the chosen learning rate is too low, the learning process is time consuming since only linear improvements are made using a small step size and the process can become stuck in a local minimum. However, if the learning rate is too high, the loss value will decay quickly in the beginning, but converge to a less desirable value. The parameters are unable to settle on good values in the optimisation landscape and instead "bounce around chaotically" when the adjustments result in large changes [57]. If the learning rate is extremely high, the loss begins to diverge. Overall, an increase in the learning rate results in faster training, but has the disadvantage of converging to a less desirable value if the rate chosen is too large. The effect of different learning rates on the optimisation process is illustrated in Figure 3.12.

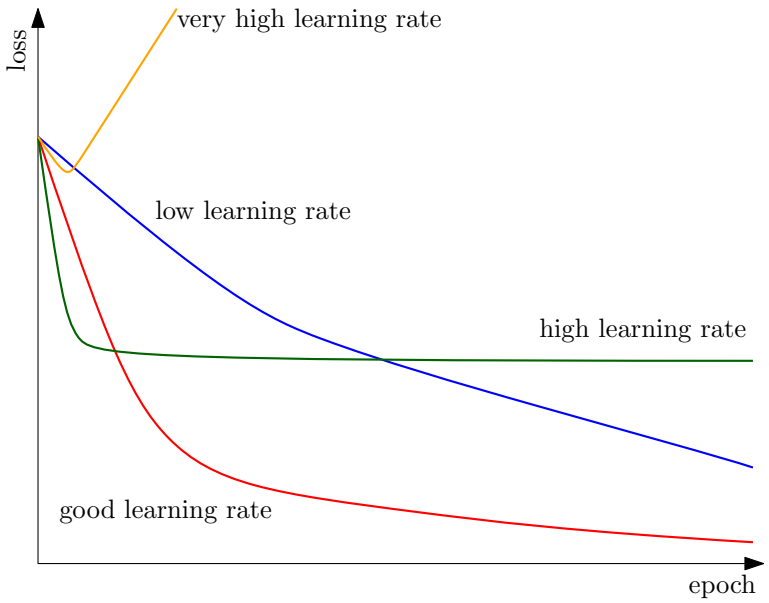


FIGURE 3.12: *Influence of the value of the learning rate adopted in gradient based algorithms [57].*

By plotting the training and validation errors insight is gained into whether a model is underfitting or overfitting the data, as illustrated in Figure 3.5. In this figure, the training and validation errors decrease at the start of the period. As subsequent training iterations are executed, the training error continues to decrease (*i.e.* accuracy continues to increase), while the validation error begins to increase again. As mentioned previously in Section 3.2.2, an increase in validation error indicates that the model is overfitting the data. The difference between the training and validation error is referred to as the *generalisation gap* [102].

Regularisation

There are multiple *regularisation* techniques that are employed to minimise the generalisation gap and to curb overfitting. However, only a few of the regularisation techniques are of interest in this dissertation, namely regularisation with *parameter norm penalties*, *data augmentation*, *dropout* regularisation and *early stopping* regularisation.

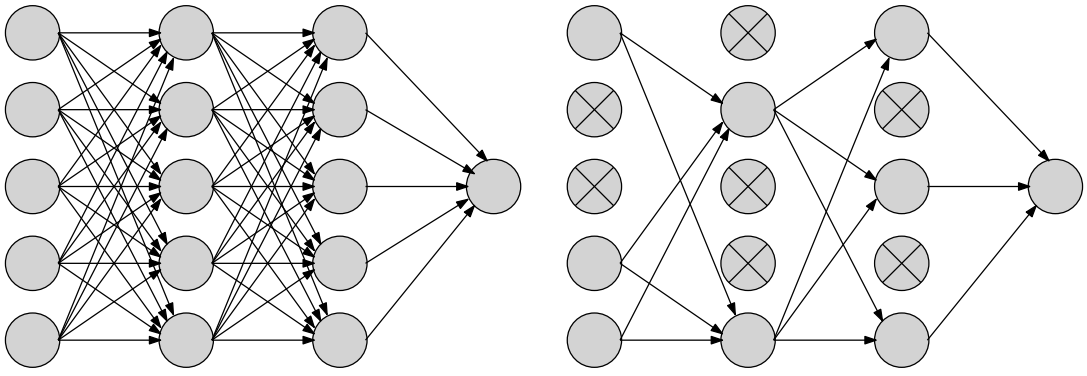
In the domain of ML, regularisation penalises complex models that are prone to overfit the data by shrinking the weights towards zero. Perhaps the simplest and most widely utilised form of regularisation is l_2 and l_1 parameter norm penalties, more commonly known as *weight decay* [118].

The l_2 regularisation strategy drives the weights of a neural network closer to the origin by adding a regularisation term $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$ to the objective function. Furthermore, l_2 regularisation deals with multicollinearity (highly correlated independent variables) problems by constricting the coefficient and keeping all the variables. This regularisation technique penalises insignificant predictors.

While l_2 weight decay is the most common form of weight decay, another option is to utilise l_1 regularisation by adding the term $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{w}\|_1$ to the objective function. This regularisation technique is the preferred choice when a large number of features are utilised, because it provides *sparse* solutions. Sparsity in this context refers to the fact that some parameters have an optimal value of zero.

Another approach is to *augment* [42] the training data by creating fake data in an attempt to simulate the wide variety of transformations that is found in the validation and test sets. In the context of image classification, for example, training images can be altered, cropped or flipped in terms of contrast and brightness to render the classifier invariant to different poses and changes in illumination.

The *dropout* regularisation technique, is a computationally inexpensive yet powerful approach that employs the concept of an ensemble model in a single neural network [42, 118]. A random subset of neurons in the network are disabled during each training iteration, as illustrated in Figure 3.13. The disabled neurons are selected by means of a *discard probability*. The probability could be the same for the entire network or different for each layer. Figure 3.13(a) illustrates a normal neural network architecture, while Figure 3.13(b) illustrates the effect of dropout. The neurons that are crossed out are deactivated with all of their incoming and outgoing connections for the specific epoch of the model. Consequently, the set of active neurons form a different model with a reduced capacity. The collection of models, each emerging during a different iteration is seen as an ensemble of networks with shared parameters.



(a) *Neural network without dropout* (b) *Neural network with dropout*

FIGURE 3.13: *Application of dropout.*

The objective of dropout is to reduce *co-adaption* between neurons. Neurons co-adapt by observing that a neighbouring neuron is extracting a certain feature, and consequently *'choosing'* to focus on a different feature [57]. If, however, a neuron is exposed to a situation in which dropout is implemented, and a

surrounding neuron is deactivated, the neuron starts to *'learn'* to identify the same features that the deactivated neuron had identified previously. Dropout therefore maintains the generalisation performance of a model and as a result the model is more robust to changes in the input data [42].

Finally, the *early stopping* regularisation technique terminates the execution of the model when overfitting is detected, or a stopping criterion is met and returns the weights that resulted in the best performance. More specifically, every time the error on the validation set improves, a copy of the weights that result in the best performance are stored. When overfitting is detected, training stops, and the neural network reverts back to the stored weights that resulted in the best performance. Training is also stopped when no parameters have improved over the best recorded validation error for a pre-specified number of iterations.

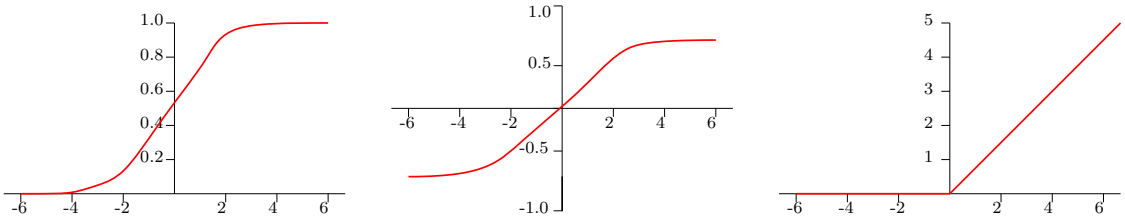
3.4.4 Considerations for network design

Increasingly long sequences of multiplications are generated when the chain rule is applied to deep neural networks. The problem of *vanishing gradients* occur when the individual gradients are significantly small, and as a result the product diminishes to a point where the gradient descent algorithm is no longer effective. Consequently, the design choices with regard to a neural network have a significant impact. More specifically, the activation functions that are employed, the processing of input data, and the initialisation of network weights have a considerable influence [70, 118].

The remainder of this section documents the design aspects that address the diminishing gradient problem. The design aspects investigated are activation functions, data pre-processing and weight initialisation.

Activation functions

Activation functions define how the weighted sum of the input is transformed into an output from a neuron. There are multiple activation functions, namely the *linear*, *step*, *ramp*, *sigmoid*, *hyperbolic tangent* ($tanh$), *Gaussian* and *rectified linear unit* (ReLU) [37]. The most popular choices for activation functions are sigmoid, hyperbolic tanh and ReLU. The shapes of these functions are illustrated in Figure 3.14.



(a) *The sigmoid function.* (b) *The hyperbolic tanh function.* (c) *The rectified linear unit function.*

FIGURE 3.14: *Plots of activation functions employed in neural networks.*

The sigmoid function in Figure 3.14(a) gives a value between zero and one, which is interpreted as a probability. For very large values the derivative of the sigmoid function becomes close to zero. The function is therefore not utilised in deep networks, because it suffers from the vanishing gradient problem [37].

The hyperbolic tanh function in Figure 3.14(b) is a mathematically stretched version of the sigmoid function. Instead of giving values between zero and one, the tanh function gives values between -1 and 1. In the tanh function, the derivative is steeper meaning that a higher value is obtained. In turn, the function is more efficient because it has a wider range for faster learning and grading compared to the sigmoid function. However, the hyperbolic tangent function still suffers from the vanishing gradient problem [42].

The ReLU function has become the default activation function for multiple neural networks because a model that employs this function often obtains a better performance and is easier to train [42]. The ReLU function is easy to implement and does not suffer from the vanishing gradient problem. However, upon examining Figure 3.14(c), it is observed that the gradient saturates and dies out completely when the input is negative, leading to a phenomenon known as "*dead ReLU*" [4, 57]. The neuron will always output zero during future iterations, regardless of the input it receives. This problem is addressed by adopting a variant of the ReLU function, namely the "*leaky*" *ReLU* activation function. The leaky ReLU function is illustrated in Figure 3.15.

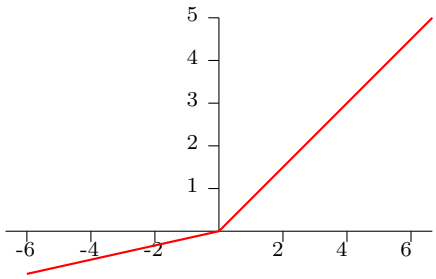


FIGURE 3.15: *The leaky ReLU activation function.*

Instead of defining the ReLU activation function as zero for negative input values, the leaky ReLU function defines the negative input values as an extremely small linear component of the input value.

Data pre-processing and weight initialisation

Among the best practises for training a neural network is to normalise the input data in order to obtain a mean close to zero [37]. By normalising the input data, the learning process is quicker and leads to a faster convergence. Furthermore, all inputs are then in the same scale range, so the same influence is weight adjustments occurs.

Normalisation, also known as *min-max normalisation*, rescales the original data range so that all values are within a specified range. By normalising the data all the distance ratios of the original data set are maintained. Min-max normalisation is expressed as

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}(u - l) + l \tag{3.10}$$

where x' is the normalised data point, x is the original data point, x_{min} and x_{max} are the minimum and maximum values of the original data set, and u and l are the upper and lower values of the new normalised data range, $x' \in [l, u]$.

The initialisation of the weight parameters and biases of a network is an important consideration, because an optimisation algorithm requires a starting point in the space of possible weight and bias values from which to begin the optmisation process to find the set of weights that minimises the loss function.

There are different methods to initialise weights. One method is to initialise all weights to zero before training. However, the initialisation of weights to zero causes the neurons to learn the same features during training, meaning the same function values are computed and the gradients are equal throughout the network. By implementing such a network, most neurons become redundant and learn nothing. As a result *symmetry breaking* is never achieved. The term symmetry breaking refers to the phenomenon when different values are assigned to different weights, which in turn enables the weights to learn independently of each other [6].

To achieve the concept of symmetry breaking, random numbers are assigned to the initial weights. However, if the random numbers are too small, the activations "die" out along the network, leading to the

vanishing gradient problem where weights vanish to zero. On the contrary, extremely high weight updates overshooting the minimal value occurs if the random numbers are too large. This phenomenon is referred to as the *exploding gradient* problem, where weights explode to infinity [42].

Popular initialisation methods for weights are *Xavier* and *He* initialisation. Xavier initialisation sets the weights of a layer to values that are chosen from a distribution with a zero mean and specific variance $Var = 1/$ *fan in* [40]. Where *fan in* refers to the incoming neurons.He initialisation, sets the weights of a layer to a random value with a Gaussian probability distribution with a mean of zero and a standard deviation of $\sqrt{\frac{2}{g}}$, where g is the number of inputs to the node [47]. Xavier initialisation is used when the activation function is a hyperbolic tanh, while He initialisation is used when the activation function is a ReLU [41].

3.5 Neural network architectures

The *architecture* of a neural network includes design choices with regard to the overall structure, given by the number of layers, the number of neurons in each layer and the manner in which the neurons are connected [42]. The architecture of a neural network is typically determined during a hyperparameter tuning process. There are various *types* of architectures that can share certain structural elements or properties.

A *multilayer perceptron* (MLP) is a class of feedforward neural network, comprised of at least three layers: an input layer, a hidden layer, and an output layer [87]. Consequently, the figure shown in Figure 3.9 is also known as a MLP. In terms of the design choice regarding the manner in which the neurons are connected, the MLP in the figure is comprised of only *fully connected* layers. Fully connected means that each neuron of a given layer is connected to every neuron in the preceding layer, as well as to every neuron in the subsequent layer. With respect to deep learning, a MLP with multiple fully connected layers has a significantly large number of weights, necessitating the acquisition of large amounts of data which renders the optimisation process difficult.

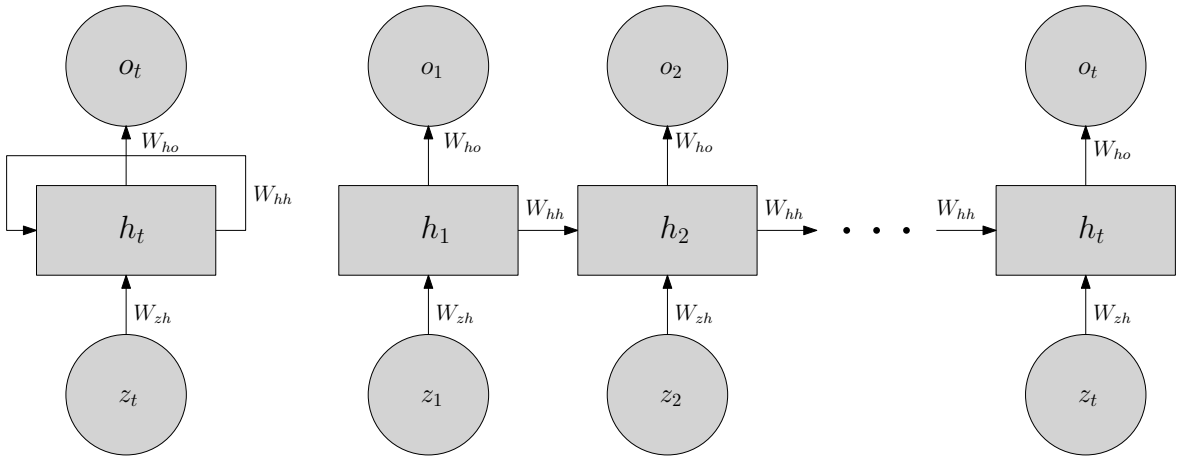
Other neural network architectures are described in the remaining section. More specifically, architectures that are typically employed in the context of TS data are investigated.

3.5.1 Recurrent neural network

A *recurrent neural network* (RNN), first proposed by Rumelhart *et al.* [100], is part of the neural network family and has the ability to process sequential data. Sequential data is defined as data points dependent on other data points within a data set and have a temporal order. There is a one-to-one correspondence between the layers in a network and the specific positions in a sequence. The position of a value in the sequence is also known as a *time-stamp*. Therefore, the network comprises a variable number of layers, each with a single input corresponding to the time-stamp, instead of a variable number of inputs in a single input layer. The inputs can interact directly with downstream hidden layers depending on their positions in the sequence. Each layer within the neural network utilises the same set of parameters to ensure that each timestep is modelled similarly. As a result, the same layer-wise architecture is repeated in time and is therefore referred to as *recurrent* [2, 4].

RNNs are a type of feedforward neural network with a structure based on *time layering*. Time layering means that a RNN can take a sequence of inputs and produce a sequence of outputs. In addition, the output from the previous step is fed as input in the current step [4]. RNNs are typically applied to TS data and text domain processing tasks like machine translation and prediction of the next word in a sequence. In order to enable intuitively simple explanations of various concepts, RNNs are explained in the context of the text domain for the remainder of this section.

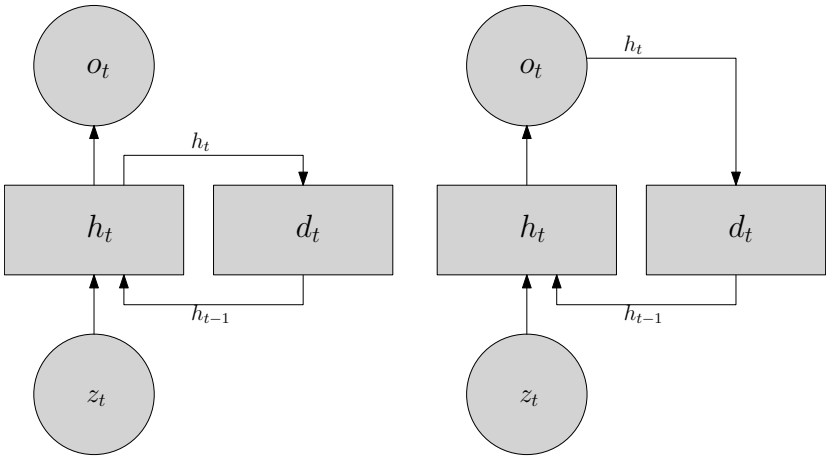
A simple RNN is illustrated in Figure 3.16(a). In this figure z_t, h_t and o_t refer to the inputs, hidden layers and outputs, respectively. Inputs z_t are passed through a neural network, where at each time step an output o_t is produced. The produced output is then passed back to the network as a secondary input during the next time step. The self loop at the hidden layer h_t will cause the hidden state of the neural network to change after the input of each word in the sequence. In Figure 3.16(b), the RNN is unfolded into a "time-layered" network that looks like a feedforward network. This figure shows a different node for the hidden layer at each into a feedforward network. Figure 3.16(b) is mathematically equivalent to Figure 3.16(a). The weight matrices W_{zh}, W_{hh} and W_{ho} in all the hidden layers $h_1, h_2..., h_t$ are shared to ensure that the same function is utilised in each timestep.



(a) *RNN* (b) *Unfolded RNN representation of (a)*

FIGURE 3.16: *A recurrent neural network architecture.*

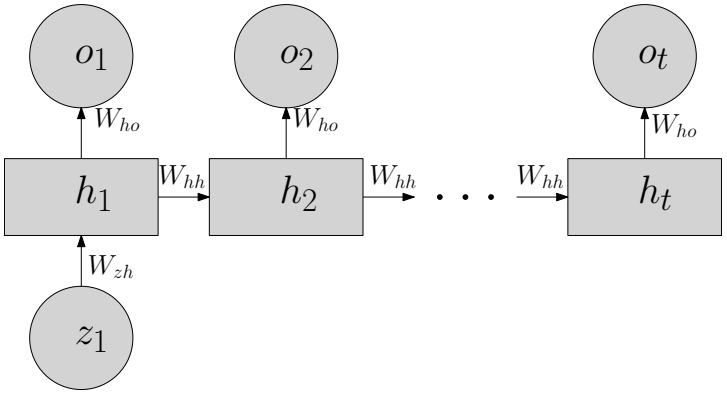
Two popular simple RNN structures are the Elman and Jordan networks [125]. The structure of the Elman and Jordan neural networks is illustrated in Figure 3.17. Both neural networks have an input layer, a hidden layer, a delay layer d_t and an output layer. The delay layer stores information from the previous layer. In the case of the Elman network in Figure 3.17(a), information from the delay layer is fed from the hidden layer. On the contrary, the Jordan network in Figure 3.17(b) shows that information from the delay layer is fed from the output layer [71].



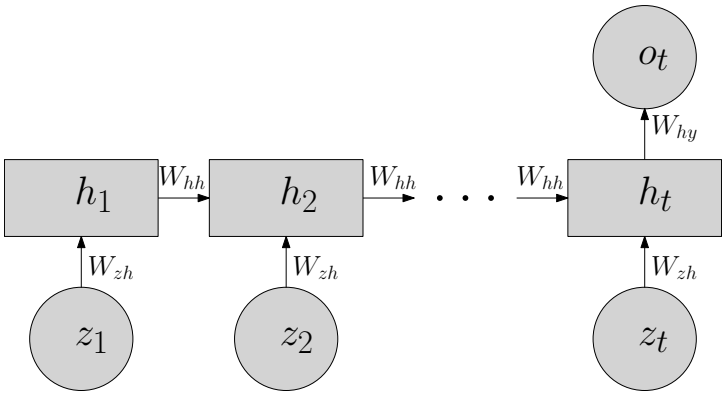
(a) *RNN* (b) *Unfolded RNN representation of (a)*

FIGURE 3.17: *Elman and Jordan RNN structures.*

RNNs are used to model various types of relationships, while feedforward neural networks are typically only employed to model *one-to-one* relationships. An example of an one-to-one relationship is the relationship between an image and the associated class label. RNNs can model *many-to-many* relationships, as illustrated in Figure 3.16, *one-to many* relationships and *many-to-one relationships* [37]. The different structures are modelled by means of an RNN, as illustrated in Figure 3.18.



(a) *One-to-many*



(b) *Many-to-one*

FIGURE 3.18: *Different types of relationships that are modelled by a RNN.*

A RNN model has the disadvantage of being unable to parallelise training, because the layers of an RNN are connected. In other words, the connection refers to the RNNs dependence on the output of the previous node to compute the output for the current timestep. RNNs are therefore not suitable for parallelising with other models.

The central problem associated with the optimisation of RNNs is the modelling of *long-term dependencies*, as discussed by Bengio *et al.* [10]. Consider a text document, which contains the phrase "I grew up in Spain" at the beginning of the document, and the phrase "I speak fluent Spanish" towards the end of the document. If a network predicts the last word of the second phrase, the information at the beginning of the document should be considered. However, the gap between the phrases is significantly large. Consequently, the long-term dependency between the words is modelled by computing the quantity of $W_{hh}h_t$.

However, the problem of vanishing and exploding gradients is still particularly prevalent in this context. As a result, a number of RNN variants have been proposed [4]. Variants of particular interest are LSTMs and GRUs.

3.5.2 Long Short Term Memory

A LSTM [4] network is an extension of RNNs, capable of learning short- and long-term dependencies. The architecture of a LSTM improves on RNNs, which fail to store information for long periods of time, struggle to control which part of the context is to be carried forward, and how much of the past should be forgotten [42].

Consider the following example to explain the contrast between RNNs and LSTMs. If a RNN is applied to the problem '*The colour of the grass is*', it could easily predict the missing word '*green*'. The RNN is not required to remember what was said before this sentence or what the context of the statement is. However, a RNN is unable to solve the problem '*I spent 20 long years working for under-privileged children in Spain. I then moved to Africa . . . I can speak fluent*'. The RNN fails to understand the context behind the input, is unable to recall what was said long ago and is incapable of distinguishing between relevant and irrelevant information. As a result, the RNN is unable to predict the word *Spanish*. On the contrary, a LSTM can predict the missing word correctly by employing a *forgetting* and *saving mechanism* [49]. The forgetting mechanism enables the LSTM to forget information that is not worth remembering, while the saving mechanism allows the LSTM to store important information.

The architecture of a LSTM network is illustrated in Figure 3.19. The key idea behind a LSTM network is the horizontal line flowing along the top of the network, which is subject to only a few linear transformations. The horizontal line is known as the *cell state* C. It is possible for information to flow through the unit along the horizontal path unchanged [4].

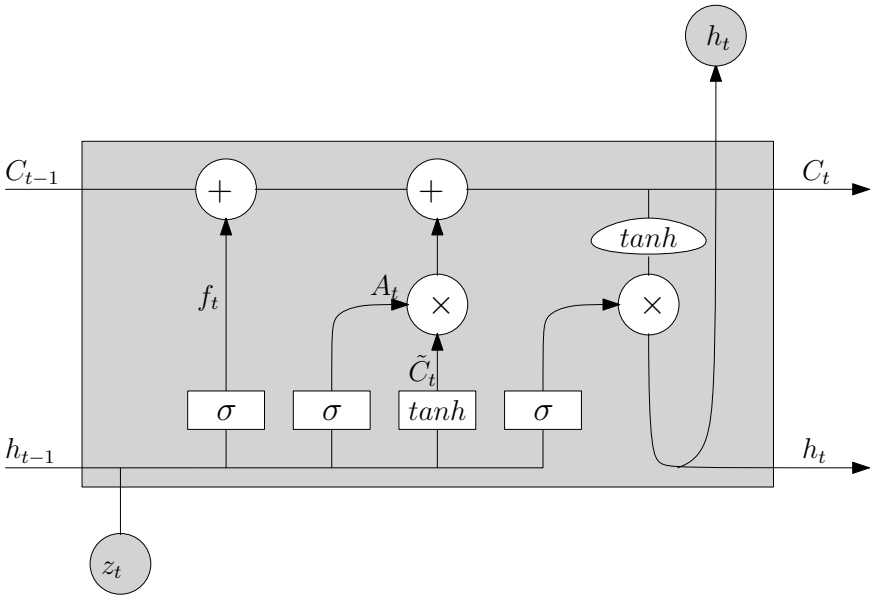


FIGURE 3.19: *A long short term memory network architecture.*

Structures known as *gates* are employed to alter the cell state of the LSTM network. There are three gates within the network, namely a *forget gate*, *input gate* and *output gate*. Each gate is composed of a sigmoid neural net layer and a point wise multiplication operation. The sigmoid layer generates a value where each element is in the interval $[0, 1]$. Each generated value is close to either zero or one. A value of zero indicates that the previous cell state should be forgotten, while a value of one suggests that the prior information is important and should be remembered.

The forget gate is responsible for removing information that is no longer deemed relevant or important [49]. For example, a language model trying to predict the next word based on the previous words. The cell state might include the gender of the present subject so that the correct pronoun is used. How-

ever, if a new subject is observed, the gender of the old subject is forgotten. In the forget gate, as illustrated in the figure, the old cell state C_{t-1} is multiplied by the output of a sigmoid layer. The output of the sigmoid layer is a function of the output of the hidden state from the previous time step h_{t-1} and the input at the current time step z_t.

The second gate in the network, the input gate, governs the addition of new information to the cell state. The gate is comprised of two parts. First, the values of h_{t-1} and z_t are passed through a $tanh$ function, as in a standard RNN. In a LSTM network, however, the $tanh$ function creates a vector of new candidate values $\tilde{C}_t$, which are then multiplied by the input gate which is given by another sigmoid layer. In the input gate, the sigmoid layer chooses how much (as a value between zero and one) of the output should be added to the cell state [66]. In the example of the language model mentioned previously, the gender of the new subject is added to the cell state to replace the one that is forgotten.

The information obtained by the forget and input gates is now used to update the old cell state C_{t-1} into the new cell state C_t. The old state is multiplied by f_t, forgetting the information that was decided upon. The product of the sigmoid and $tanh$ function $A_t * \tilde{C}_t$, is then added to the cell state. In the case of the language model example, the information about the gender of the old subject is dropped and the new information is added. The output gate decides which values of the cell state should be returned. It functions similarly to the input gate, except that the new cell state C_t is passed through the $tanh$ and sigmoid functions instead of h_{t-1} and z_t.

LSTMs address the limitations experienced by RNNs, namely the vanishing and exploding gradient, and the short term memory problems [43]. However, LSTMs have drawbacks, namely being computationally inefficient during training in terms of time and memory, having a tendency to overfit, dropout regularisation is harder to implement, and being sensitive to different random weight initialisations [42].

3.5.3 Gated recurrent unit

A variation of the LSTM that mitigates the mentioned drawbacks is the GRU network, introduced by Cho *et al.* [24]. Like the LSTM network, a GRU network employs gating units to modulate the flow of information and solves the vanishing gradient problem. A GRU differs from a LSTM in its employment of cell states and gates [42]. GRUs do not use explicit cell states to store information (as is done by LSTMs) that instead incorporate cell states into the hidden state. In the context of gates, a LSTM employs a separate forget and output gate to control the amount of information altered in the hidden state. In comparison, a GRU employs a single *reset gate* to achieve the same goal. The GRU has fewer parameters by only employing a single gate, making it less complex than a LSTM. The decrease in complexity results in a significant decrease in computational time while still enabling the model to obtain a similar or superior performance in comparison with LSTMs [19]. The architecture of a GRU network is illustrated in Figure 3.20.

The *update gate*, U_t, captures long-term dependencies by determining how much of the previous hidden state h_{t-1} should be retained in the current hidden state h_t, and how much of the new memory content z_t should be added [72]. To calculate the update gate, an input vector and previous hidden state value is required as input. The two inputs are summed and passed through the sigmoid function. The sigmoid function in the GRU has the same purpose and definition as the sigmoid function in the LSTM network.

The *reset gate*, r_t, determines how much of the past information is irrelevant by deciding how to combine the input, z_t, with the previous hidden state, h_{t-1}. The reset gate enables the GRU to "reset" itself by blocking information that is no longer relevant [31]. The reset gate, r_t, follows the same process as the update gate, with the exception of adding the input, z_t, and hidden state, h_{t-1}, with *point-by-point* addition [24]. The reset gate is analogous to the output gate of a LSTM.

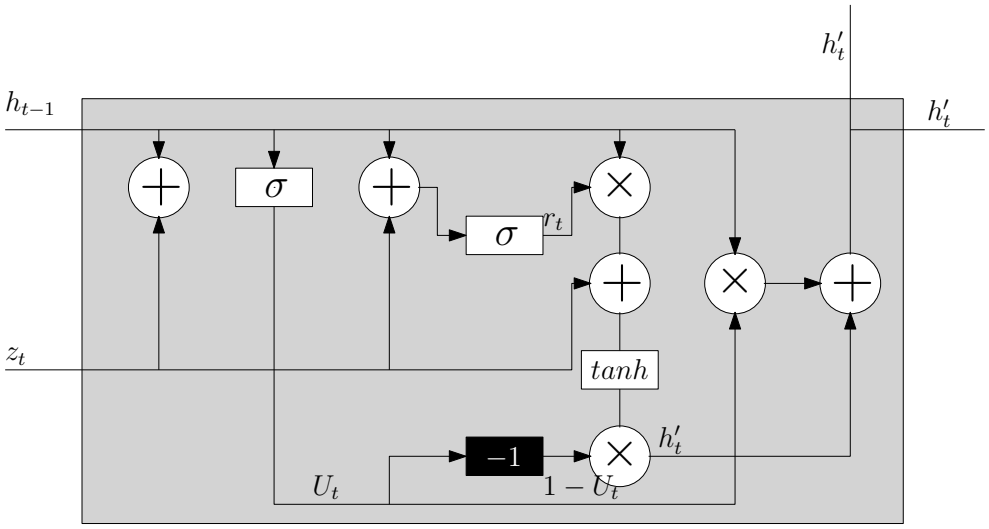


FIGURE 3.20: *A gated recurrent unit network architecture [25].*

The element-wise product (Hadamard) between the reset gate, r_t, and hidden state, h_{t-1}, is then calculated and summed with the input, z_t. The result is applied to a non-linear function like $tanh$ to produce the current memory content, h'_t. Finally, the GRU determines which information in the current unit is passed down to the next cell, h_t. The update gate plays a vital role in this section, since it determines which information to collect from the current memory content, h'_t, and previous time step, h_{t-1}. In order to determine which information is collected, element-wise multiplication is applied to the update gate and h_{t-1}. The computed value is then summed using the Hadamard product operation between $1 - U_t$ and h'_t [27, 73].

The simpler structure of a GRU enables the network to be more compact and efficient than previously discussed models. In addition, a GRU network has an improved memory capacity and does not suffer from the vanishing gradient problem [42].

3.6 Chapter summary

This chapter introduced the notion of machine learning and the various associated paradigms in this field, namely supervised, unsupervised, semi-supervised, reinforcement, and evolutionary learning. Subsequently, a description of the general procedure for training and evaluating machine learning models is followed. More specifically, the procedure included a detailed description of the hyperparameter tuning process, machine learning model generalisability and the bias-variance trade-off and performance evaluation metrics. The hyperparameter tuning process outlined the most common optimisation approaches, namely manual search, grid search, random search, Bayesian optimisation, and the F-race algorithm. The performance evaluation metrics investigated include the *mean absolute error* (MAE), *mean absolute percentage error* (MAPE), *mean square error* (MSE), *root mean square error* (RMSE) and accuracy.

Several machine learning algorithms were described next, namely, linear regression, *autoregressive integrated moving average* (ARIMA) and *support vector machines* (SVMs). The model description was followed by a review of neural networks, which include an overview of the training, design and monitoring processes and techniques associated with this type of machine learning algorithm. Finally, an account of several neural network architectures was provided. The neural networks discussed are *multilayer perceptrons* (MLPs), *recurrent neural networks* (RNNs), *long short term memory networks* (LSTMs) and *gated recurrent units* (GRUs).

CHAPTER 4

Empirical process

Contents

This chapter presents the empirical process followed in this dissertation to compare the forecasted future price movements of financial assets. The chapter opens with a high-level description of the empirical process followed. The description is followed with a more in depth explanation of the primary components of which the process is made up of using data flow diagrams.

4.1 Financial time series empirical procedure

The empirical process followed to compare and contrast the capabilities of ML models based on the types of FTS and validation technique selected is outlined in the *financial time series empirical procedure* (FTSEP). The FTSEP is concerned with the data preparation, modelling, evaluation and analysis phases that are executed in pursuit of identifying general ML approaches for FTS forecasting. Before describing the FTSEP, it is necessary to state the components of the data set. Given FTS data from financial markets, the FTSEP assumes data inputs stored in a tabular format which describe:

1. the financial assets from different financial markets, and
2. the daily close price of each respective asset for a period of time.

Assets from different financial markets react differently in terms of price movement under different economic conditions. For example, during times of fiscal uncertainty, precious metals in the commodity market experience a positive price movement, while assets in the stock market that produce non-essential goods experience a sell-off resulting in a negative price movement [34]. A selected ML model may successfully identify the underlying patterns within the data over a specified period for an asset in the commodity market and obtain a high evaluation performance. However, the same ML model may obtain a low performance in the stock market, because the model was unable to identify the underlying patterns over the same specified period. Hence, the performance of a selected ML model is dependant on the type of asset data utilised, introducing the problem of *selection bias*. Selection bias refers to a bias in the selection of data for training machine learning models [69]. To prevent selection bias in pursuit of comparing and contrasting the capabilities of different ML models, asset data from different financial markets are utilised.

The implementation of the proposed FTSEP is documented by means of a *data flow diagram* (DFD), introduced by Kendall and Kendall [59]. Kendall and Kendall describe a DFD as a graphical representation to conceptualise how data flows between a variety of *entities*, *processes* and *data stores*. Four basic symbols are utilised to chart data movement on DFDs, namely a double square representing an external entity that receives or sends data, an arrow indicating the flow of data, a rectangle with rounded corners representing a process in which data are transformed, and an open-ended rectangle representing a data store that allows for the examination, addition and retrieval of data. A DFD is represented at various levels of abstraction. In the context of the FTSEP, a top-down approach is followed by first providing a broad overview of the system, known as a *Level 0* diagram. Each of the processes within the Level 0 diagram is decomposed into various sub-processes by means of *child diagrams* known as *Level 1* diagrams.

Figure 4.1 illustrates the proposed Level 0 DFD for preparing, modelling, evaluating and analysing FTS data. The FTSEP does not account for unstructured data and therefore, the user should ensure that the required data sets are in a structured format. From a high-level overview, the Level 0 DFD is comprised of three primary components, namely, *processing*, *modelling*, and *analysis*. The three components are linked to a database, in which relevant data are stored and retrieved. In addition, each component is comprised of multiple processes.

In the processing component, raw data are provided by the user. The *data collection* step of the data science process is assumed to have already been completed by the user. These data are then processed by multiple functionally coherent modules, which perform several tasks, including data formatting, data filtering and data cleaning. Processed data are then employed as input to the modelling component. Modules residing in the modelling component perform tasks related to model building and evaluation. In order to extract valuable information and insight from these results, a subsequent analysis of the results is necessary. The data concerned with the evaluated modules are employed as input for the analysis component, which comprises multiple modules that generate table summaries and visualisations of the results.

The processing and modelling components in Figure 4.1 are executed for each of the chosen assets in the financial market. For each iteration, the performance metrics for the selected asset are visualised and stored. After all iterations are completed, final summaries and visualisations are generated to compare and contrast the capabilities of different ML models.

The code used to implement the FTSEP is available at `https://github.com/Judene/fts_predictors`. All the code is written in the Python programming language. The code used in this dissertation is accessible to anyone who wants to do further research in this area while also encouraging the repeatability of results.

4.2 The processing component

The processing component comprises three modules numbered 1.0 to 3.0 as illustrated in the Level 0 DFD in Figure 4.1. The objective of the respective modules in the processing component is to process the raw data into an acceptable format for the subsequent component, *i.e.* modelling.

The first step in the processing component is the *formatting* of data, which entails sorting the data by date, as well as defining the data type of the features as quantitative. The raw data set is then *filtered* based on the asset type selected. The filtered subset of data is *cleaned* by either inserting known values in the place of missing values or removing observations with missing values. To make an accurate performance comparison between the ML models, the outliers are necessary and not removed. Essentially, the outliers have a significant impact on the error magnitude of each ML model. In turn, the error magnitude (influenced by the outliers) indicates how well the ML model could model the data. As mentioned previously, the processing component contains elements of repetition, and hence Modules 2 and 3 are executed for each asset type. On the contrary, Module 1 is executed only once.

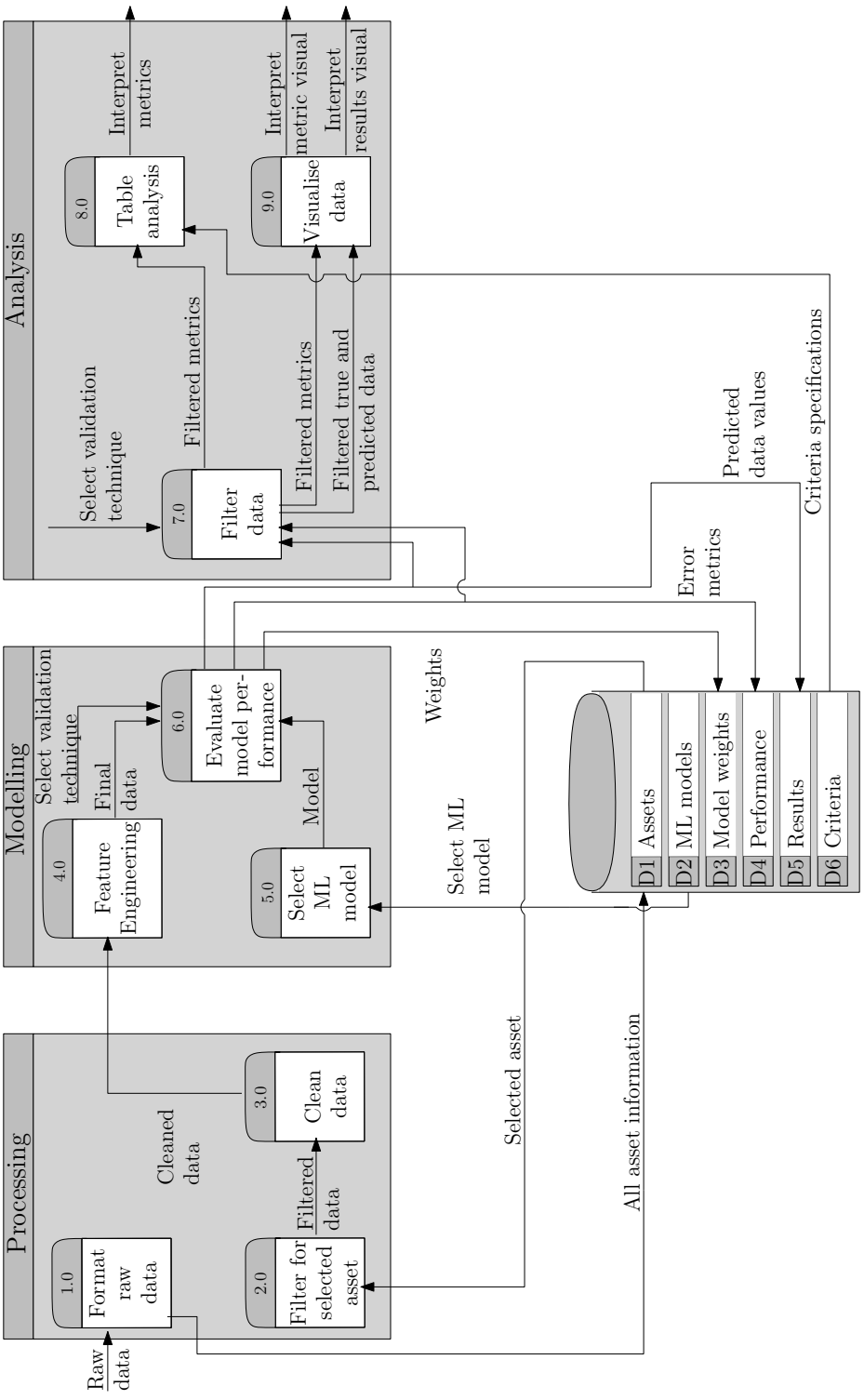


FIGURE 4.1: *Level 0 DFD of the financial TS modelling framework.*

A more detailed representation of the processing component is illustrated in the Level 1 DFD in Figure 4.2. The formatting of data (Module 1.0) is partitioned into two child processes. In the first child process, the user supplies the raw FTS data to the system, which sorts the data in ascending order. The raw data is comprised of the following assets: the S&P 500 index, the bond yield, the USD/ZAR cur-

rency pair, gold and Bitcoin. In Module 1.2, the sorted data is defined as a quantitative data type and stored in data store D1, referred to as Assets. The Assets data store enables the user to iteratively access the formatted data set for each asset without executing Module 1.0 again. Module 2.0 is not expanded into child processes, because it represents a simple process in which the user selects an asset from the Assets data store.

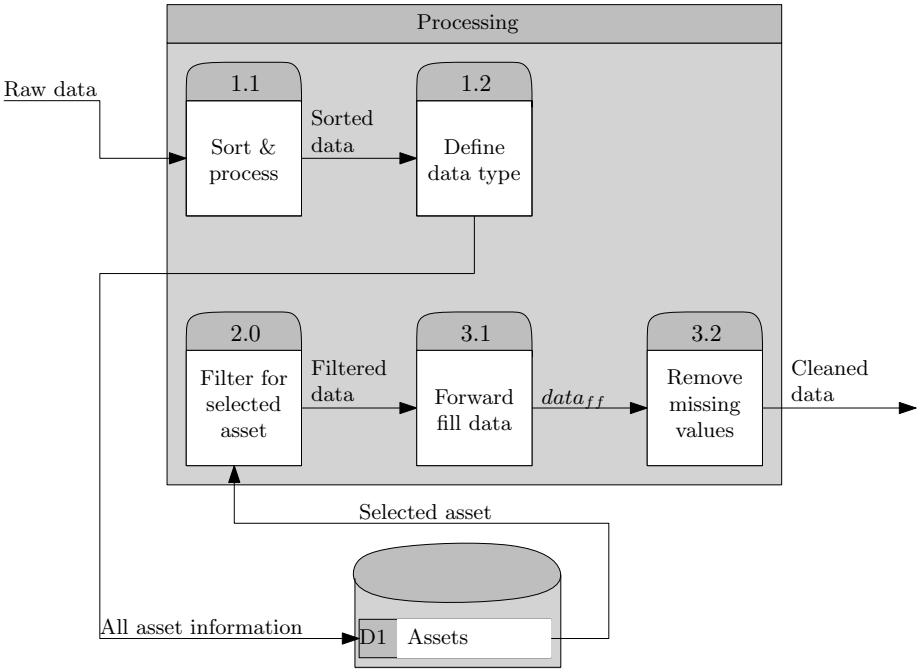


FIGURE 4.2: *Level 1 DFD of the processing component.*

The cleaning of data (Module 3.0) is partitioned into two child processes. First, the filtered data is *forward filled* by propagating the last valid observation forward in order to address missing values between valid observations. In Module 3.2, observations at the start of the data set with missing values are removed.

4.3 The modelling component

The modelling component comprises three modules numbered 4.0 to 6.0 in continuation of the numbering in the processing component, as illustrated in the Level 0 DFD in Figure 4.1. The objective of the respective modules in the modelling component is to transform and model the data for the subsequent component, "analysis".

Modules 4.0 and 5.0 are employed to develop ML models for FTS modelling, while Module 6.0 trains and evaluates the performance of the ML models. More specifically, Module 4.0 transforms the data and generates additional features in a process referred to as *feature engineering*. The purpose of Module 4.0 is to create data sets that are modelled in Module 6.0. In Module 5.0, the ML model employed on the engineered data set is chosen from the data store D2, also known as ML models. The ML models data store is comprised of the ML models described in Section 3.3 and Section 3.4, namely linear regression, SVR, ARIMA, MLP, RNN, LSTM and GRU. The linear regression model is the baseline model for this dissertation. Modules 6.0 is repeated for all of the ML models in the ML data store.

A more detailed representation of the modelling component is given by the Level 1 DFD in Figure 4.3. In this diagram, the process of feature engineering (Module 4.0) is expanded into three child processes,

namely data transformation, feature generation, and feature selection. In Module 4.1, the feature 'Price' is transformed into the percentage change between one day and the next in order to fulfill the scope and assumption requirements in Section 1.3. In addition, the module allows for the transformation of data according to the input format requirements of the chosen ML model (*i.e.* data should be stationary or normalised).

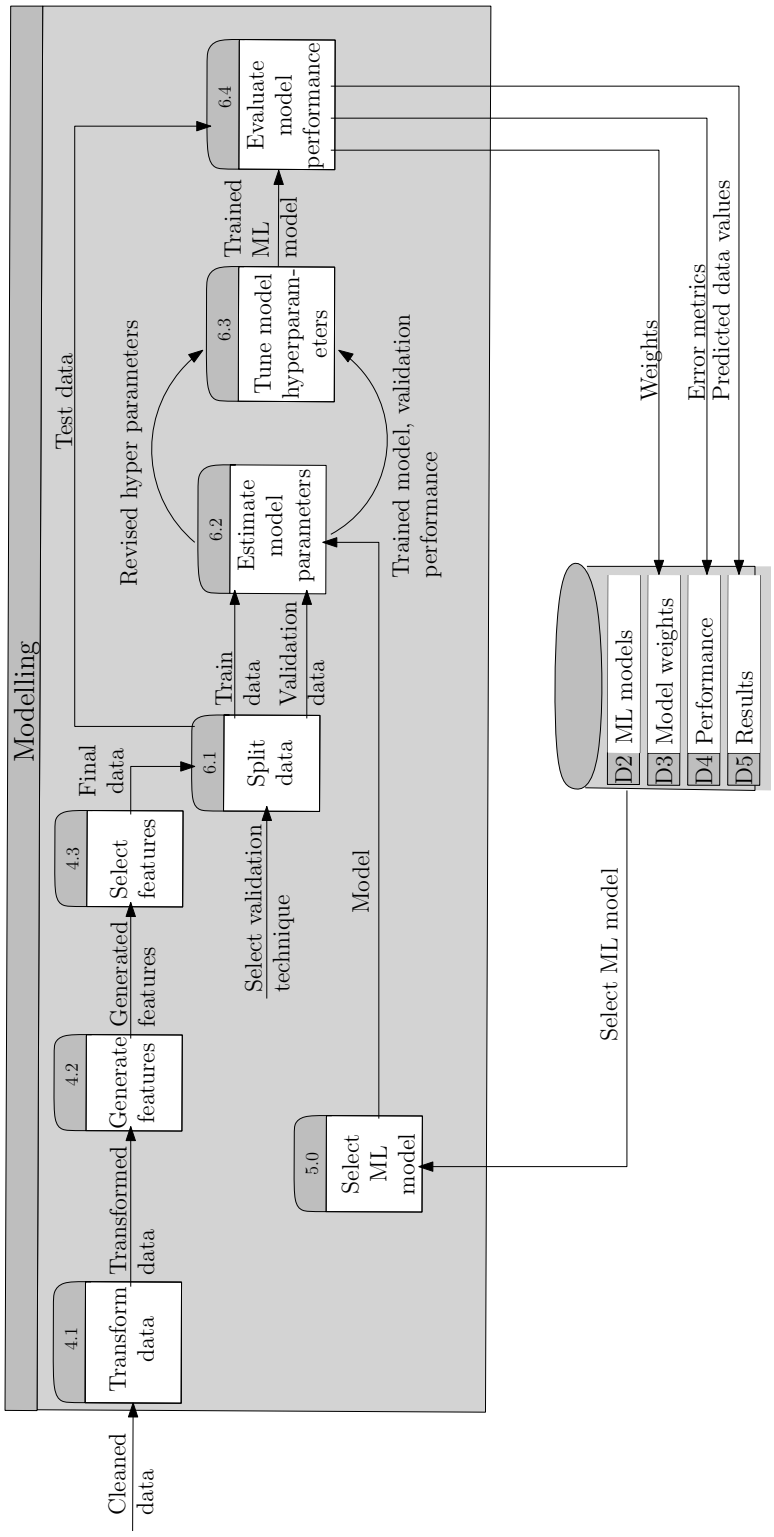


FIGURE 4.3: *Level 1 DFD of the modelling component.*

Module 4.2 generates additional features from which the ML model may learn. As mentioned in the scope and assumption requirements in Section 1.3, the daily returns for the previous two days are utilised to predict the current return. Consequently, two additional features are created from the feature for the current day x_t, namely x_{t-2} and x_{t-1}. The features x_{t-2} and x_{t-1} are the percentage change data from the previous two and one days, respectively. The current data set is comprised of three features, namely x_{t-2}, x_{t-1} and the current day x_t. The features x_{t-2} and x_{t-1} are selected as input features, while the feature x_t is selected as the output feature in Module 4.3.

Module 5.0 is not expanded into child processes, because a simple process is represented. Module 6.0 is expanded into four child processes aimed at splitting the data into smaller subsets, estimating the parameters of the model, tuning the model hyperparameters, and evaluating the performance of the model. In Module 6.1, the data is split into a training, validation and testing set based on the validation requirements outlined in Section 3.2.2. The validation techniques employed are the single out-of-sample technique and the walk-forward technique. The single out-of-sample technique is selected as the baseline validation technique, because it is a widely applied technique in literature for FTS forecasting. The walk-forward technique is considered a robust approach if not the best in TS modelling and is therefore selected. The input data for Module 6.1 is employed for both validation techniques, and therefore, the module is repeated twice for every financial asset.

The curved arrows between Modules 6.2 and 6.3 indicate that the process is executed iteratively. For each selected ML model in Module 5.0, the model parameters are estimated for a fixed set of hyperparameters in respect of the training data set in Module 6.2. The model performance is then evaluated in respect of a validation data set, and the values of the hyperparameters are adjusted accordingly in Module 6.3. The hyperparameter tuning process in Module 6.3 is accomplished by means of a manual search, grid search, random search, Bayesian optimisation or the F-race algorithm as described in Section 3.2.1. This dissertation, however, only employs the random search technique. Furthermore, a range of hyperparameters to explore are specified during the algorithm selection in Module 5.0.

Module 6.4 generates three data related outputs, namely *Weights*, *Error metrics* and *Predicted data values*, during the process of evaluating the performance of each algorithm. The weight data contains the optimal weights, parameter and hyperparameter information for each algorithm. The weight data is stored in data store D3, also known as Model weights. The weight data is used to load previously trained models, rather than training the models from scratch. Models that do not have model weights, were serialised in order to avoid retraining from scratch. Even though the weights data is stored, it will not be utilised in the remainder of the FTSEP, because the data does not provide additional insights into comparing and contrasting the capabilities of different ML models. The data store D4, also known as Performance, stores performance error metrics obtained by the algorithm. The error metrics that are stored are the MSE, RMSE, MAE and accuracy. The actual and predicted values of the test data set are stored in data store D5, also known as Results.

4.4 The analysis component

The analysis component comprises three modules numbered 7.0 to 9.0 in continuation of the numbering in the modelling component, as illustrated in the Level 0 DFD in Figure 4.1. The modules are designed to analyse the data returned by the evaluated algorithm to enrich the process of identifying a generic approach to FTS modelling.

Module 7.0 filters the data from the data stores Performance and Results on the validation technique selected. Modules 8.0 and 9.0 analyses the error metrics and visualises the predicted data values. The capabilities of ML models based on the types of FTS and type of validation technique selected is then compared.

A more detailed representation of the analysis component is given by the Level 1 DFD in Figure 4.4. Module 7.0 is not expanded into child processes, because a simple process is represented. Module 8.0 is expanded into two child processes. The data filtered on the chosen validation technique selected in Module 7.0 is tabulated in Module 8.1.

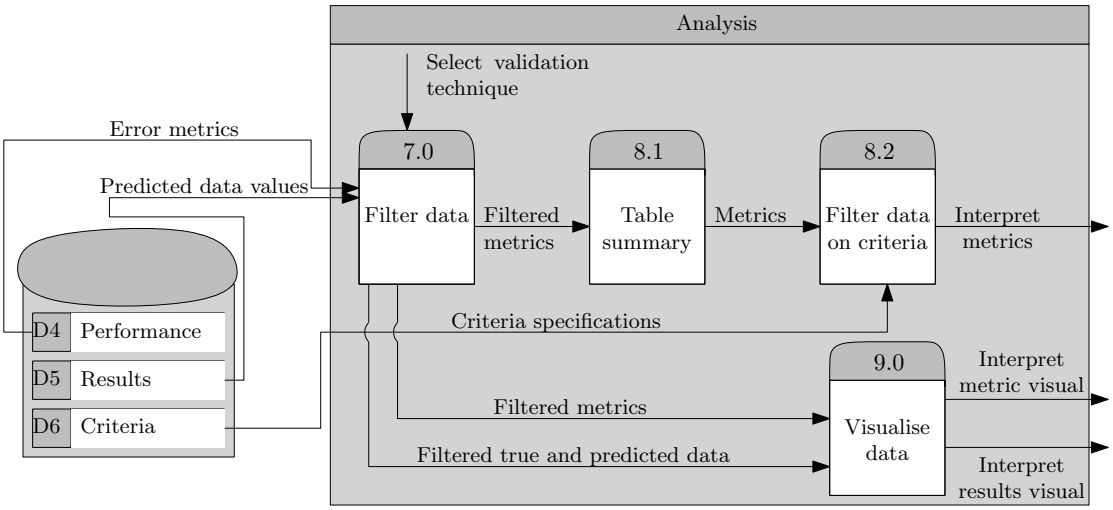


FIGURE 4.4: *Level 1 DFD of the analysis component.*

If the single out-of-sample technique is selected, the filtered metrics are tabulated without any adjustments. If the walk-forward validation technique is selected, Module 8.1 receives multiple MSE, RMSE, MAE and accuracy metrics for each ML model and asset. In order to gain insight from the metrics, the mean is found for each performance metric in Module 8.1 and tabulated. The tabulated summary provides insight into the average performance of a ML model over time.

Module 8.2 filters the tabulated data based on the criteria selected in data store D6, also known as Criteria. The criteria in the data store act as a specification for filtering the data, which in turn enables the analysis of *individual* assets and a comparison and analysis between *all* the assets (*i.e.* general comparison). The individual and general comparison of assets are based on employing the performance metrics MSE, RMSE, MAE and accuracy. The criteria for studying individual assets and all the assets are as follows:

Asset specific criteria

1. ML model to ML model comparison for each financial asset, based on validation technique.
2. Compare the baseline ML model, linear regression, to each ML model for each financial asset, based on validation technique.
3. Compare the performance of the neural network ML models to all of the other ML models for each financial asset, based on validation technique.

General asset criteria

1. Comparison of each ML model for *all* the financial assets, based on validation technique.
2. Compare the baseline ML model, linear regression, to each ML model for *all* the financial assets, based on validation technique.
3. Compare the performance of the neural network ML models to all of the other ML models for *all* the financial assets, based on validation technique.

The filtered metrics and filtered true and predicted data from Module 7.0 are received as input, in Module 9.0. The purpose of Module 9.0 is to visualise data to enable the user to gain additional insights or identify interesting patterns. Only the filtered true and predicted data values are visualised if the single out-of-sample technique is selected in Module 7.0. A visualisation of the predicted and actual data provides insight into the performance of the model in the single testing set. An analysis of where the model struggled and where it did not can therefore be done. The metrics data for the single out-of-sample technique is not visualised, because a visualisation does not provide any additional information compared to the table summary generated in Module 8.1.

However, for the walk-forward validation technique, only the metrics data are visualised. A visualisation of the metrics data *over time* provides insight into the error made by a specific ML model over a long period. The visualised data are then utilised in conjunction with the data in Module 8.2.

4.5 Chapter summary

This chapter introduced the empirical process FTSEP followed in this dissertation to forecast future price movements of financial assets. The chapter opened with a high-level description of the empirical process FTSEP. The input data assumptions were clearly stated, and then the FTSEP was illustrated by means of a DFD. The FTSEP comprises three components, namely processing, modelling, and analysis, which were described in a concise manner.

Tasks performed by the processing component in the FTSEP include data formatting, data filtering and data cleaning. The processed data is then employed in the modelling component, which performs tasks related to model building and model evaluation. Subsequently, the analysis component extracted valuable information and insights from the generated data in the modelling component. Detailed descriptions of the three components followed to provide a complete understanding of the FTSEP with additional DFDs.

CHAPTER 5

Case study implementation

Contents

The purpose of this chapter is to implement Modules 1.0 to 6.4 based on the FTSEP described in Figure 4.1. The TS data utilised for this implementation are the S&P 500 index, the bond yield, the USD/ZAR currency pair, gold, and Bitcoin. The chapter opens with an implementation of the necessary data transformations outlined in Modules 1.0 to 4.0. The focus then shifts to the configuration of the machine learning model architectures and associated input data requirements considered in Module 5.0. An outline of the data set sizes based on the validation technique chosen in Module 6.1 follows. Finally, an implementation of the hyperparameter selection, training procedure and performance evaluation for each selected ML model is described, in fulfilment of the requirements outlined in Modules 6.2 to 6.4.

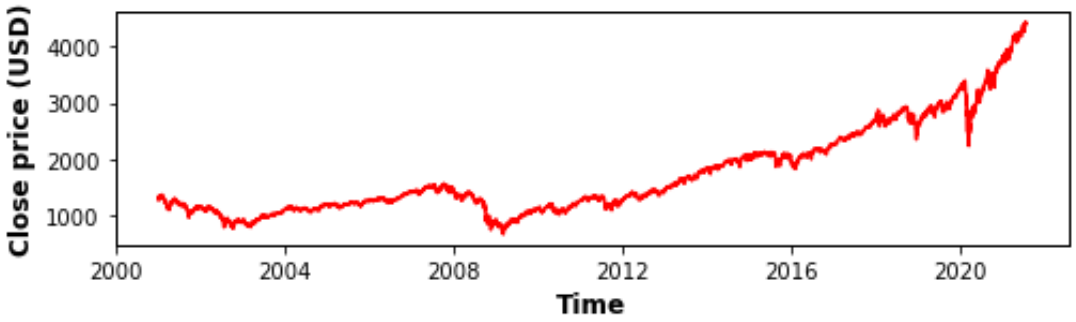
5.1 Data processing and transformations

The data for each of the financial assets employed in this study were obtained from Investing.com. As stipulated in Module 1.0, each asset was sorted in ascending order by date, and the feature '*Close*' was classified as a quantitative data type. With the exception of the Bitcoin data set, the data set size for each financial asset is approximately 20 years, as documented in Table 5.1. Bitcoin was only introduced in 2012 and therefore contains approximately nine years of data. The last column in the table provides the data set size for each asset in days. The financial markets are closed on weekends and public holidays, and therefore data for these days in the specified date ranges are not available.

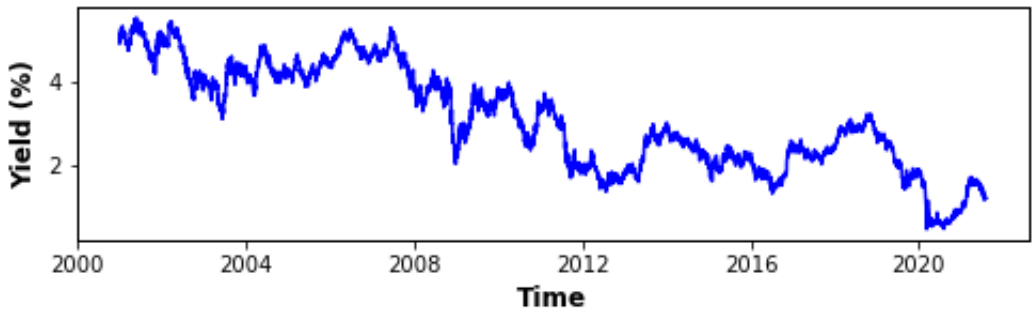
TABLE 5.1: *Data set size for each financial asset.*

Financial asset data set size			
Asset	**Date start**	**Date end**	**Number of records**
S&P 500 index	2002/01/02	2021/06/01	4921
Bond yield	2002/01/02	2021/06/01	4921
USD/ZAR	2002/01/02	2021/06/01	4921
Gold	2002/01/02	2021/06/01	4921
Bitcoin	2012/01/02	2021/06/01	3466

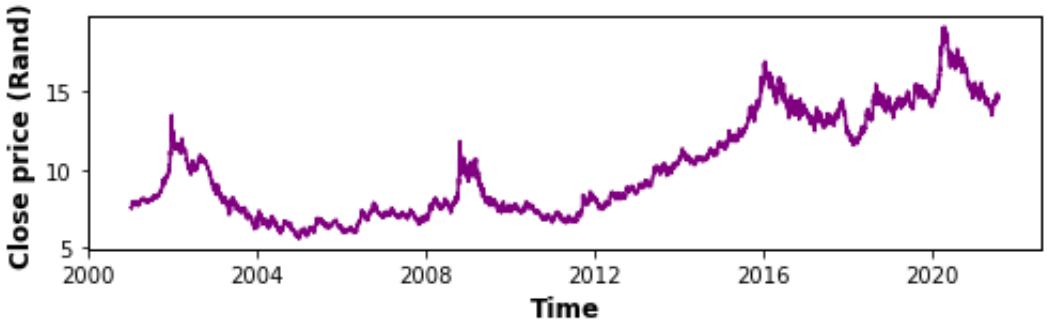
The data set was filtered based on the type of asset selected (Module 2.0). The missing values in each asset are either filled or removed, as per the requirements of Module 3.0. The cleaned data set for each asset, after addressing the missing values, are plotted in Figure 5.1.



(a) *S&P 500 index close price time series*

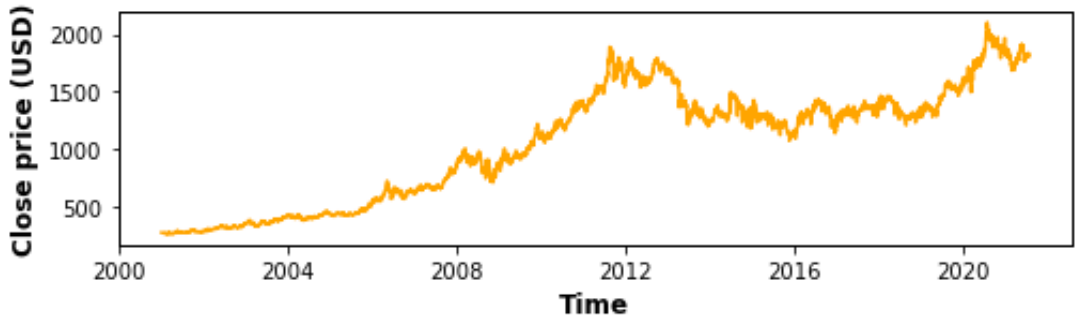


(b) *Bond yield time series*

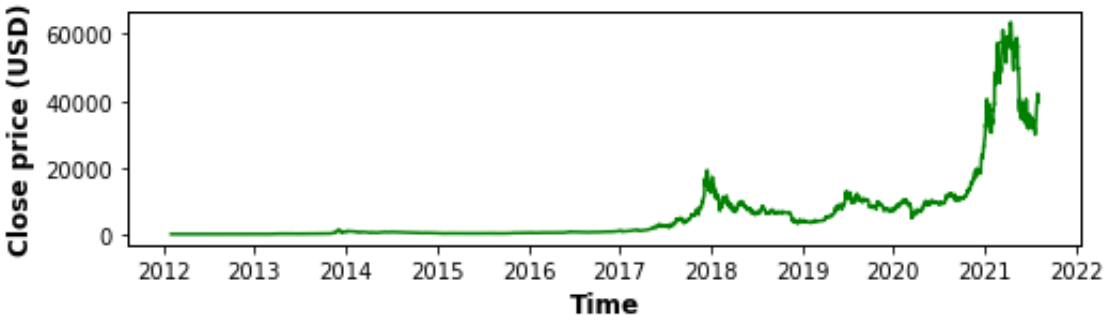


(c) *USD/ZAR close price time series*

FIGURE 5.1: *The close price time series for the S&P 500 index, bond yield, USD/ZAR, gold and Bitcoin, for the date ranges specified in Table 5.1.* (cont.)



(d) *Gold close price time series*



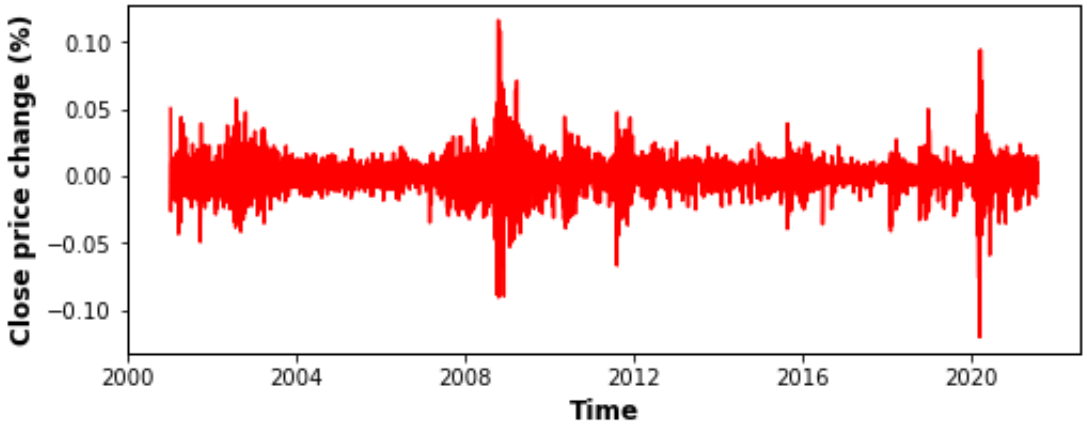
(e) *Bitcoin close price time series*

FIGURE 5.1: *The close price time series for the S&P 500 index, bond yield, USD/ZAR, gold and Bitcoin, for the date ranges specified in Table 5.1.*

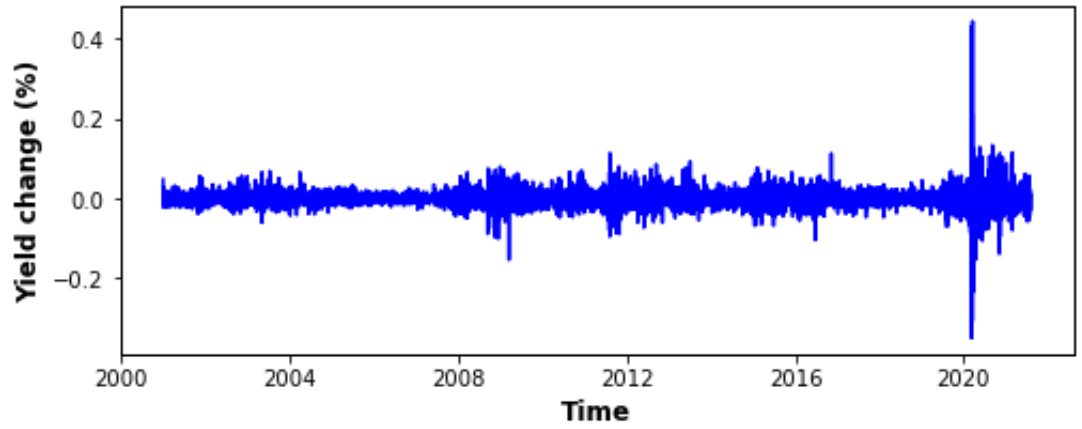
In Figure 5.1 (a), the S&P 500 index follows a steady upward trend, with an almost linear decrease and increase in close price for the year 2020. The large decrease in trend observed in 2008 and 2020 is attributed to the global financial crises and the COVID-19 pandemic, respectively. The bond yield in Figure 5.1 (b) follows a general downward trend. However, the trend is more jagged compared to the increasing trend of the S&P 500 index. In Figure 5.1 (c), the USD/ZAR currency pair TS indicates that there are multiple increasing and decreasing trends over the 20 year period. The gold TS in Figure 5.1 (d) shows a gradual increase in trend from the year 2001 to 2012. After the year 2012, the trend moves in a side-ways direction for the remaining period. The Bitcoin TS in Figure 5.1 (e) is different from the previous financial asset TS illustrations. There is no visible trend from 2012 to 2017. In 2017, a significant increase in trend was observed. The trend then decreases again gradually. The same phenomenon is observed for the year 2021. Furthermore, Figure 5.1 shows that the data for each financial asset are very different from one another. As a result, it can be assumed that selection bias is prevented. This assumption can only be confirmed when the data is transformed in Module 4.1.

The cleaned data moves from the processing component to the modelling component. In the modelling component, Module 4.1 is executed by transforming the '*Close*' price feature into a new feature named '*Change %*'. The '*Change %*' feature indicates the daily percentage change for a financial asset. The transformed data set for each asset is plotted in Figure 5.2.

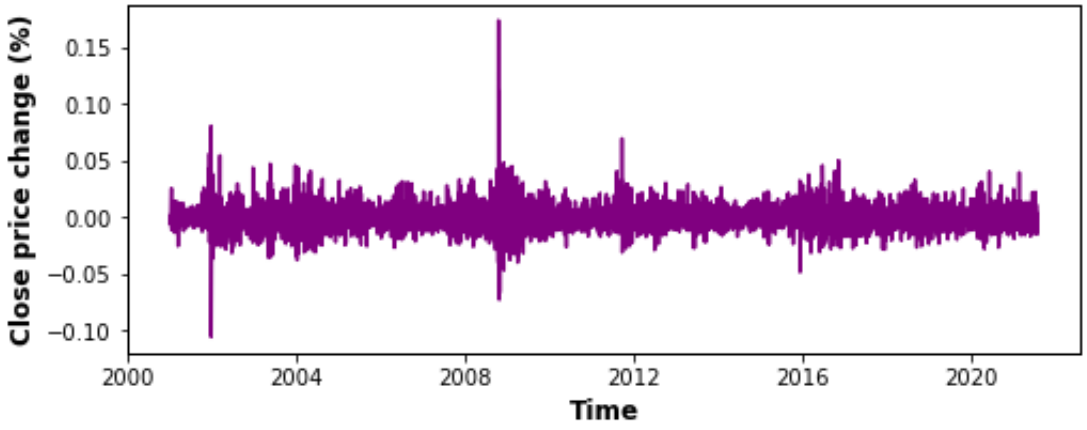
The S&P 500 index in Figure 5.2 (a) experienced significant percentage price changes during the market crashes in 2008 and 2020. The percentage changes observed for these periods are between -10% and 10%. For the other periods, however, the percentage change prices are on average between -3% and 3%. On the contrary, the bond yield in Figure 5.2 (b) shows the yield change percentage was relatively stable and small between the years 2001 and 2020. The yield percentage range for this period was between -2% and 2%. A significant yield change percentage between -20% and 40% is observed for the year 2020.



(a) *S&P 500 index close price change (%) TS*



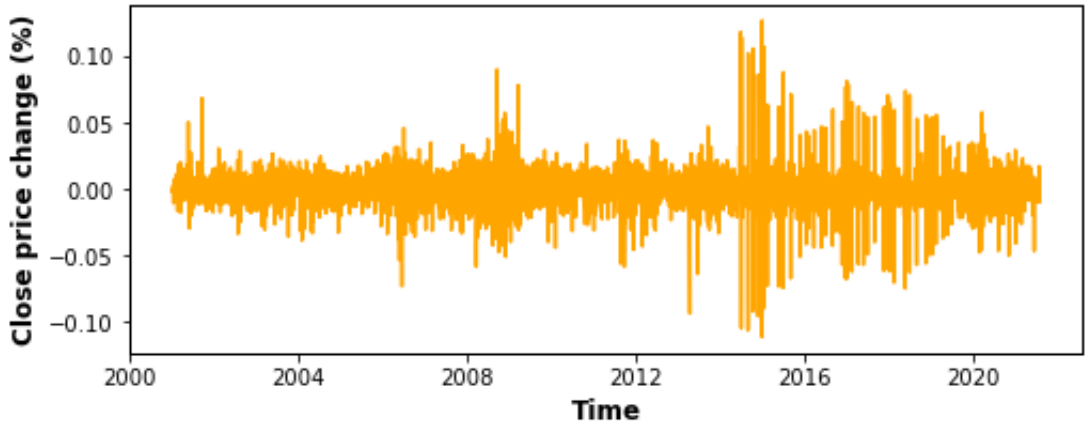
(b) *Bond yield change (%) time series*



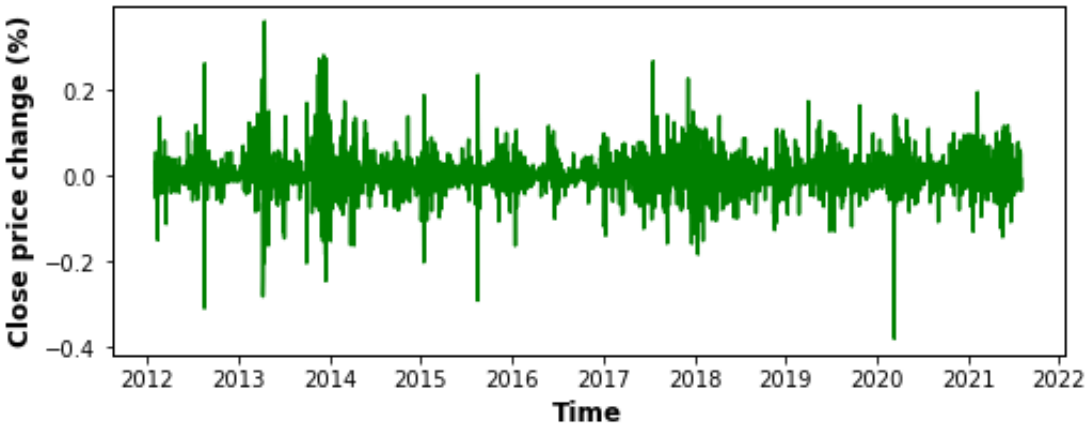
(c) *USD/ZAR close price change (%) time series*

FIGURE 5.2: *The close price change (%) time series for the selected financial assets.* (cont.)

The USD/ZAR currency pair plot in Figure 5.2 (c) shows that for most of the 20 year period under investigation, the close price percentage was relatively the same (*i.e.* between -5% and 5%). Significant percentage changes only occurred in 2002 and 2008 for a small period. The gold percentage change price in Figure 5.2 (d) shows significant volatility between 2015 and 2020. The gold price percentage



(d) *Gold close price change (%) time series*



(e) *Bitcoin close price change (%) time series*

FIGURE 5.2: *The close price change (%) time series for the selected financial assets.*

change ranged between -10% and 10%, and -5% and 5% for the five year period. The Bitcoin TS in Figure 5.2 (e) indicates that the close price percentage change was volatile over the entire period. On average, the percentage change ranged between -10% and 10%. An extract of the data utilised for each asset in Figures 5.1 and 5.2 are provided in Appendix A. Furthermore, Figure 5.2 shows that the close price percentage plot for each asset is substantially different. As a result, the assumption that selection bias is prevented with the employment of the different types of financial TS data sets is confirmed.

For the execution of Module 4.2, two additional features for each financial asset were generated and named 'x_{t-1}' and 'x_{t-2}'. The features refer to the number of lagged periods. For example, the feature 'x_{t-1}' refers to a one-day lag period, while the feature 'x_{t-2}' refers to a two-day lag period. Per the requirements of Module 4.3, the newly created features are set as the input variables, while the feature '*Change (%)*' is set as the output variable. An extract of the newly created data set for each asset is provided in Appendix B.

5.2 Model requirements

This section provides an overview of the range of hyperparameter values that can be chosen and the pre-determined architecture for each model under investigation, as per the requirements of Module 5.0. Subsequently, a brief discussion on the input data requirements for the transformed TS, based on the selected ML model follows. Finally, the data set sizes for each validation technique chosen in Module 6.1 are outlined. As mentioned previously, the only validation techniques employed in this dissertation are single out-of-sample validation and walk-forward validation.

5.2.1 Model parameters and architectures

Each model has hyperparameter or architectural requirements that are considered when implementing Module 5.0. The random search technique is employed in order to determine the optimal hyperparameters. Table 5.2 documents the parameter requirements of the ARIMA and SVR models and a range of values that could be chosen as the parameter value.

TABLE 5.2: *Range of hyperparameters for each model.*

Model hyperparameters		
Model	**Parameter**	**Range**
ARIMA	p	[0,3]
	q	[0,3]
	d	[0,3]
SVR	C	0.001, 0.01, 0.1, 1, 10, 100
	Kernel	Linear, Polynomial, RBF

The hyperparameter definitions for the ARIMA model were discussed in Section 3.3.2. For the SVR model, the parameter C is also known as the *soft margin*. The margin increases by ignoring points close to the boundary when the value of C is small. On the other hand, the margin decreases when a large value for C is chosen because a large penalty is assigned to errors and margin errors. In addition, the decision boundary is affected by the kernel that can either be linear, polynomial or a *radial basis function* (RBF).

The ARIMA, SVR and linear regression models do not require a pre-determined model architecture. On the other hand, all the investigated neural networks require a pre-determined model architecture. The network architecture employed for all neural networks in this dissertation is documented in Table 5.3.

TABLE 5.3: *Neural network architecture for each neural network under investigation in this dissertation.*

Neural network architecture	
Architecture	**Value**
Layers	3
Number of hidden neurons	128
Activation function	ReLu
Regularisation	None
Dropout	None
Early stopping	50
Learning rate	0.001
Batch size	64
Epochs	2000

5.2.2 Data requirements

The ARIMA model requires data to be in a stationary format. Stationarity is tested with the ADF test discussed in Section 3.3.2. Since the null hypothesis assumes the presence of unit root, that is $\alpha = 1$, the p-value obtained should be less than the significance level in order to reject the null hypothesis. Thereby, inferring that the series is stationary. The ADF results generated for each financial asset are provided in Table 5.4.

TABLE 5.4: *Augmented dickey fuller test for testing stationarity*

Augmented dickey fuller test results								
Asset	**Statistics**		**Critical values**			**Is the data stationary based on critical values (confidence)**		
	Test statistic	**P-value**	**1%**	**5%**	**10%**	**99%**	**95%**	**90%**
S&P500	-13.44	0.99	-3.43	-2.86	-2.56	Yes	Yes	Yes
Bond	-15.22	0.00						
USD/ZAR	-21.18	0.75						
Gold	-32.37	0.00						
Bitcoin	-9.67	0.00						

Table 5.4 shows that, for each asset, the test statistic is smaller than each of the critical values (1%, 5% and 10%). The null hypothesis is therefore rejected, signifying that each TS is stationary. As a result, each TS does not have to undergo further transformations and can be utilised immediately in the ARIMA model. For the linear regression, SVR and neural network input data requirements, each data set was normalised to a range between -1 and 1.

5.2.3 Data set size

Module 6.1 requires the data to be split into a training, validation and testing data set in order to evaluate the performance of each model. For the single out-of-sample tests, the raw data for each asset is split into a training, validation and testing set in the ratio 70:20:10. The separation is illustrated in Table 5.5. The training, validation, and test data set sizes for Bitcoin differ because the complete data set size is smaller than the other assets.

TABLE 5.5: *Data set size for each financial asset.*

Data set sizes in number of days			
Assets	**Set name**		
	Training set	**Validation set**	**Testing set**
Bitcoin	2426	693	347
All other assets	3445	984	492

For the walk-forward validation implementation, the block period for each iteration k is one year, corresponding to 251 trading days. The testing data set size is 15 days, which corresponds to 3 weeks. The validation set size is 20 days. The difference between the block period and the sum of the validation and test sets is the number of days in the training set. The size of the training set will therefore be 216 days.

The linear regression, ARIMA and SVR models do not require a validation set, and therefore only the training and testing data sets are employed. On the contrary, the remaining ML models employ a training, validation and testing data set.

5.2.4 Model implementation

After selecting the range of hyperparameter values and the data set sizes based on the selected validation technique in Module 6.1, the training and validation data sets are received as input in Modue 6.2. The linear regression, ARIMA and SVR models do not utilise a validation data set, and therefore only a training data set is received as input in Module 6.2.

Modules 6.2 and 6.3 were executed iteratively in order to find the best hyperparameters. The performance of each ML model on each financial asset was evaluated in Module 6.4. The weights, error metrics and predicted data values were stored in the Model weights, Performance and Results data stores, respectively. As mentioned previously, the weights are used to load previously trained models rather than training the models from scratch. Models that do not have model weights were serialised in order to avoid retraining from scratch. However, the Model weights data store is not investigated in this dissertation. The error metrics stored in the Performance data store include the MSE, RMSE, MAE and accuracy. The MSE, RMSE and MAE have similar methods of calculation and therefore, only the MSE performance metric is employed in the subsequent modules.

A simple strategy is implemented to calculate the accuracy to evaluate whether each ML model correctly forecasts when to enter and exit the market. The strategy comprises two actions, of which one is selected for every forecasted data point:

1. If the predicted price is *greater* than the actual price, then the asset should be *bought* (go long), and a variable named strategy takes the value of 1.
2. If the predicted price is *smaller* than the actual price of today, then the asset should be *sold* (go short), and a variable named strategy takes the value of -1.

The accuracy metric records the number of times that the strategy was in agreement with the actual price movements. For example, if the chosen strategy is 1 and the actual price movement is positive, then the strategy was correct and the number of right guesses increases by 1. However, if the chosen strategy is 1 and the actual price movement is negative, then the strategy is incorrect.

If the final accuracy of a model is below 50%, the performance capability of the model is inadequate. Stated differently, the performance of the model is worse than random guessing. Therefore, one would be better off randomly guessing whether the percentage change of an asset will be positive or negative. On the other hand, if the final accuracy of a model is above 50%, the performance capability of the model is adequate. Therefore, the model has the potential to beat the market since the accuracy performance is better than random guessing. It is important to note that any accuracy above 50% is substantial. For example, the returns obtained by the company Renaissance Technologies, discussed in Chapter 1. The company obtained an average return of 40.18%, outperforming the market return of 12.43%. Renaissance Technologies claims to only have a 1% edge on the market, meaning that their methodology has a 51% accuracy [128].

5.3 Chapter summary

Modules 1.0 to 6.4 of the FTSEP described in Figure 4.1 was executed, in this chapter, . The chapter starts with an implementation of the processing component, namely Modules 1.0 to 3.0. The five financial asset TS were sorted, filtered and cleaned according to the outlined specifications of the modules. Subsequently, Module 4.1 was implemented by transforming the close price feature to a new feature, '*Change %*'. The transformed feature for each financial asset was visualised and discussed. The visualisations confirmed that the problem of selection bias had been mitigated. New features were generated as per the requirements of Module 4.2. An implementation of Module 4.3 followed, where the input and output features were selected.

Modules 5.0 and 6.1 were implemented in Section 5.2. An overview of the hyperparameter values that can be chosen and the pre-determined architecture for each ML model was documented upon the implementation of Module 5.0. A brief discussion followed on the input data requirements for the transformed TS based on the different ML models. Module 6.1 was implemented with a brief discussion of the data set sizes for each validation technique. Finally, the implementation of Modules 6.2 to 6.4 to the FTS data and the selected ML models were described.

CHAPTER 6

Case study results and analysis

Contents

The purpose of this chapter is to implement Modules 7.0 to 9.0 from the analysis component based on the FTSEP described in Figure 4.1. The performance results of the ML models are trusted, because the necessary precautions to avoid underfitting and overfitting were taken in Module 5.0. The chapter opens with an implementation of Modules 7.0 to 9.0 on the data filtered for the single out-of-sample validation technique. A table summary is generated and interpreted according to the criteria question selected from the Criteria data store, discussed in Section 4.4. The following section implements Modules 7.0 to 9.0 on the data filtered for the walk-forward validation technique. In this section, a table summary is also generated, visualised and interpreted. Subsequently, the single out-of-sample technique is compared and contrasted with the walk-forward validation technique based on the generated table summaries. Finally, a summary of the key findings of the comparison between ML models with single out-of-sample validation and walk-forward validation is provided.

6.1 Single out-of-sample validation

Data from the Performance and Results data stores were filtered on the single out-of-sample technique selected in Module 7.0. Table 6.1 summarises the metric results from the Performance data store without any adjustments in fulfilment of Module 8.2. The table includes the MSE and accuracy performance metrics for each ML model and type of financial asset.

TABLE 6.1: *Performance evaluation of machine learning models employing single out-of-sample validation.*

Performance evaluation of machine learning models employing single out-of-sample validation								
Asset	**Model/ Metric**	**LR**	**ARIMA**	**SVR**	**MLP**	**RNN**	**LSTM**	**GRU**
S&P 500 index	MSE	0.0002580	0.0002584	0.0002596	0.0003415	0.0003970	0.0003299	0.0003206
	Accuracy (%)	54.25	51.54	48.84	**57.92**	41.89	**55.41**	**55.02**
Bond yield	MSE	**0.0027859**	0.0027860	0.0028050	0.0036454	0.0035730	0.0036159	0.0032489
	Accuracy (%)	47.89	48.06	47.36	47.01	47.18	44.89	47.01
USD/ZAR	MSE	0.0000943	**0.0000943**	**0.0000942**	**0.0000936**	**0.0000941**	**0.0000961**	**0.0000955**
	Accuracy (%)	50.65	51.21	48.04	51.77	48.42	46.93	47.49
Gold	MSE	0.0001472	0.0001471	0.0001463	0.0001452	0.0001517	0.0001576	0.0001640
	Accuracy (%)	53.51	**53.51**	50.85	48.58	**52.94**	51.42	49.91
Bitcoin	MSE	0.0016884	0.0033383	0.0016936	0.0018790	0.0019116	0.0018585	0.0018832
	Accuracy (%)	**54.47**	44.96	**54.47**	55.04	46.97	49.28	48.70

To gain an understanding of the scale of the data tabulated in Table 6.1, the predicted and true data values from the Performance data store for the linear regression model applied to the S&P 500 index is visualised in Figure 6.1. This figure shows a substantial difference between the actual and predicted values for the S&P 500 index. The MSE of 0.000258 in Table 6.1 is therefore considered large.

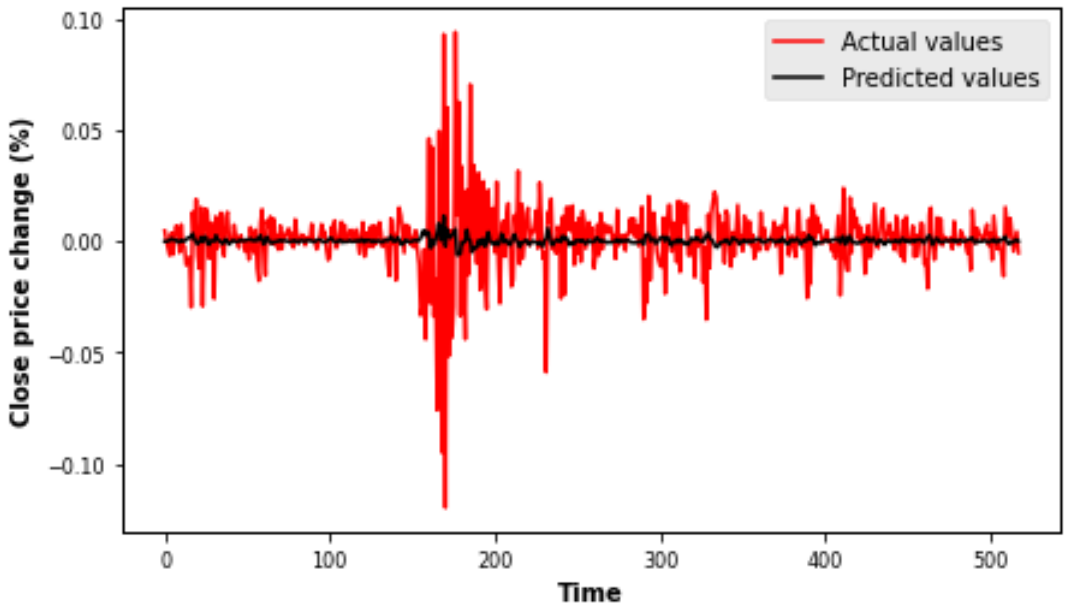


FIGURE 6.1: *Comparison between actual and predicted values for a linear regression model applied to the S&P 500 index data set.*

The remainder of this section investigates, analyses and interprets the performance metrics for each ML model in Table 6.1. The analysis is completed by comparing and contrasting the table summary according to each criterion from the Criteria data store in Module 8.2. The criteria are addressed in two parts: The first part is dedicated to the 'Asset specific criteria' group, and the second part addresses the 'General asset criteria' group.

6.1.1 Asset specific criteria analysis

The three criteria outlined in the 'Asset specific criteria' in Section 4.4 are investigated, analysed and discussed per financial asset in this subsection.

S&P 500 index

It is observed from Table 6.1 that the smallest MSE for the S&P 500 index is with linear regression and the largest MSE is with the RNN. The baseline model outperformed all the other ML models in terms of MSE. However, the ARIMA and SVR models had similar MSEs to linear regression. Based on this observation, ARIMA and SVR model the data in a similar manner to the linear regression model. The group of neural networks tested did not outperform the other ML models in terms of MSE.

The accuracy performance metrics show that the MLP had the highest accuracy of 57.92%. With an accuracy above 50%, it is concluded that the structure of the MLP is favourable towards the S&P 500 index TS. On the contrary, RNN and SVR are considered unfavourable models because both models had an accuracy below 50%. Furthermore, the baseline model could not outperform the other ML models in terms of accuracy. However, the baseline model, with its simpler structure still had an accuracy above 50%. As a result, the baseline model is considered a suitable model for TS like the S&P 500 index.

When considering the MSE and accuracy metrics for each model, the following observations and insights are gained: The baseline model is a strong contender in terms of MSE and accuracy (above 50%). The MLP had the highest accuracy and the second-highest MSE, confirming that error metrics by themselves do not provide an absolute evaluation. Furthermore, the RNN is not a well-suited model for forecasting the future price movement of a TS similar to the S&P 500 index. RNN had the highest MSE, indicating that it struggled to model the data, and it had the lowest accuracy.

Bond yield

The smallest MSE for the bond yield is with linear regression and the largest MSE is with the MLP, as shown in Table 6.1. The difference in MSEs between linear regression, ARIMA and SVR are significantly small. As a result, the three models are considered to model the bond yield data set in a similar manner. However, the baseline model still outperformed the other models. The group of neural networks tested did not outperform the other models.

The accuracy performance metrics show that all the models had an accuracy below 50%. As a result, all the models are considered unfavourable models when applied to the bod yield data set. ARIMA had the highest accuracy of 48.06%, while the LSTM had the lowest accuracy of 44.89%. Furthermore, the baseline model could not outperform the other ML models. However, the accuracy difference between the baseline model and ARIMA is considerably small. Taking into account that the baseline model has a simpler structure, it is considered to outperform ARIMA and the other ML models. The group of neural networks tested could not outperform the other ML models in terms of accuracy. When considering the MSE and accuracy metrics for each model, it is concluded that all the investigated models are unfavourable models for the bond yield data set.

USD/ZAR currency pair

It is observed from Table 6.1 that the smallest MSE for the USD/ZAR currency pair is with the MLP and the largest MSE is with the LSTM. The difference between the lowest and highest MSEs for the models is significantly small, leading to the assumption that the models model TS data like the USD/ZAR currency pair in a similar manner. The baseline model was not able to outperform the MLP. However, since the MSEs are very similar, an argument can be made that the baseline model is the best performing model. Linear regression has a simpler structure and is more computationally efficient than MLP. Two of the four neural networks outperformed the other ML models. The neural networks are MLP and RNN.

The accuracy performance metrics show that the MLP had the highest accuracy of 51.77% and the LSTM the lowest accuracy of 46.93%. The MLP architecture is considered favourable towards the USD/ZAR currency pair. On the contrary, the LSTM is considered an unfavourable model, because the model had the lowest accuracy below 50%. The baseline model could not outperform the other models. However, the model has a simpler structure and an accuracy above 50%. As a result, the baseline model is still considered a suitable model for TS, like the USD/ZAR currency pair. Only the MLP from the group of neural networks was able to outperform the other ML models.

When considering the MSE and accuracy metrics for each model, the following observations and insights are gained: The MSE indicates that all the models have a similar technical capability. In terms of accuracy, the models are grouped around 50%, with the MLP, linear regression and ARIMA being the only models with an accuracy above 50%. The linear regression model is of particular interest, because of its simpler structure compared to the other two models.

Gold

The smallest MSE for the gold TS is with the MLP and the largest MSE is with the GRU, as shown in Table 6.1. The baseline model outperformed some, but not all of the ML models in terms of MSE. ARIMA and linear regression had very similar MSEs and therefore model the data in a similar manner. With the exception of the MLP model, the linear regression model outperformed the group of neural network models.

The accuracy performance metrics show that linear regression and ARIMA had the highest accuracy of 53.51%. With an accuracy above 50%, it is concluded that the structure of the linear regression and ARIMA are favourable towards the gold TS. On the contrary, the MLP is considered an unfavourable model, because the model had the lowest accuracy of 48.58%. Furthermore, the baseline model outperformed the other models.

When considering the MSE and accuracy metrics for each model, the following observations and insights are gained: The baseline model and ARIMA are strong contenders in terms of MSE and accuracy (above 50%). However, the baseline model is simpler in structure and more computationally efficient and is therefore preferred over ARIMA. The MLP had the lowest accuracy and the lowest MSE, confirming that error metrics by themselves do not provide an absolute evaluation. The GRU had the second-highest MSE and the second-lowest accuracy, which is also below 50%. Consequently, the GRU is not a well-suited model for forecasting the future price movement of a TS similar to the gold TS.

Bitcoin

It is observed from Table 6.1 that the smallest MSE for the Bitcoin TS is with linear regression and the largest MSE is with ARIMA. The baseline model outperformed all the other models in terms of MSE. The group of neural networks was only able to outperform ARIMA.

The accuracy performance metrics show that the MLP had the highest accuracy of 55.04% and ARIMA the lowest accuracy of 44.96%. With an accuracy above 50%, the MLP is favourable towards the Bitcoin TS. On the other hand, ARIMA is considered an unfavourable model. Furthermore, RNN, LSTM and GRU also had an accuracy below 50% and are therefore considered unfavourable models for the Bitcoin TS. The baseline model could not outperform the MLP. However, the baseline model had an accuracy above 50% and is therefore still considered a suitable model for TS like the Bitcoin TS.

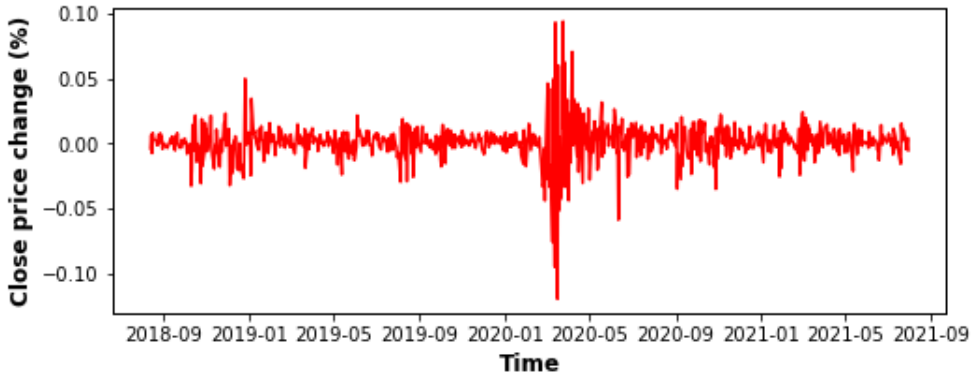
The following observations and insights are gained when considering the MSE and accuracy metrics for each model: The baseline and SVR are very strong contenders in terms of MSE and accuracy (above 50%). ARIMA had the highest MSE, indicating that the model struggled to model the data, and it had the lowest accuracy below 50%. As a result, ARIMA is considered unfavourable for TS similar to the Bitcoin TS, in general.

6.1.2 General asset criteria analysis

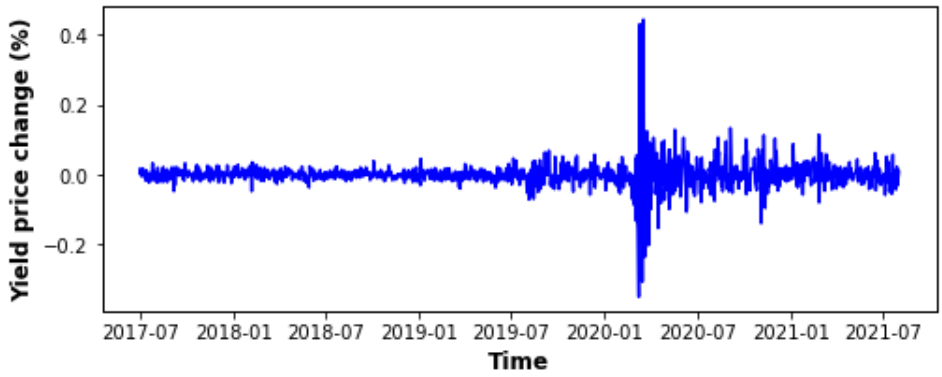
The three criteria outlined in the 'General asset criteria' in Section 4.4 are investigated, analysed and discussed per ML model in this subsection.

All models had the lowest MSE when applied to the USD/ZAR currency pair, as documented in Table 6.1. The remaining assets, ordered by MSE from lowest to highest are the gold TS, S&P 500 index, the Bitcoin TS, and the bond yield. The percentage change testing data set for each financial asset shown in Figure 6.2, provides a possible explanation for the ranks of the financial assets.

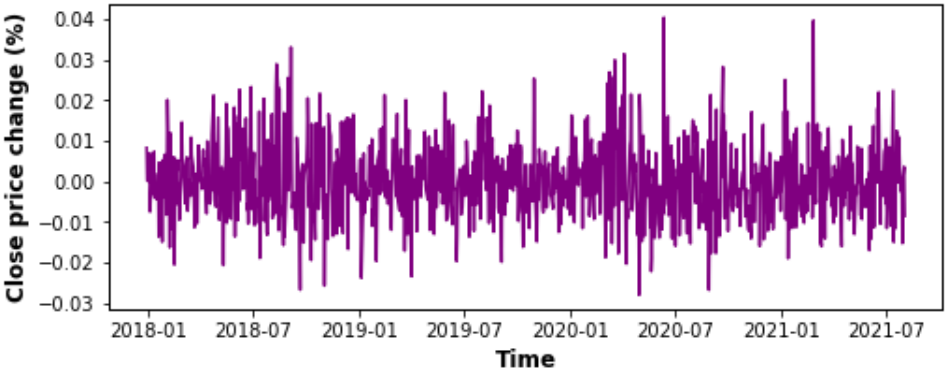
Figure 6.2(c) exhibits approximately the same amount of volatility over the entire period. As a result, the ML models are able to better forecast the future price movement of the currency pair compared to the other assets. The gold TS in Figure 6.2(d) is very similar to the USD/ZAR currency pair in Figure 6.2(c). Both figures are consistently volatile over time. Consequently, it is easier for the ML models to learn from the training data and make good predictions compared to a data set with sporadic volatilities over time. The S&P 500 index in Figure 6.2(a) exhibits a low level of volatility for the whole period, except for one instance at approximately 2020-02-26. At this period, a sporadic spike in volatility is observed. The ML models struggled in forecasting this type of TS in comparison to the currency pair or gold TS, as confirmed by the higher MSEs in Table 6.1.



(a) *S&P 500 index close price change(%) test data set time series*



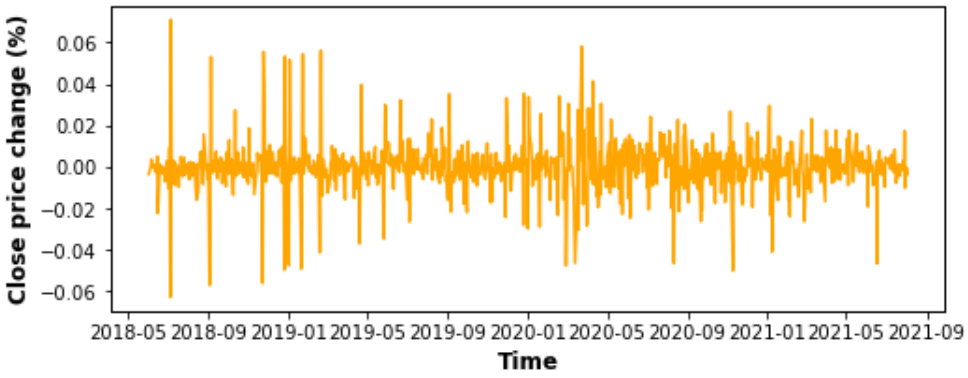
(b) *Bond yield change(%) test data set time series*



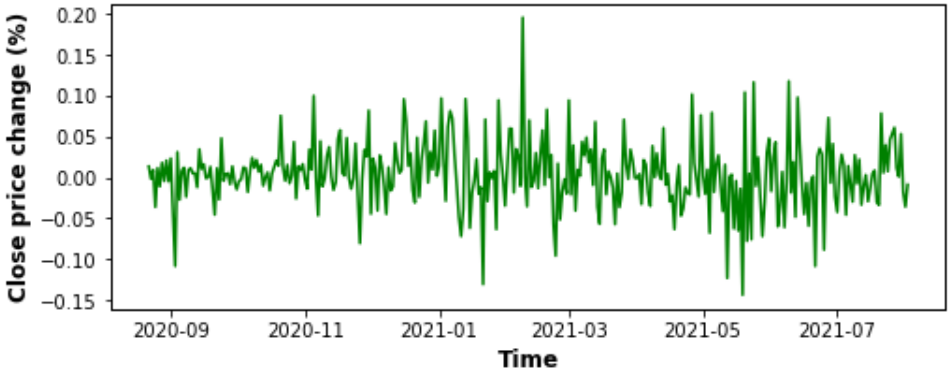
(c) *USD/ZAR close price change(%) test data set time series*

FIGURE 6.2: *The close price time series for each financial asset for the date ranges specified in Table 5.1.*(cont.)

Figure 6.2(e) provides a possible explanation as to why the Bitcoin TS had the second-highest MSE — the TS exhibits random fluctuations in volatility, where the volatility ranges between very small and very large close price percentage changes over the whole period. The ML models struggled to model this type of TS. The bond yield TS in Figure 6.2(b) exhibits a very low level of volatility for the whole period, except for one instance at approximately 2020-02-26. At this period, a sporadic spike in volatility is observed. The percentage change in volatility between the most volatile period and the volatility on average is larger for the bond yield than for the S&P 500 index.



(d) *Gold close price change(%) test data set time series*



(e) *Bitcoin close price change(%) test data set time series*

FIGURE 6.2: *The close price time series for the S&P 500 index, bond yield, USD/ZAR, gold and Bitcoin, for the date ranges specified in Table 5.1.*

Linear regression

The accuracy metric for linear regression in Table 6.1 shows that the ML model had an accuracy above 50% for all the financial assets, except for the bond yield. The linear regression model had the highest accuracy of 54.47% for the Bitcoin TS. Considering that the Bitcoin TS had the second-highest MSE, the high accuracy metric confirms that error metrics by themselves do not provide an absolute evaluation.

The following insights are gained for the linear regression model: Linear regression is well-suited for the gold TS. When applied to the gold TS, linear regression had the second-lowest MSE and an accuracy above 50%. In addition, linear regression is also considered a well-suited model for the USD/ZAR currency pair and S&P 500 index data sets. Linear regression had an accuracy above 50% and low MSEs for both data sets. For the bond yield TS, linear regression had the highest MSE and an accuracy below 50%. As a result, linear regression is not a well-suited model for the bond yield TS.

Autoregressive integrated moving average

ARIMA had an accuracy below 50% for the bond yield and Bitcoin TS. When ARIMA is applied to the bond yield TS the accuracy was 48.06% and 44.96% for the Bitcoin TS. On the other hand, ARIMA had the highest accuracy of 53.51% for the gold TS. ARIMA also had an accuracy above 50% for the S&P 500 index and USD/ZAR currency pair.

ARIMA is considered a favourable model for the gold TS, because it had the second-lowest MSE and the highest accuracy above 50%. In addition, ARIMA is also considered favourable for the S&P 500 index and currency pair. The ARIMA is not well-suited for the bond yield and Bitcoin TS, because it had accuracies below 50% and the highest MSEs.

Support vector regression

SVR is a favourable model for TS similar to the gold TS. When applied to the gold TS, SVR had the second-lowest MSE compared to when it was applied to the other financial assets. SVR also had an accuracy of 50.85%. SVR had the highest accuracy above 50% for the Bitcoin TS and the second-highest MSE. Based on this performance, the SVR is considered an unfavourable model for the Bitcoin TS. Similarly, SVR is also not considered well-suited for the bond yield, because it had the lowest accuracy of 47.36% and the highest MSE.

Multilayer perceptron

The accuracy metric for the MLP in Table 6.1 has an accuracy above 50% when applied to the S&P 500 index, Bitcoin and USD/ZAR currency pair. The MLP had an accuracy of 57.92% for the S&P 500 index, an accuracy of 55.04% for the Bitcoin TS and an accuracy of 51.77% for the currency pair TS. When applied to the bond yield, MLP had the lowest accuracy of 47.01% and the second-lowest accuracy of 48.58% for the gold TS.

The following insights are gained for the MLP: The MLP model had a low MSE and an accuracy above 50% for both the S&P 500 index and USD/ZAR currency pair. The MLP is therefore considered a favourable model for these TS. The bond yield and Bitcoin TS are the worst suited types of TS for MLP because the MLP had an accuracy below 50% and higher MSEs compared to the other financial assets.

Recurrent neural network

RNN only had an accuracy above 50% for the gold TS. The high volatilities exhibited throughout Figure 6.2(d) may have contributed to the ability of the RNN to have such an accuracy and a low MSE. The RNN is therefore considered a favourable model for highly volatile TS. The RNN is considered an unfavourable model for the remaining financial assets, because it had an accuracy below 50% and high MSE.

Long short term memory

LSTM only had an accuracy above 50% for the S&P 500 index and gold TS. When LSTM is applied to the S&P 500 index, the accuracy is 55.41% and 51.42% for the gold TS. On the other hand, LSTM had the lowest accuracy of 44.89% for the bond yield TS.

LSTM is therefore considered a favourable model for the S&P 500 index and gold TS. This is an interesting statement considering that the volatility exhibited by Figures 6.2(a) and (d) are very different from one another. The S&P 500 index has a relatively low constant volatility with an instance of a sporadic high volatility, while the gold TS has a relatively high constant volatility. LSTM was able to model the different TS adequately, to get an accuracy above 50% for both TS.

Gated recurrent unit

The accuracy metric for the GRU in Table 6.1 shows that the model only had an accuracy above 50% for the S&P 500 index. When applied to the bond yield, the GRU had the lowest accuracy of 47.01%.

The following insights are gained for the GRU: The GRU is well-suited for TS similar to the S&P 500 index. When applied to the S&P 500 index, GRU had the third-lowest MSE and the highest accuracy of 55.02%. The bond yield and Bitcoin TS are not considered as TS that are favourable to the GRU, because the model had an accuracy below 50% and the highest MSEs.

6.2 Walk-forward validation implementation

Data from the Performance and Results data stores were filtered on the walk-forward validation technique selected in Module 7.0. A summary of the metric results from the Performance data store, with adjustments, is shown in Table 6.2 in fulfilment of Module 8.1. The metrics have been adjusted so that only the mean of each performance metric is documented. Table 6.2 includes the MSE and accuracy performance metrics for each ML model and type of financial asset.

The analyses and comparisons between the ML models with walk-forward validation and financial assets follow a similar outline to the single out-of-sample validation implementation in Section 6.1.

6.2.1 Asset specific criteria analysis

For each asset, an analysis of the close price percentage change, in Figure 5.2, is conducted. The analysis includes the identification of periods with low volatility, high volatility, and sporadic volatility. Identifying different periods of volatility aids in determining whether a ML model is favourable for a certain TS during a specific period. For example, during a period of sporadic volatility the MSEs of the ML models with walk-forward validation indicate which models were able to better forecast the period. Furthermore, Table 6.2 is discussed per financial asset, in terms of the 'Asset specific criteria' in Section 4.4 in this section.

S&P 500 index

Figure5.2(a) shows the close price percentage change for the S&P 500 index. Periods of high volatility, between -10% and 10%, are observed for 2009 and 2020. Periods with medium volatility between -5% and 5% are observed for 2000-2004, 2011-2012, 2016 and 2019. Finally, periods with volatility between -2% and 2% are observed for 2004-2008, 2013-2015 and 2017.

The MSE performance over time for each ML model is shown in Figure 6.3. Each sub-figure shows a high MSE for the periods of high volatility (between -10% and 10%) identified in Figure5.2(a). Similarly, a higher MSE is observed for the period with volatility between -5% and 5%, compared to the periods with a low volatility.

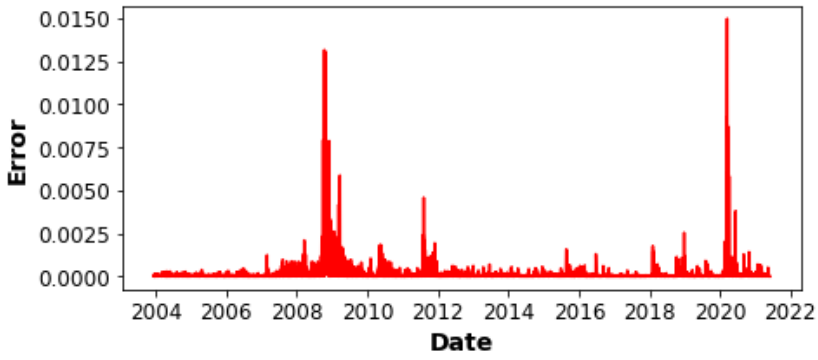
Figures 6.3(b), (d), (f) and (g) show that the MSEs for ARIMA, MLP, LSTM and GRU, when applied to the S&P 500 index, are similar. In other words, the models modelled the index in more or less the same way, because the size of the MSEs are similar for each period. Similarly, linear regression and SVR are considered to model the data in the same manner, because the MSE sizes for each period in Figures 6.3(a) and (c) are similar. Furthermore, linear regression and SVR models have higher MSEs compared to ARIMA, MLP, LSTM and GRU for 2008, 2009, 2012, 2018 and 2021.

Figure 6.3(e) shows that RNN had a similar MSE performance over time as the other models, except for 2020. In 2020 RNN had an MSE of 0.035, while the other models had an average MSE of 0.014. The higher MSE can be explained with the sporadic increase in volatility in 2020 in Figure5.2(a) - the RNN struggled to forecast this period more than the other models.

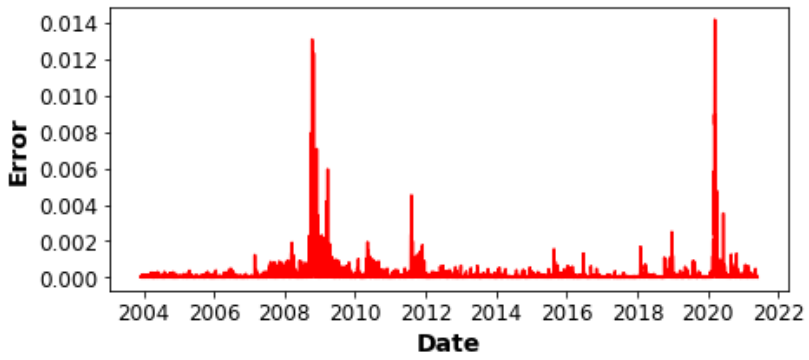
In terms of the average MSE, MLP and LSTM had the lowest MSEs. ARIMA, which had a similar MSE over time to MLP, LSTM and GRU, had a higher average MSE. With reference to Figure 6.3(b), it is found that the MSEs for ARIMA was larger by approximately 0.001 for every error. The GRU had the highest average accuracy of 53.07%, while MLP was to only model with an accuracy below 50%.

TABLE 6.2: *Performance evaluation of ML models employing walk-forward validation.*

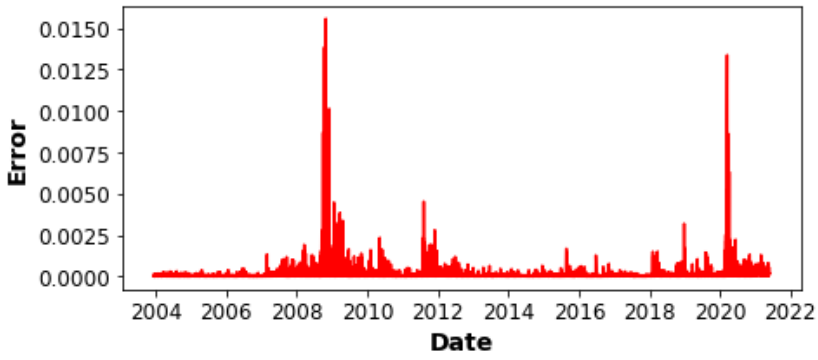
Performance evaluation of ML models employing walk-forward validation								
Asset	**Model/ Metric**	**LR**	**ARIMA**	**SVR**	**MLP**	**RNN**	**LSTM**	**GRU**
S&P 500 index	MSE	0.0001530	**0.0001972**	0.0001480	0.0001470	0.0001640	0.0001470	0.0001480
	Accuracy (%)	52.42	**51.56**	52.94	49.00	51.45	52.56	**53.07**
Bond yield	MSE	0.0006590	0.0008440	0.0006700	0.0006710	0.0008470	0.0006730	0.0006730
	Accuracy (%)	49.22	48.77	48.93	51.26	49.88	50.56	50.37
USD/ZAR	MSE	**0.0001290**	0.0002495	**0.0001270**	**0.0001270**	**0.0001590**	**0.0001270**	**0.0001270**
	Accuracy (%)	50.23	49.01	50.17	**52.09**	50.72	49.92	50.10
Gold	MSE	0.0002060	0.0003370	0.0002160	0.0002170	0.0002290	0.0002170	0.0002160
	Accuracy (%)	51.47	49.97	51.22	47.63	50.48	52.14	51.62
Bitcoin	MSE	0.0020080	0.0029862	0.0020310	0.0020060	0.0022920	0.0020020	**0.0022420**
	Accuracy (%)	**52.43**	50.67	**53.15**	48.47	**51.82**	**53.26**	51.44



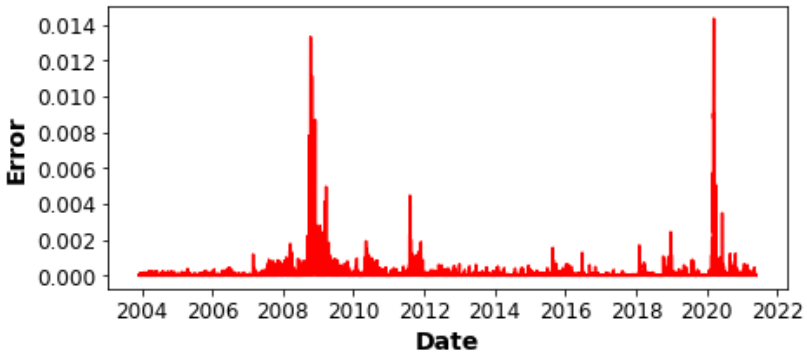
(a) *MSE over time for linear regression*



(b) *MSE over time for ARIMA model*

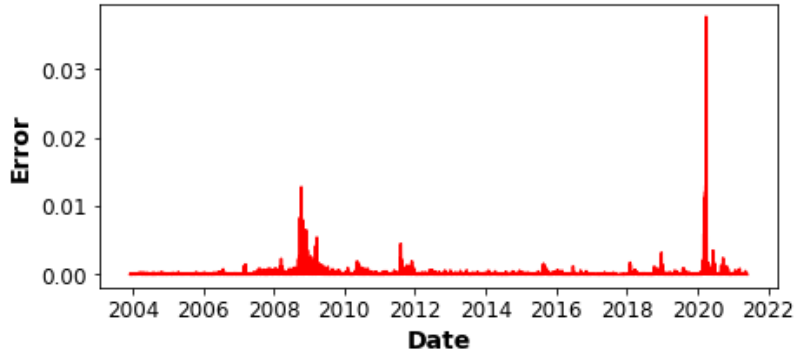


(c) *MSE over time for SVR model*

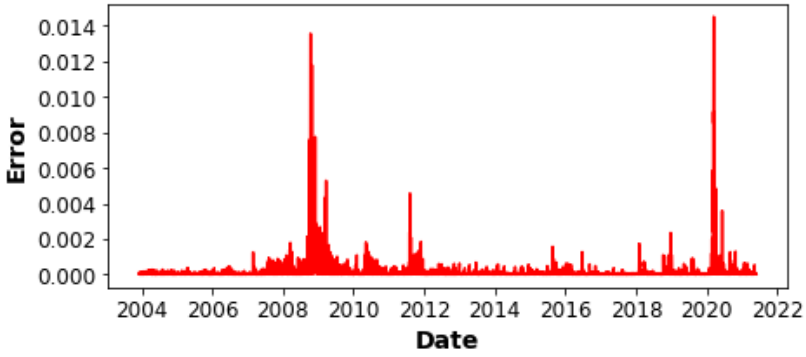


(d) *MSE over time for MLP model*

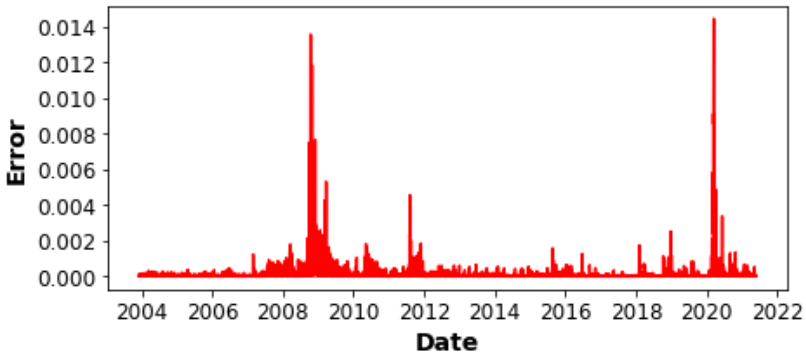
FIGURE 6.3: *The MSE over time for each ML model applied to the S&P 500 index data set walk-forward validation.* (cont.)



(e) *MSE over time for RNN model*



(f) *MSE over time for LSTM model*



(g) *MSE over time for GRU model*

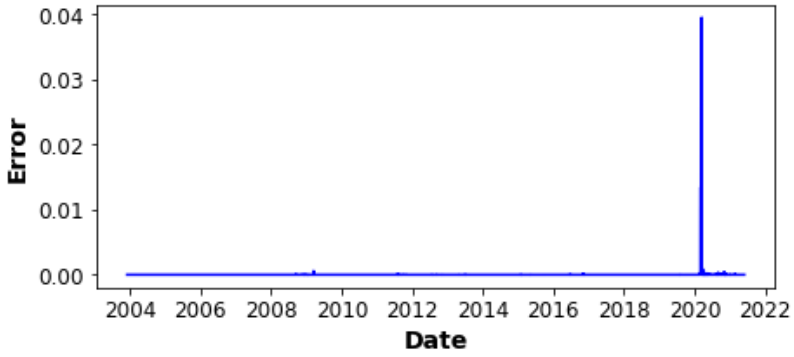
FIGURE 6.3: *The MSE over time for each ML model applied to the S&P 500 index data set walk-forward validation.*

The baseline model did not outperform the other models in terms of MSE and accuracy. The baseline model, however, still had an accuracy above 50% and the third lowest MSE. The baseline model is therefore not disregarded in favour of the other models. Two of the four neural networks (MLP and GRU) outperformed the other models in terms of average MSE, but were unable to outperform in terms of average accuracy. GRU and SVR had the same MSE, but GRU had a higher average accuracy. When considering both metrics, GRU is considered to be a more favourable model than SVR.

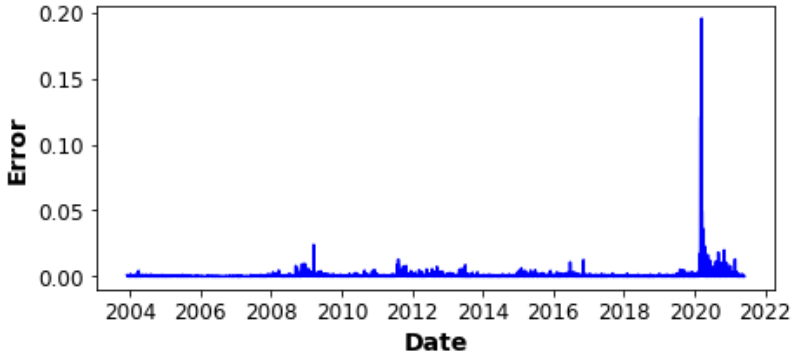
When considering the walk-forward MSE, average MSE, and average accuracy metrics for each model, the following observations and insights are gained: In periods of high or sporadic volatility, all models had a higher MSE in comparison to the other periods. RNN struggled more than the other models to forecast TS similar to the S&P 500 index with a sporadic volatility change. As a result, RNN is considered unfavourable for this type of TS. Linear regression is considered a well-suited model, because it has an accuracy above 50%, similar MSEs over time to the other models, and a simpler structure.

Bond yield

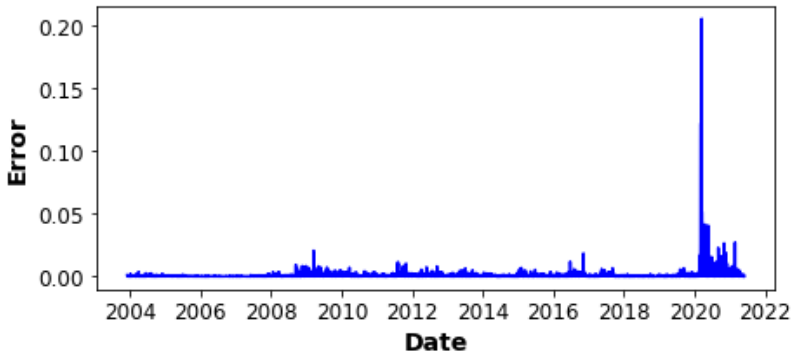
In Figure5.2(b), a period of high volatility, between -30% and 40%, is observed for the year 2020. Periods with medium volatility between -10% and 10% are observed for 2008-2010 and 2012-2017. The MSEs over time for each model in Figure 6.4 corresponds to the outlined periods of volatility. Furthermore, the MSEs over time for the remaining period for all models are significantly smaller, indicating that the models could make better forecast for these periods.



(a) *MSE over time for linear regression*

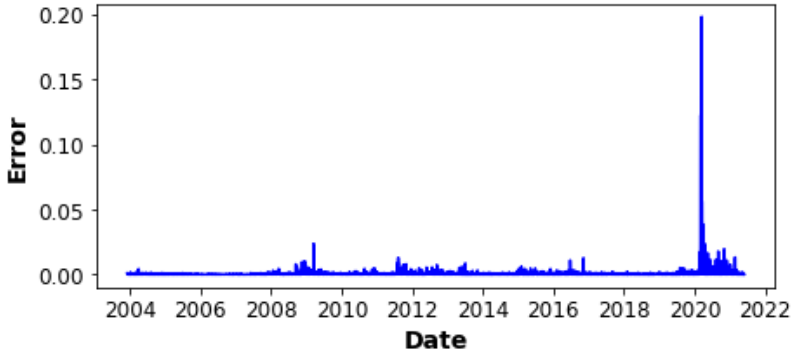


(b) *MSE over time for ARIMA model*

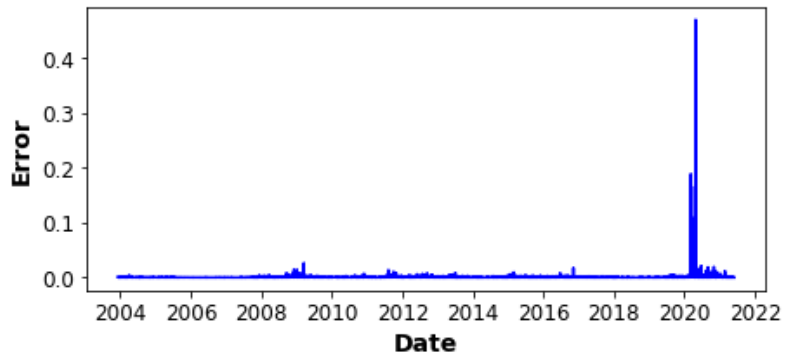


(c) *MSE over time for SVR model*

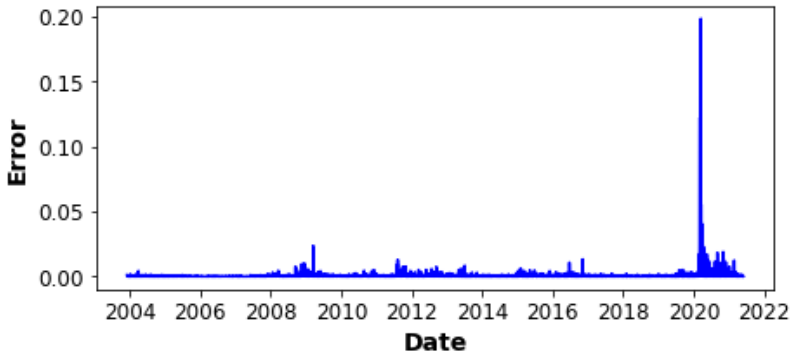
FIGURE 6.4: *The MSE over time for each ML model applied to the bond yield data set walk-forward validation.* (cont.)



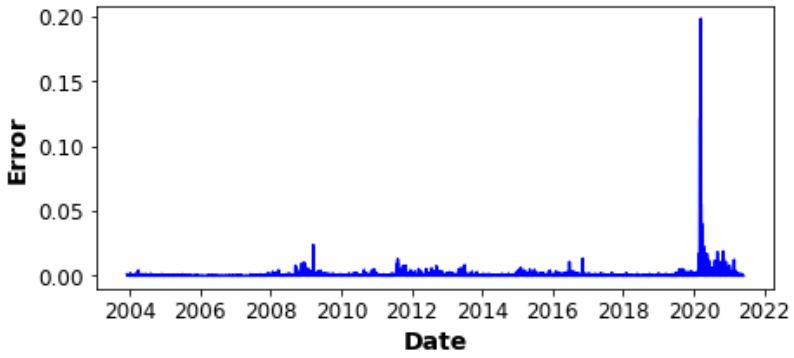
(d) *MSE over time for MLP model*



(e) *MSE over time for RNN model*



(f) *MSE over time for LSTM model*



(g) *MSE over time for GRU model*

FIGURE 6.4: *The MSE over time for each ML model applied to the bond yield data set walk-forward validation.*

The MSEs for ARIMA, SVR, MLP, LSTM and GRU in Figures 6.4(a), (b), (c), (d), (f) and (g) are similar to one another. Consequently, the models model the bond yield in the same manner. The MSE for the RNN model in Figure 6.3(e) is similar to the other models, except for 2020. The RNN had a MSE of 0.5, while the other models had an average MSE of 0.2. In Figure5.2(b), a sporadic increase in volatility is shown for this period. The higher MSE for RNN, therefore, shows that the model is unfavourable for TS, like the bond yield with a sporadic increase in volatility.

Linear regression and GRU had the lowest MSE, and RNN had the highest MSE when considering average MSEs. In terms of the average accuracy, MLP had the highest accuracy of 51.26%. MLP is one of three models that had an accuracy above 50%. The other two models are LSTM and GRU. The ARIMA model had the lowest accuracy of 48.77%.

The following observations and insights are gained when considering all the performance metrics: The MSEs observed over time for each ML model are similar to one another. As a result, the best-suited model when applied to the bond yield cannot be identified. However, RNN is the worst-suited model, because it had a higher MSE for the 2020 period. When considering the average MSE and accuracy, linear regression and SVR are not considered favourable models for the bond yield TS. Even though they have simpler structures, both had an average accuracy below 50%. The MLP, LSTM and GRU models are considered well-suited models for TS similar to the bond yield, because the models had an accuracy above 50% and the MSEs are not very different from the lowest MSE.

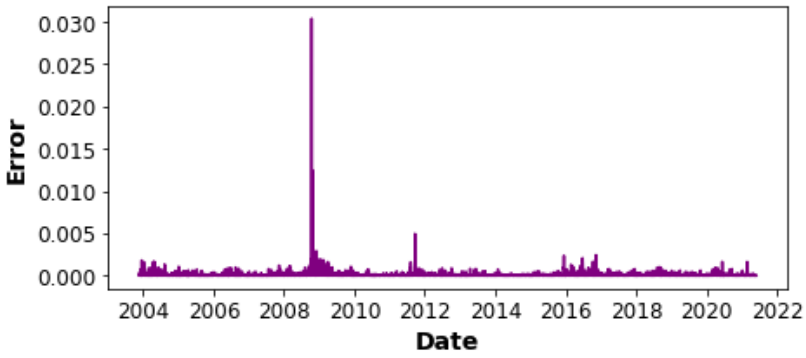
USD/ZAR currency pair

Figure 5.2(c) shows the close price percentage change for the USD/ZAR currency pair. A period of high volatility, between -10% and 15%, is observed for the year 2008. Furthermore, periods with medium volatility between -5% and 8% are observed for 2012 and 2016-2017. The MSE performance over time for each ML model, in Figure 6.5 shows a high MSE for 2008 and a MSE for the periods with volatility between -5% and 8%.

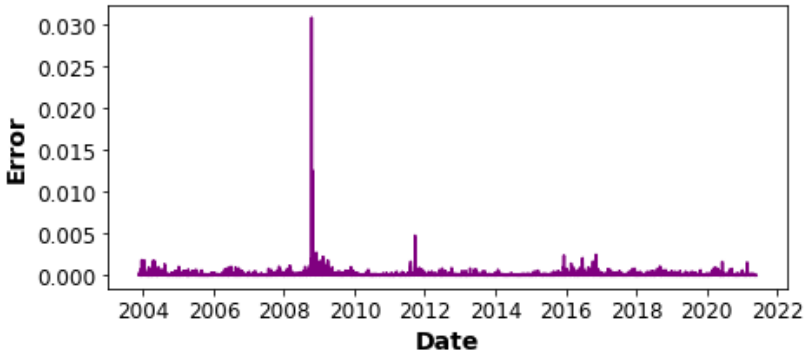
Figures 6.5(a), (b), (d), (f) and (g) shows that the MSEs for linear regression, ARIMA, MLP, LSTM and GRU, when applied to the currency pair, are similar. In other words, the ML models modelled the data in a similar manner because similar MSEs were observed. The MSE results of RNN in Figure 6.3(e) differs from the MSE results of the linear regression, ARIMA, MLP, LSTM and GRU in 2008. In 2008, RNN had a MSE of 0.035, while the other models had an average MSE of 0.03. The MSEs, in Figure 6.3(c), for 2008-2010 and 2012-2013 are higher for SVR than for the other models.

In terms of the average MSE, SVR, MLP, LSTM and GRU had the lowest MSEs. However, linear regression had a very similar MSE and is therefore considered to model the data in a similar manner as SVR, MLP, LSTM and GRU. Furthermore, the MLP model had the highest average accuracy of 52.09%. MLP is one of five models that had an accuracy above 50%. The other four models, ordered from highest to lowest accuracy, are RNN, linear regression, SVR and GRU.

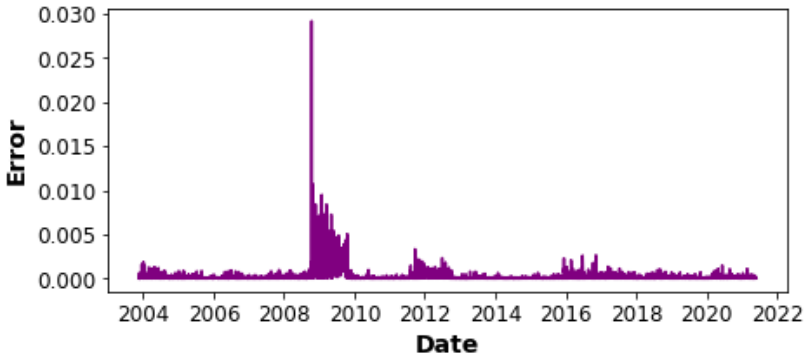
When considering the walk-forward MSE, average MSE and average accuracy metrics for each model, the following observations and insights are gained: The MSEs for each ML model in Figure 6.5 and the average MSE is similar for all the models, except SVR and RNN. Both SVR and RNN had higher MSEs and average MSEs. The average accuracy is, therefore the metric that is used to identify which models are favourable for the currency TS. Linear regression is one such model — it has the simplest structure and an accuracy above 50%.



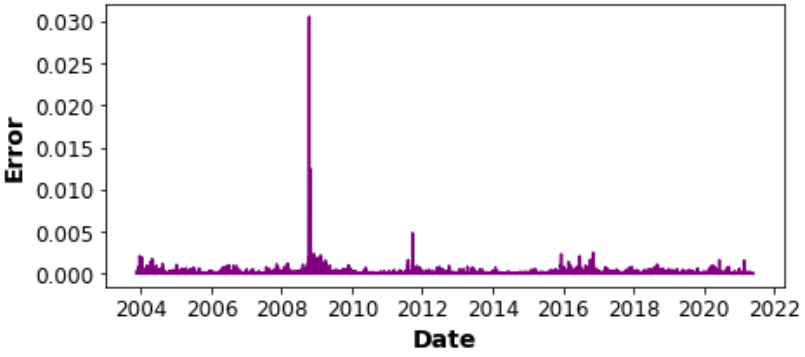
(a) *MSE over time for linear regression*



(b) *MSE over time for ARIMA model*

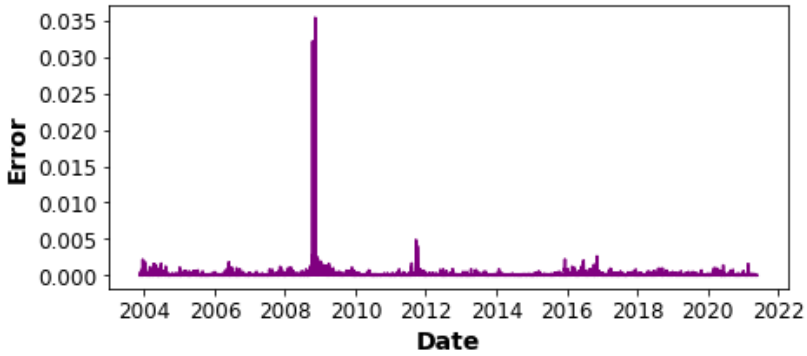


(c) *MSE over time for SVR model*

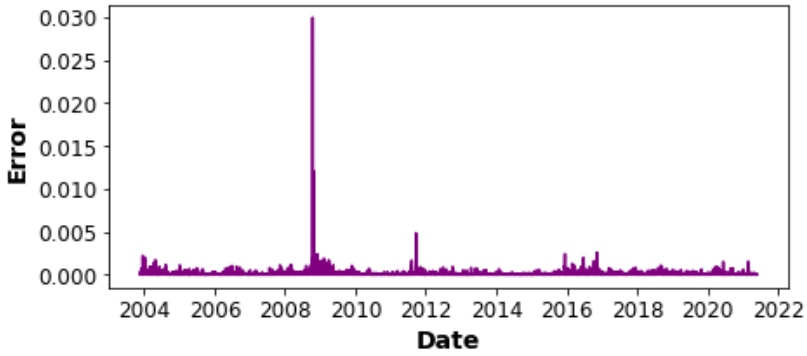


(d) *MSE over time for MLP model*

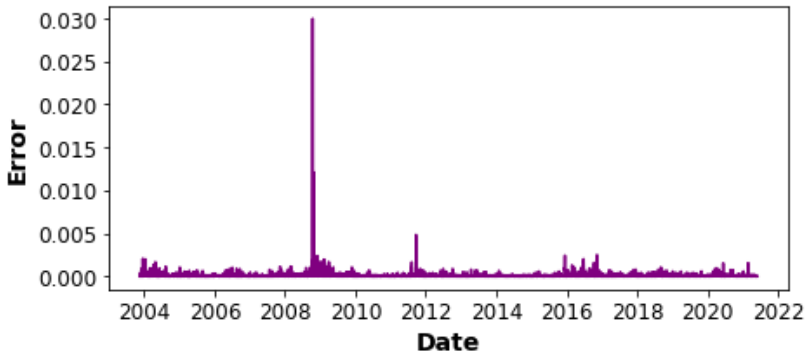
FIGURE 6.5: *The MSE over time for each ML model applied to the USD/ZAR currency pair data set walk-forward validation.* (cont.)



(e) *MSE over time for RNN model*



(f) *MSE over time for LSTM model*



(g) *MSE over time for GRU model*

FIGURE 6.5: *The MSE over time for each ML model applied to the USD/ZAR currency pair data set walk-forward validation.*

Gold

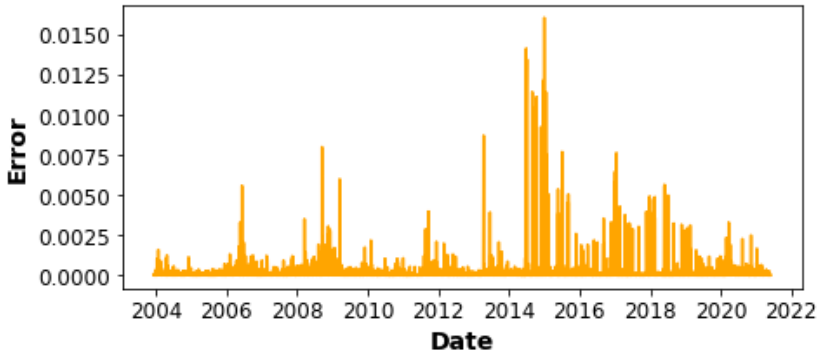
In Figure 5.2(d), a period of high volatility, between -10% and 10%, is observed for 2015. Periods of medium volatility, between -6% and 6%, are observed for 2000-2001, 2006, 2008-2010 and 2016-2020. Furthermore, volatility between -10% and 2% is observed for the years 2012 and 2013 and periods of small volatility, between -3% and 3%, are observed for the remaining periods. The MSEs over time for each model in Figure 6.6 corresponds to the outlined periods of volatility.

The MSEs for linear regression, ARIMA, MLP, LSTM and GRU in Figures 6.6(a), (b), (d), (f) and (g) are similar to one another. The MSE for the RNN model in Figure 6.6(e) is similar to the other models, except for 2009. The RNN had a MSE of 0.0175, while the other models had an average MSE of 0.008.

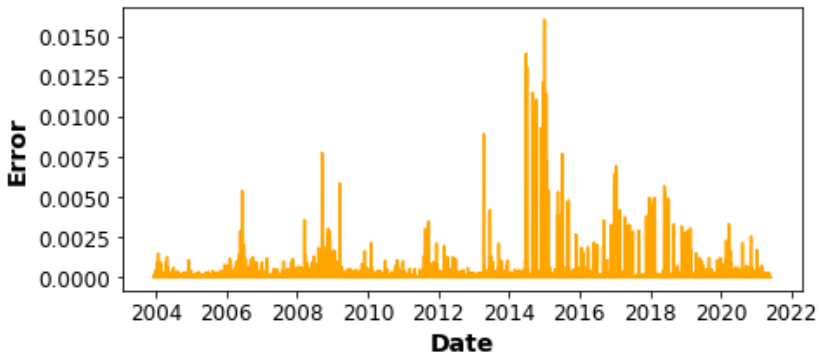
Furthermore, a higher MSE for SVR in 2015 was observed in Figure 6.6(c), in comparison to the MSEs of the other models for this period. The MSE performance over time for the SVR model in Figure 6.6(c) is very similar to the other ML models, except for the year 2015. The SVR had a MSE of 0.019, while the other models had an average MSE of 0.016.

Linear regression had the lowest MSE when considering the average MSEs. In terms of the average accuracy, the LSTM model had the highest accuracy of 52.14%. Only two models had an accuracy below 50%, namely ARIMA and MLP. The MLP model had the lowest accuracy of 47.63%.

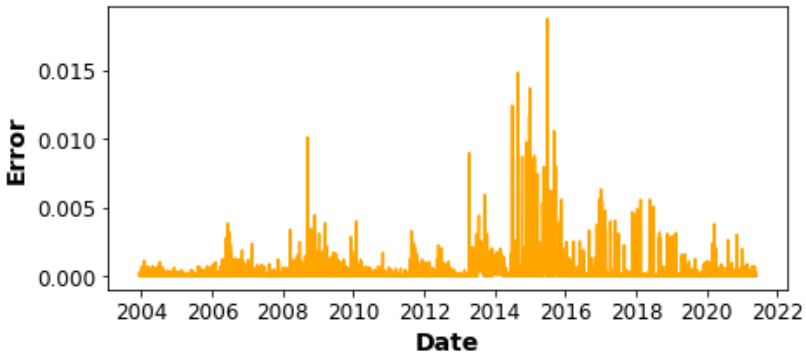
The following observations and insights are gained when considering all the performance metrics: The RNN model had a higher MSE compared to the other ML models in 2009. However, the model had the fourth-lowest MSE and an average accuracy above 50%. RNN is therefore still considered a well-suited model for the USD/ZAR currency pair. Linear regression is also considered as a favourable model, even more than RNN, because linear regression had the lowest average MSE and an accuracy above 50%.



(a) *MSE over time for linear regression*

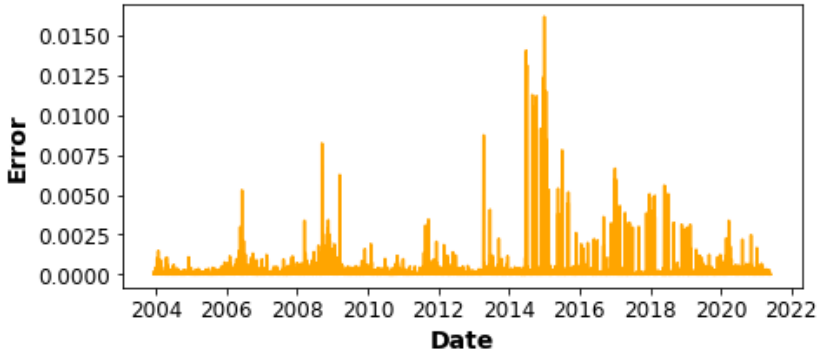


(b) *MSE over time for ARIMA model*

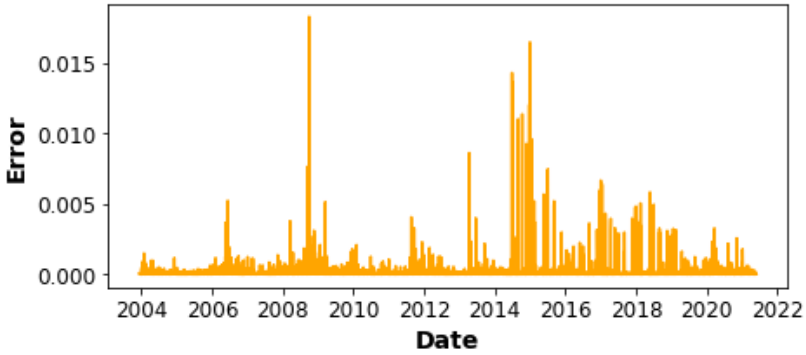


(c) *MSE over time for SVR model*

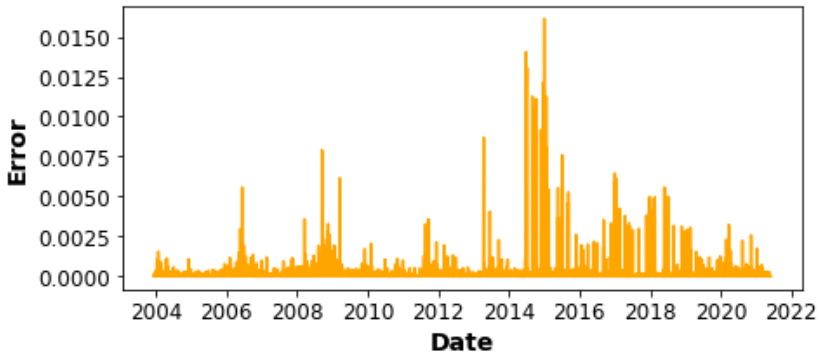
FIGURE 6.6: *The MSE over time for each ML model applied to the gold time series data set walk-forward validation.* (cont.)



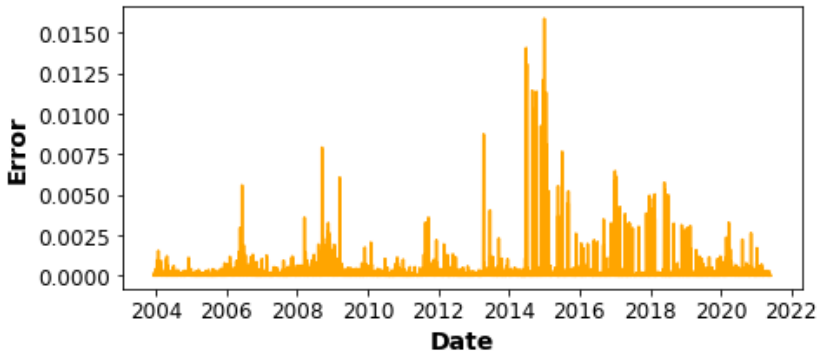
(d) *MSE over time for MLP model*



(e) *MSE over time for RNN model*



(f) *MSE over time for LSTM model*



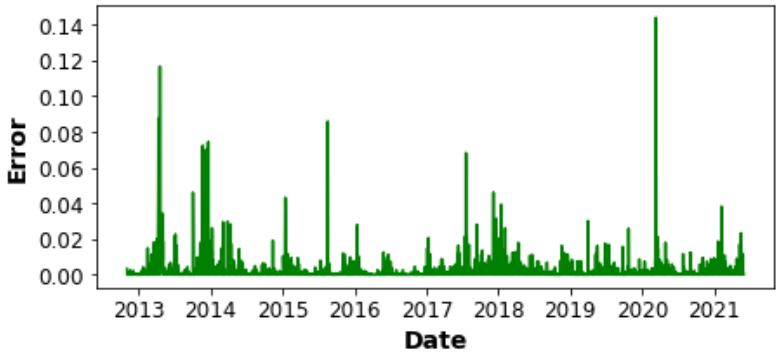
(g) *MSE over time for GRU model*

FIGURE 6.6: *The MSE over time for each ML model applied to the gold time series data set walk-forward validation.*

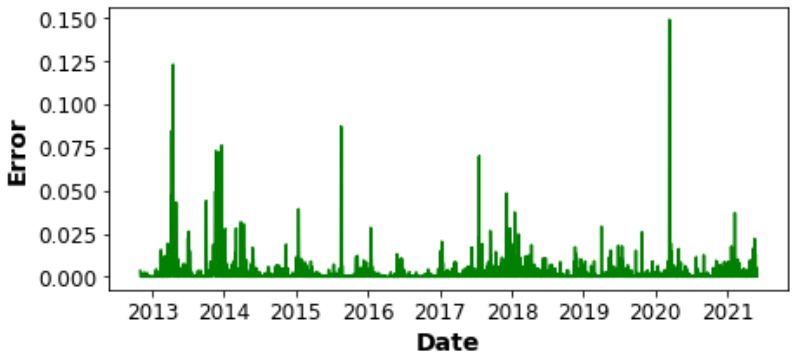
Bitcoin

Figure 5.2(e) shows the close price percentage change for the Bitcoin TS. For the entire period, sporadic increases and decreases in volatility are observed. Periods with volatility between -40% and 20% are observed for 2013, 2014, 2015 and 2020. Periods with volatility between -20% and 20% are observed for the years 2015 and 2018. The remaining periods experience volatility between -10% and 10% on average.

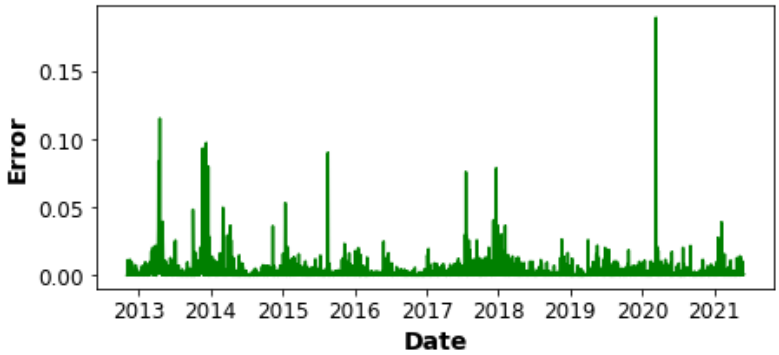
The MSE performance over time for each ML model is shown in Figure 6.7. Each sub-figure shows a high MSE for the periods of high volatility (between -40% and 20%) identified in Figure5.2(e). Similarly, a higher MSE is observed for the period with volatility between -20% and 20%.



(a) *MSE over time for linear regression*

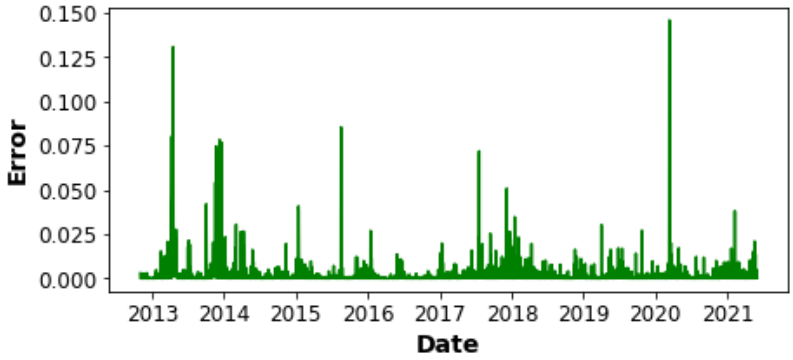


(b) *MSE over time for ARIMA model*

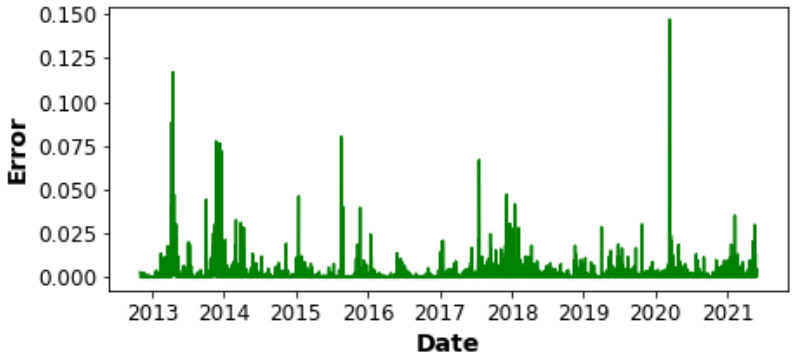


(c) *MSE over time for SVR model*

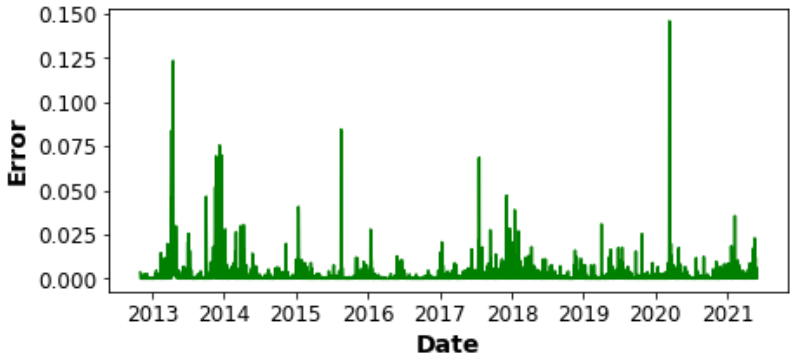
FIGURE 6.7: *The MSE over time for each ML model applied to the Bitcoin time series data set walk-forward validation.* (cont.)



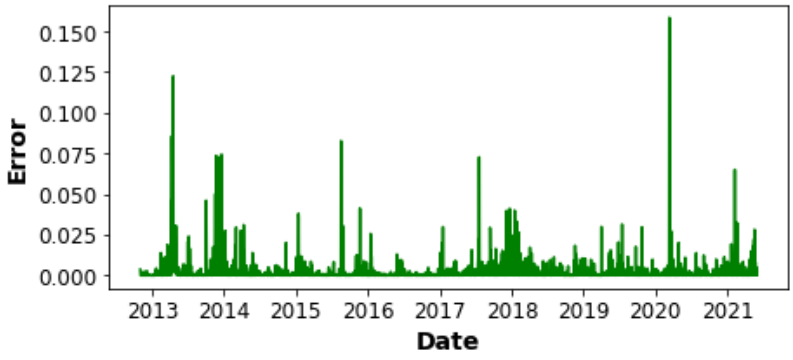
(d) *MSE over time for MLP model*



(e) *MSE over time for RNN model*



(f) *MSE over time for LSTM model*



(g) *MSE over time for GRU model*

FIGURE 6.7: *The MSE over time for each ML model applied to the Bitcoin time series data set walk-forward validation.*

Figures 6.5(a), (b) and (f) show that the MSEs for linear regression, ARIMA and LSTM, when applied to the Bitcoin TS, are similar. ML models, therefore, model the data in a similar manner, because the size of MSEs is similar for each period. Figure 6.5(e) shows that SVR had a similar MSE performance over time as the other models, except for 2018 and 2020. In 2018 the SVR model had a MSE of 0.075, while the other ML models had an average MSE of 0.04. In 2020 the SVR model had a MSE of 0.175, while the other ML models had an average MSE of 0.14.

The MSE performance over time for the MLP model in Figure 6.5(d) shows that the MLP model had a larger MSE in 2013, in comparison with the other ML models. In 2013, the MLP had a MSE of 0.13, while the other models had an average MSE of 0.12. The difference between MSEs is not large, and therefore SVR is considered to model the data in a similar manner as the other models. Figure 6.5(e) shows that the GRU had a higher MSE in 2016 and between 2018-2021. In 2016 the GRU model had a MSE of 0.04 while the other ML models had an average MSE of 0.03. For the year 2018, it was observed that GRU had a larger number of observations with a higher MSE compared to the same period for the other models. Between 2019 and 2020, a similar phenomenon is observed as in 2018. In 2020 the GRU model had a MSE of 0.15 while the other ML models had an average MSE of 0.14. In 2021 the GRU model had a MSE of 0.06 while the other ML models had an average MSE of 0.03. Based on the observed MSEs for GRU, the GRU is not considered as a favourable model for the Bitcoin TS data set.

In terms of the average MSE, LSTM had the lowest average MSE. The average MSE difference between linear regression and SVR, MLP, LSTM and GRU is small. In addition, linear regression has a simpler structure. Linear regression is therefore considered a more well-suited model for the Bitcoin TS than the other models. LSTM had the highest average accuracy of 53.26%, and MLP had the lowest accuracy of 48.47%. The MLP was the only model with an accuracy below 50% when applied to the Bitcoin TS.

When considering the walk-forward MSE, average MSE and average accuracy metrics for each model, the following observations and insights are gained: The MSEs for each ML model in Figure 6.7 are very different from one another for certain periods. However, the average MSEs are very similar to one another. As a result, it is concluded that an ensemble of the different ML models would be useful when applied to the Bitcoin TS.

6.2.2 General asset criteria analysis

When applied to the USD/ZAR currency pair, all of the ML models had the lowest MSE compared to the other financial assets. The remaining assets, ordered by MSE performance from lowest to highest, are the S&P 500 index, the gold TS, the bond yield and the Bitcoin TS.

Linear regression

The average accuracy metric for linear regression in Table 6.2 shows that the ML model had an accuracy above 50% for all the financial assets, except for the bond yield. When applied to the bond yield, linear regression had an accuracy of 49.22%. Furthermore, linear regression had the highest accuracy of 52.43% for the Bitcoin TS, even though it had the highest MSE. The remaining assets ordered by accuracy performance from highest to lowest are the S&P 500 index, gold TS and the currency pair.

The following observations and insights are gained for linear regression: When applied to the S&P 500 index, the gold TS and the currency pair, linear regression is considered a favourable model. Linear regression had low MSEs and an accuracy above 50% for the three TS data sets. Linear regression is the not a well-suited model for the bond yield, because it had the highest MSE and an accuracy below 50%.

Autoregressive integrated moving average

ARIMA had an accuracy below 50% for the gold TS, USD/ZAR currency pair and the bond yield. When applied to the bond yield, ARIMA had the lowest accuracy of 48.77%. Furthermore, ARIMA had an accuracy of 49.01% and 49.97% for the USD/ZAR currency pair and gold TS, respectively. On the other hand, ARIMA had an accuracy above 50% when applied to the S&P 500 index and the Bitcoin TS.

ARIMA is considered a favourable model for TS similar to the S&P 500 index, because it had the lowest MSE and the highest accuracy above 50%. Furthermore, ARIMA is not considered a favourable model for bond yield, USD/ZAR currency pair and gold TS, since it had accuracies below 50% and the highest MSEs.

Support vector regression

The average accuracy metric for SVR shows that the ML model had an accuracy above 50% for all the financial assets, except for the bond yield. SVR had the highest accuracy of 53.15% for the Bitcoin TS, even though it had the highest MSE. The remaining assets ordered by accuracy performance from highest to lowest are the S&P 500 index, gold TS and the USD/ZAR currency pair.

The following observations and insights are gained for SVR: SVR is well-suited for the S&P 500 index. When applied to the S&P 500 index, SVR had the second-lowest MSE and an accuracy above 50%. There is a large difference in the MSE and accuracy rankings for SVR when applied to the Bitcoin TS and USD/ZAR currency pair. For example, SVR had the highest accuracy and the highest MSE for the Bitcoin TS and the second-lowest accuracy and lowest MSE for the currency pair TS. Furthermore, SVR is not considered as a favourable model for the bond yield, because it had the second-highest MSE and an accuracy below 50%

Multilayer perceptron

MLP is a favourable model for TS similar to the USD/ZAR currency pair. When applied to the currency pair, MLP had the lowest MSE compared to when it was applied to the other financial assets. Furthermore, MLP had an accuracy of 52.09% when applied to the currency pair. SVR was to only model, thus far, with an accuracy above 50% for the bond yield data set, even though it had the second-highest MSE. As a result, SVR is still considered as a favourable model for the bond yield TS. However, SVR is not considered a well-suited model for the Bitcoin TS, because it had the highest MSE and an accuracy below 50%.

Recurrent neural network

RNN only had an accuracy below 50% for the bond yield TS. When applied to the bond yield, RNN had an accuracy of 49.88%. On the other hand, RNN had the highest accuracy of 51.82% for the Bitcoin TS. Furthermore, the accuracy difference for RNN when applied to the Bitcoin TS, the S&P 500 index, the USD/ZAR currency pair and gold TS is very small. The accuracy ranges between 50.48% and 51.82% for the RNN when applied to these TS.

RNN is considered a favourable model for the USD/ZAR currency pair and S&P 500 index. RNN had the lowest MSE and second-lowest MSE for the respective TS and an accuracy above 50%. RNN is not considered as a well-suited model for the bond yield, since it had an accuracy below 50% and the second-highest MSE.

Long short term memory

The accuracy metric for LSTM in Table 6.2 shows that the ML model had an accuracy above 50% for all the financial assets, except for the USD/ZAR currency pair. The LSTM had the highest accuracy of 53.26% for the Bitcoin TS. LSTM and MLP were the only models thus far with an accuracy above 50% for the bond yield data set. When applied to the bond yield, LSTM had an accuracy of 50.56%.

The following insights are gained for the LSTM model: LSTM is well-suited for the S&P 500 index and gold TS data sets, because it had a low MSE and high accuracy above 50%, in comparison to the other financial assets. LSTM is not considered as a favourable model for the Bitcoin TS, because it had the lowest MSE.

Gated recurrent unit network

GRU had an accuracy above 50% for all the financial assets. GRU had the highest accuracy of 53.07% for the S&P 500 index and the lowest accuracy of 50.10% for the USD/ZAR currency pair. The GRU had an accuracy of 50.37% for the bond yield. In general, GRU has a low MSE, and an accuracy above 50% regardless of the TS utilised. As a result, GRU is considered a favourable model when applied to any of the financial assets.

6.3 Validation techniques comparison

This section compares single out-of-sample validation with walk-forward validation for each ML model when applied to different financial asset data sets. Separate summaries for the 'Asset specific criteria' and 'General asset criteria' specifications are provided. However, the comparison is executed by considering both summaries. The concept of a parsimonious model is not considered in the comparison, because the effect of different validation techniques on all the ML models is compared.

A summary of the 'Asset specific criteria' is provided in Table 6.3. For each asset in Table 6.3 the ML models with the lowest MSE and highest accuracy are sorted by validation technique. In Table 6.3 the following codes are employed: 'WF' refers to walk-forward validation and 'SOOS' refers to single out-of-sample validation. A ML model shown with an asterisk (*) indicates that the model had an accuracy above 50% for the validation technique it is listed under, while the model did not have an accuracy above 50% when the other validation technique was employed. A ML model shown with a hash character, for example #LR, indicates that the model had a *higher* accuracy above 50% for the validation technique it is listed under. The model, however still had an accuracy above 50% for the other validation technique. A ML model that is shaded in grey indicates that the model had the highest accuracy when both validation techniques were considered. Finally, a ML model that is emphasised, for example **LSTM**, indicates that the model had the highest accuracy in comparison to the other ML models for the validation technique it is listed under. The term "All ML models" in the MSE column refers to all the ML models utilised, namely linear regression, ARIMA, SVR, MLP, RNN, LSTM and GRU.

A summary of the 'General asset criteria', documented in Table 6.4, shows the assets that result in the lowest MSE and highest accuracy by validation technique for each ML model. The codes employed in Table 6.4 are the same as for Table 6.3.

In terms of MSE metric in Tables 6.3 and 6.4, all ML models employing walk-forward validation had a lower MSE when applied to the S&P 500 index and bond yield TS. On the other hand, all ML models employing single out-of-sample validation had a lower MSE when applied to the USD/ZAR currency pair and gold TS. The ARIMA model with walk-forward validation had a lower MSE for the Bitcoin TS. The remaining ML models with single out-of-sample validation, therefore, had a lower MSE for the Bitcoin TS.

Linear regression, MLP, LSTM and GRU had a higher accuracy for the S&P 500 index with single out-of-sample validation than walk-forward validation. MLP with single out-of-sample validation had the highest overall accuracy of 57.92%. However, the MSE is double the size of the MSE for MLP with walk-forward validation. A similar phenomenon is observed for the other ML models and the size of their MSEs. As a result, the models with walk-forward validation are considered — SVR and GRU had the same MSEs. SVR had an accuracy of 52.94%, and GRU had an accuracy of 53.07%. SVR is however considered a more favourable model than the GRU, because it has a simpler structure.

Only MLP, LSTM and GRU with walk-forward validation had an accuracy above 50% for the bond yield. MLP had the highest accuracy of 51.26% and is considered the most suited model for the bond yield TS.

TABLE 6.3: *Summary of model performance by validation technique and financial asset.*

Summary of model performance by validation technique and financial asset				
Asset	**MSE**		**Accuracy**	
	WF	**SOOS**	**WF**	**SOOS**
S&P 500 index	All ML models		LR MLP LSTM ARIMA SVR* RNN* **GRU**	#LR #LSTM #GRU ARIMA #MLP
Bond yield	All ML models		LSTM* GRU* MLP*	
USD/ZAR		All ML models	LR SVR* RNN* GRU #MLP	#LR ARIMA* **MLP**
Gold		All ML models	LR #SVR RNN GRU* **#LSTM**	SVR #RNN LSTM #LR ARIMA*
Bitcoin	ARIMA	All ML models, except ARIMA	LR ARIMA* SVR RNN* GRU* **LSTM**	#LR #SVR MLP*

MLP with walk-forward validation had the highest accuracy of 52.09% for the USD/ZAR currency pair. However, the MSE is 1.3 times larger than the MSE for MLP with single out-of-sample validation. The same phenomenon is observed for the other ML models with walk-forward validation. Consequently, the models with single out-of-sample validation are considered — MLP had the highest accuracy of 51.77%, in this case. The MLP with single out-of-sample validation is therefore considered as a favourable model for the USD/ZAR currency pair.

Linear regression and ARIMA with single out-of-sample validation had the highest overall accuracies of 53.51% for the gold TS. Both models have a lower MSE than linear regression and ARIMA with walk-forward validation. As a result, linear regression and ARIMA with single out-of-sample validation are considered well-suited models for the gold TS. Linear regression is however preferred, because it has a simpler structure and a lower computational cost.

MLP with single out-of-sample validation had the highest overall accuracy of 55.04% for the Bitcoin TS. The MSE was also smaller than the MSE of the MLP with walk-forward validation. Consequently, MLP with single out-of-sample validation is considered as a favourable model for the Bitcoin TS.

TABLE 6.4: *Summary of comparison by validation technique and model.*

Summary of comparison by validation technique and model				
ML model	**MSE**		**Accuracy**	
	WF	**SOOS**	**WF**	**SOOS**
Linear regression	S&P 500 index Bond yield	USD/ZAR Gold Bitcoin	Bond yield	S&P 500 index USD/ZAR Gold Bitcoin
ARIMA	S&P 500 index Bond yield Bitcoin	USD/ZAR Gold	Bond yield Bitcoin* S&P 500 index	USD/ZAR* Gold* S&P 500 index
SVR	S&P 500 index Bond yield	USD/ZAR Gold Bitcoin	Bond yield USD/ZAR S&P 500 index*	Bitcoin* Gold
MLP	S&P 500 index Bond yield	USD/ZAR Gold Bitcoin	Bond yield* USD/ZAR Gold	S&P 500 index* Bitcoin*
RNN	S&P 500 index Bond yield	USD/ZAR Gold Bitcoin	Bond yield USD/ZAR* Bitcoin* S&P 500 index	Gold
LSTM	S&P 500 index Bond yield	USD/ZAR Gold Bitcoin	Bond yield* USD/ZAR Bitcoin* Gold	S&P 500 index
GRU	S&P 500 index Bond yield	USD/ZAR Gold Bitcoin	Bond yield* USD/ZAR* Bitcoin* Gold*	S&P 500 index

6.4 Results summary

In this section, the key findings in terms of the performance of each ML model when applied to the financial assets based on the validation technique are documented. First, the *individual* findings of the single out-of-sample and walk-forward validation techniques in Section 6.1 and 6.2 are documented. The key findings when *comparing* the single out-of-sample and walk-forward validation techniques in Section 6.3 follows in the subsequent section.

6.4.1 Individual findings based on validation technique

For the single out-of-sample validation technique, linear regression had the best performance on the S&P 500 index, USD/ZAR currency pair, gold TS and Bitcoin TS data sets. Linear regression had the lowest MSEs in general, an accuracy above 50%. The data sets comprised only closing percentage data over three days, which are not considered as data with complex relationships and patterns. Considering the simple structure of a linear regression model, the type of data utilised, and the low MSE and above 50% accuracy, linear regression is deemed the best model. In addition, linear regression has the least assumptions and is therefore considered as the parsimonious model.

The walk-forward validation technique provides additional insight into the forecasts of a model for different periods. For example, the MSE over time shows the period in time that the ML model struggled to forecast. RNN had a higher MSE for periods with a sporadic and high volatility, regardless of the TS data set used. It was concluded that RNN requires more training observations with sporadic and high volatilities to have a similar average MSE to other ML models. With reference to parsimonious models, linear regression with walk-forward validation is considered as a favourable model.

Linear regression is generally the best-suited model for both validation techniques when approximate MSEs and accuracies are considered. The simplicity of the data set contributed to this finding, because linear regression performed as well or better than the more complex models. The architecture of the neural networks was too complex for the input data. As a result, the neural networks did not perform the best or better than the simpler models.

6.4.2 Findings of comparison between validation techniques

All the ML models with walk-forward validation had a lower MSE for the S&P 500 index and bond yield data set. On the other hand, all ML models with single out-of-sample validation had a lower MSE for the currency pair, gold TS and Bitcoin TS. ARIMA is however an exception for the Bitcoin TS. ARIMA with walk-forward validation had a lower MSE. SVR and MLP with walk-forward validation had the highest accuracies for the S&P 500 index and bond yield TS, respectively. SVR had an accuracy of 52.94% for the S&P 500 index, and MLP had an accuracy of 51.26% for the bond yield.

Furthermore, MLP with single out-of-sample validation had the highest accuracies for the currency pair and Bitcoin TS. MLP had an accuracy of 51.77% for the currency pair and 55.04% for the Bitcoin TS. Linear regression and ARIMA with single out-of-sample validation had the highest accuracy of 53.51% for the gold TS.

Although ML models with single out-of-sample validation had a lower MSE and higher accuracy for three of the five data sets, ML models with walk-forward validation are still preferred. Walk-forward validation is a more 'honest' way of testing a ML model, because the technique allows for a clearer understanding of how the ML model performs in different periods. As mentioned previously, single out-of-sample validation is biased towards the most recent period of data and only utilises a small amount of the available data to generate an out-of-sample validation.

6.5 Chapter summary

The purpose of this chapter was to implement Modules 7.0 to 9.0 from the analysis component based on the FTSEP described in Figure 4.1. The chapter opened with a summary of the performance metric results for the ML models with single out-of-sample validation. The results were interpreted according to the Criteria data store. A summary of the performance metrics for the ML models with walk-forward validation was provided in a tabular format. The results were analysed and discussed, as in the previous section. Finally, a summary of the key findings of comparing ML models with single out-of-sample validation and walk-forward validation was provided.

For the individual analysis of the single out-of-sample and walk-forward validation technique, linear regression was identified as the best ML model for all FTS. Linear regression performed as well or better than the other ML models in terms of MSE and accuracy. The simplicity of the data set contributed to linear regression being favoured. Furthermore, ML models with walk-forward validation had a lower MSE when forecasting the S&P 500 index and the bond yield compared to their counterparts with single out-of-sample validation. SVR is considered a favourable model for the S&P 500 index, because it has

the highest accuracy of 52.94%. MLP had the highest accuracy of 51.26% for the bond yield and is therefore considered the best-suited model for this TS.

ML models with single out-of-sample validation had lower MSEs and higher accuracies for the USD/ZAR currency pair, gold and Bitcoin TS than their counterparts with walk-forward validation. MLP had the highest accuracies of 51.77% and 55.04% for the USD/ZAR currency pair and Bitcoin TS, respectively. Linear regression had the highest accuracy of 53.51% for the gold TS.

CHAPTER 7

Conclusion

Contents

The purpose of this chapter is threefold: First, to present a summary of the work contained in this dissertation, secondly to offer an appraisal of the contributions to data science, and thirdly to provide suggestions for future follow-up work.

7.1 Project summary

The continuous pursuit for higher yields in financial markets motivated researchers and practitioners to investigate ML models from other domains on *financial time series* (FTS) data sets, the general predictive capabilities of ML models and the types of validation techniques that can be used. This type of investigation can ultimately help one increase the potential yield of an investment and beat the market.

This dissertation investigated the predictive capabilities of widely accepted ML models for FTS forecasting when different types of FTS data sets are employed in pursuit of identifying ML models that can increase the return of an investment in the financial market. The ML models investigated include linear regression, ARIMA, SVR, MLP, RNN, LSTM and GRU. Different FTS data sets were utilised to avoid selection bias when testing the ML models. The FTS data sets employed were the S&P 500 index, US 10 year bond yield, the USD/ZAR currency pair, gold futures and Bitcoin TS data sets. The effect of single out-of-sample and walk-forward validation on the ML models was also studied. The MSE and accuracy performance metrics were used to compare and contrast different validation techniques and FTS data sets on the ML models. The accuracy metric formed part of the comparison process, because error metrics by themselves (MSE) do not provide an absolute evaluation in FTS forecasting. The accuracy was defined as the ratio between the number of correct guesses about whether the percentage change was positive or negative for the next day and the total number of guesses. Accuracy was interpreted as follows: If the accuracy of a ML model was below 50%, the model was not considered a well-suited model. The performance of the model is worse than randomly guessing whether the percentage change will be positive or negative. On the other hand, if the performance is above 50%, the model has the potential to beat the market, because it performs better than random guessing. An accuracy above 50% was considered

substantial, because Renaissance Technologies have proved that with an accuracy of 51% they were able to have an average return of 40.18%. In addition, they were able to beat the market, which only had a return of 12.43%.

The goal of this dissertation was accomplished with the following steps: A thorough survey of the literature pertaining to financial markets, time series, ML models, and validation and verification techniques were documented. An empirical process was designed to facilitate the preprocessing and transformation of data and the development, evaluation and comparison of ML models. FTS data was then obtained and prepared in accordance with the requirements of the empirical procedure. Subsequently, the performances of the ML models were evaluated with MSE and accuracy. Finally, the performances of each ML model based on financial asset and validation technique were compared and contrasted. To avoid overfitting and underfitting, the early stopping regularisation technique was implemented with the neural networks.

It was found that all ML models with walk-forward validation had a lower MSE for the S&P 500 index and bond yield data sets compared to their counterparts with single out-of-sample validation. SVR is considered a favourable model for the S&P 500 index, because it has the highest accuracy of 52.94%. MLP had the highest accuracy of 51.26% for the bond yield and is therefore considered the best-suited model for this TS. ML models with single out-of-sample validation had lower MSEs and higher accuracies for the USD/ZAR currency pair, gold and Bitcoin TS than their counterparts with walk-forward validation. MLP had the highest accuracies of 51.77% and 55.04% for the USD/ZAR currency pair and Bitcoin TS, respectively. Linear regression had the highest accuracy of 53.51% for the gold TS.

Regarding the validation technique selected, ML models with walk-forward validation are preferred and considered as more favourable for FTS forecasting. Walk-forward validation allowed for a clearer understanding of how a ML model performed in different periods. One could quickly gauge whether a ML model struggled more than the other models for a certain period by comparing MSEs. For example, RNN had a higher MSE than the other ML models for all TS that had a sporadic increase in volatility.

In general, linear regression with single out-of-sample validation or walk-forward validation performed as well or better than the other ML models in terms of MSE and accuracy. The simplicity of the data set contributed to this observation, because the other models (especially neural networks) are more suited for data with complex patterns and relationships.

7.2 Appraisal of the contributions to data science

The focus of this dissertation was twofold: The first aim was to compare and contrast the predictive capabilities of widely accepted ML models (statistical, ML and ANN models) for FTS forecasting when different types of FTS data sets are employed. The second aim was to compare and contrast the predictive capabilities of the ML models when the single out-of-sample validation technique and the walk-forward validation technique were employed.

This dissertation successfully compared the performance and technical capabilities of selected ML models when applied to different FTS data sets. The effect of TS data with low, high, sporadic or constant volatility on the performance of the ML model was thoroughly analysed and documented. Not only were the effects of different FTS observed, but also the effect of the validation technique selected. In addition, the FTSEP followed to obtain the results and insights documented in this dissertation is available at `https://github.com/Judene/fts_predictors`.

The insights gained in this dissertation may be of considerable assistance to individuals from both academia and industry. Both types of individuals can utilise the insights gained in this dissertation to further their investigation into FTS forecasting regarding the type of ML model employed, financial asset predicted and the validation technique selected.

7.3 Suggestions for future work

In fulfilment of Objective VII in Section 1.4, various suggestions for possible future work involving improvements of, or extensions to, the work documented in this dissertation are discussed. The suggestions are partitioned into two sections. The first section addresses future work concerning the types of ML models utilised, and the second section addresses the utilisation of different data types. In each section, the suggestions are stated formally and then elaborated upon and motivated.

7.3.1 Suggestions related to the machine learning models employed

Suggestion I *Apply other machine learning models from the natural language processing domain.*

The field of ML and AI is constantly growing, adapting and improving. For example, there are new developments and successes in forecasting and prediction in the NLP domain. More specifically, attention networks and transformer networks have been found to be successful. Considering that TS and documents have similarities, it would be insightful to evaluate if these newer architectures from the NLP domain could also be applied to FTS.

The attention mechanism, introduced by Bahdanau *et al.* [7], is an evolution of the encoder-decoder model. In the context of NLP, the attention mechanism enables the model (decoder) to look at previous words and focus on specific words required for the current prediction. If applied to FTS data, the attention mechanism will look at previous financial data in general and focus on specific past data observations that will aid in predicting the current period correctly.

The paper "Attention Is All You Need" authored by Vaswani *et al.* [121] introduces a novel encoder-decoder architecture based on attention layers, termed the "transformer network". A transformer network differs from the neural networks investigated in this dissertation, because it does not imply any recurrent networks. A transformer eschews recurrence and instead relies entirely on an attention mechanism to draw global dependencies between input and output. The idea of not employing RNNs was presented by Vaswani *et al.* because they proved that an architecture with only attention mechanisms improves the performance of the network in terms of technical capability (MSE).

Suggestion II *Include a broader range of hyperparameters to analyse different neural network architectures.*

This dissertation employed a simplistic approach with regard to the architecture of neural networks. The same hyperparameters were defined for each neural network, as shown in Table 5.3. The reason for utilising the same hyperparameters was to obtain a baseline performance for each model, which could be compared. However, the hyperparameters selected for a neural network play a crucial role in the performance of the neural network [4]. In other words, the selection of a specific value for a hyperparameter has a potentially positive or negative effect on the performance of the network. For example, the number of hidden layers, hidden units, number of epochs, type of regularisation and batch size influences the bias and variance of a model, which in turn introduces problems of overfitting or underfitting [123]. Optimisation of the hyperparameters for each neural network may provide interesting insights into the *true* capabilities of each model to the different financial assets.

Suggestion III *Use different data set sizes for the walk-forward validation technique.*

For the walk-forward validation technique, this dissertation implemented a fixed testing data set size of 10 days, a fixed validation data set size of 20 days, and a fixed training data set size of 216 days. The data set size has a significant influence on the performance of the model in

terms of bias and variance [17]. If the set period k is kept constant to 251 days and the training, validation and testing data sets are adjusted, the optimal combination for the different data sets can be determined. In turn, the optimal combination provides the highest performance possible for the model and provides more *accurate* insights into the capabilities of the models under investigation.

7.3.2 Suggestions related to the utilisation of data

Suggestion IV *Incorporate more detailed demand and supply data.*

The *price* data in terms of the principle of supply and demand for FTS includes the open, close, high and low prices per day for a financial asset. In this dissertation, however, only the close price was employed. Incorporation of the open, low and high price data per day can provide additional insights and possibly improve the predictive capabilities of a ML model. The *quantity* data in terms of the principle of supply and demand includes the volume of shares traded per day. Incorporating volume data may also provide additional insights.

Suggestion V *Incorporate data typically employed in fundamental analysis.*

Fundamental analysis examines and combines the underlying forces that affect the well-being of an economy, industry sector and company [65]. Economic forces include the overall strength of the economy, interest rate, GDP and inflation rate, to name but a few. Forces that influence the industry sector include the industry life cycle, conditions, overall trends and competitive advantages. Forces that directly influence the company include profits, profit margins, losses, expenses, liabilities, earnings per share, and cash balances.

Data related to the forces are employed in fundamental analysis and used to determine the company's intrinsic value. By utilising this type of data in the forecasts, the future price movements of an asset can be predicted more accurately. The data is typically of a lower granularity. For example, economic and industry sector data are available monthly, and company data is available quarterly or biannually. However, after applying data transformations to the data, it can be used in conjunction with the supply and demand data which have a higher granularity.

Suggestion VI *Incorporate data from other financial assets that correlate with the financial asset under investigation.*

The gold price and the crude oil price are both dollar-denominated, meaning that they are strongly linked to one another [105]. When considering the gold price, crude oil price and inflation the following is observed: When the crude oil price rises, inflation also rises. Gold is a good hedge against inflation, and therefore the value of gold will rise. It has been found that more than 60% of the time there is a direct relationship between the gold and crude oil price.

Other financial assets that have a positive or negative relationship with the financial asset under investigation may provide additional insight and possibly increase the predictive performance of a ML model. Ideally, other financial assets correlated with the asset under investigation should be identified and utilised as input data for the ML model.

References

[1] ABARBANELL J & BUSHEE B, 1997, *Fundamental analysis, future earnings, and stock prices*, Journal of accounting research, **1(35)**, pp. 1–24.

[2] ABDULKARIM SA & ENGELBRECHT AP, 2019, *Time Series Forecasting using Neural Networks: Are Recurrent connections necessary*, Neural Processing Letters, **3**, pp. 2673–2795.

[3] ADAMS J, 2014, *8 Key Factors that Affect Foreign Exchange Rates*, [Online], [Cited April 2020], Available from `https://www.compareremit.com/money-transfer-guide/key-factors-affecting-currency-exchange-rates/`.

[4] AGGRAWAL C, 2018, *Neural networks and deep learning*, 1st Edition, Springer, Switzerland.

[5] ANTONOPOULOS AM, 2014, *Mastering Bitcoin: unlocking digital cryptocurrencies*, O'Reilly Media, Inc.

[6] BACHTIS D, AARTS G & LUCINI B, 2021, *Adding machine learning within Hamiltonians: Renormalization group transformations, symmetry breaking and restoration*, Physical Review Research, **3(1)**.

[7] BAHDANAU D, CHO K & BENGIO Y, 2015, *Neural machine translation by jointly learning to align and translate*, Proceedings of the 3rd International Conference on Learning Representations, SanDiego, United States, pp. 209–216.

[8] BAUER S, 2019, *3 Factors That May Impact the S&P 500 in Q2*, [Online], [Cited July 2021], Available from `https://www.thestreet.com/markets/3-factors-that-may-impact-the-s-p-500-in-q2-14913558`.

[9] BEERS B, 2015, *Which Economic Factors Impact Treasury Yields*, [Online], [Cited May 2021], Available from `https://www.investopedia.com/ask/answers/062315/which-economic-factors-impact-treasury-yields.asp`.

[10] BENGOI Y, SIMARD P & FRASCONI P, 1994, *Learning long-term dependencies with gradient descent is difficult*, IEEE Transactions on Neural Networks, **5(2)**, pp. 157–166.

[11] BERGSTRA J & BENGOI Y, 2012, *Random search for hyper-parameter optimization*, Journal of machine learning research, **13(2)**.

[12] BODIE Z, KANE A & MARCUSL A, 2013, *Essentials of Investments: Global Edition*, McGraw Hill.

[13] BOEHMKE B & GREENWELL B, 2019, *Hands-on machine learning with R*, Chapman and Hall/CRC.

[14] BOEING, 2020, *Boeing time series data*, [Online], [Cited October 2021], Available from `https://www.investing.com/equities/boeing-co`.

[15] BONACCORSOL G, 2020, *Mastering Machine Learning Algorithms*, 2nd Edition, Packt Publishing, Upper Saddle River (NJ).

[16] BOX GE, JENKINS GM, REINSEL GC & LJUNG GM, 2008, *Time series analysis: forecasting and control John Wiley & Sons*, Hoboken, NJ.

[17] BRAIN D & WEBB GI, 1999, *On the effect of data set size on bias and variance in classification learning*, Proceedings of the 4th Australian Knowledge Acquisition Workshop, University of New South Wales.

[18] BRONWLEE J, 2018, *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*, 1st Edition, Machine Learning Mastery, Cambridge (MA).

[19] BROWNLEEL J, 2018, *Deep Learning for Time Series Forecasting*, 1st Edition, Machine learning mastery, Upper Saddle River (NJ).

[20] BURNISKE C & TATAR J, 2018, *Crypto-Assets: Das Investoren-Handbuch für Bitcoin, Krypto-Token und Krypto-Commodities*, Vahlen.

[21] BURNS S, 2020, *Current Renaissance Technologies Returns*, [Online], [Cited October 2021], Available from `https://www.newtraderu.com/2021/01/31/current-renaissance-technologies-returns/`.

[22] CERQUEIRA V, TORGO L & MOZETIC I, 2020, *Evaluating time series forecasting models: An empirical study on performance estimation methods*, Machine Learning, **109(1)**, pp. 1997–2028.

[23] CHEUNG Y & LAI KS, 1995, *Lag order and critical values of the augmented Dickey–Fuller test*, Journal of Business & Economic Statistics, **13(3)**, pp. 277–280.

[24] CHO K, MERRI B, GULCEHRE C & BAHDANAU D, 2014, *Empirical Methods in Natural Language Processing*, 1st Edition, Association for Computational Linguistics, Doha, Qatar.

[25] CHO K, MERRIËNBOER BV, GULCEHRE C, BAHDANAU D, BOUGARES F, SCHWENK H & BENGIO Y, 2014, *Learning phrase representations using RNN encoder-decoder for statistical machine translation*, arXiv preprint arXiv:1406.1078.

[26] CHRISTOPER N, WELLER P & DITTMAR R, 1997, *Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach.*, The Journal of Financial and Quantitative Analysis, **4(32)**, pp. 405–426.

[27] CHUNG J, GULCEHRE C, CHO K & BENGIO Y, 2014, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, arXiv preprint arXiv:1412.3555.

[28] CLAESEN M & DE MOOR B, 2015, *Hyperparameter search in machine learning*, Proceedings of the 11th Metaheuristics International Conference, Agadir, pp. 1–5.

[29] CRISTELLI M, 2013, *Complexity in financial markets: modeling psychological behavior in agent-based models and order book models*, Springer Science & Business Media.

[30] DANGI J, 2019, *Commodity Trading: Top 10 Most Traded Commodities In The World*, [Online], [Cited April 2021], Available from `https://www.starthubpost.com/most-traded-commodities-worldwide/`.

[31] DEY R & SALEM F, 2017, *Gate-variants of gated recurrent unit (GRU) neural networks*, Proceedings of the 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS), pp. 1597–1600.

[32] DORGAN G, 2016, *What Drives Government Bond Yields*, [Online], [Cited July 2021], Available from `https://snbchf.com/markets/bonds/government-bond-yields/`.

[33] DUNIS CL & WILLIAMS M, 2002, *Modeling and trading the UER/USDexchange rate: Do Neural Network models perform better?," in Derivatives Use, Trading and Regulation*, **8(3)**, pp. 211–239.

[34] EKMEKCIOGLU E, 2013, *Role of Financial Markets in a Global Economy and the Concept of Uncertainty*, International Journal of Academic Research in Economics and Management Sciences, **2(4)**, pp. 199–206.

[35] ERDOGDU A, 2017, *The Most Significant Factors Influencing the Price of Gold: An Empirical Analysis of the US Market*, Economics World, **5(5)**, pp. 399–406.

[36] FAMA E, 1960, *Efficient market hypothesis*, PhD Thesis, University of Chicago, Chicago.

[37] FAUSETT L, 1994, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, 1st Edition, Prentice Hall, Cambridge (MA).

[38] FRISBY D, 2014, *Bitcoin: the future of money?*, Unbound Publishing.

[39] GERLEIN E, MCGINNITY M, BELATRECHE A & COLEMAN S, 2016, *Evaluating machine learning classification for financial trading: An empirical approach*, Expert Systems with Applications, **54**, pp. 193–207.

[40] GLOROT X & BENGIO Y, 2010, *Understanding the difficulty of training deep feedforward neural networks*, Proceedings of the 13th international conference on artificial intelligence and statistics, pp. 249–256.

[41] GODOY D, 2020, *Hyper-parameters in action! part ii-weight initializers*.

[42] GOODFELLOW I, BENGIO Y & COURVILLE A, 2016, *Deep Learning*, 3rd Edition, MIT Press, Cambridge (MA).

[43] GRAUPE D, 2013, *Principles of artificial neural networks*, World Scientific.

[44] GRAY C, 2018, *Stock Prediction with ML: Walk-forward Modeling*, [Online], [Cited October 2021], Available from `https://alphascientist.com/walk-forward-model-building.html`.

[45] HABIB M, ALJARAH I, FARIS H & MIRJALILI S, 2020, *Multi-objective particle swarm optimization: theory, literature review, and application in feature selection for medical diagnosis*, Evolutionary Machine Learning Techniques, **241**, pp. 175–201.

[46] HAMILTONI J, 1994, *Time series analysis*, Princeton University Press, Princeton, NJ.

[47] HE K, ZHANG X, REN S & SUN J, 2015, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, Proceedings of the IEEE international conference on computer vision, pp. 1026–1034.

[48] HINTON J & SEJNOWSKI T, 1999, *Unsupervised learning: Foundations of neural computation*, (Unpublished) Technical Report ISBN 9780262581684, MIT Press, New York.

[49] HOCHREITER S & SCHMIDHUBER J, 1997, *Long short-term memory*, Neural computation, **9(8)**, pp. 1735–1780.

[50] HOSSIN M & SULAIMAN MN, 2015, *A review on evaluation metrics for data classification evaluations*, International journal of data mining & knowledge management process, **5(2)**, pp. 1.

[51] HULL JC, 2019, *Options, futures and other derivatives*, Pearson.

[52] HUTCHINSON JM, 1993, *A radial basis function approach to financial time series analysis*.

[53] HYNDMAN RJ & KOEHLER AB, 2006, *Another look at measures of forecast accuracy*, International journal of forecasting, **22(4)**, pp. 679–688.

[54] JAKKULA V, 2006, *Tutorial on support vector machine (svm)*, School of EECS, Washington State University, **37**.

[55] JAMES G, WITTEN D, HASTIE T & TIBSHIRANI R, 2013, *An introduction to statistical learning*, Springer, New York (NY).

[56] KAELBLING L & LITTMAN M, 1996, *Reinforcement learning: A survey*, Artificial Intelligence Research, **4**, pp. 237–285.

[57] KAZMAIER J, 2020, *A framework for evaluating unstructured text data using sentiment analysis*, PhD Thesis, Stellenbosch University, Stellenbosch.

[58] KELLEHER J, NAMEE B & D'ARCY A, 2015, *Fundamentals of machine learning for predictive data analytics*, The MIT Press, Cambridge (MA).

[59] KENDALL K & KENDALL J, 2013, *Systems analysis and design*, 9th Edition, Pearson, Upper Saddle River (NJ).

[60] KENNY T, 2017, *Factors That Affect U.S. Treasury Yields*, [Online], [Cited March 2021], Available from https://www.thebalance.com/historical-u-s-treasury-yield-charts-417126.

[61] KHONDKAR K, 2020, *Berkshire Hathaway(Brk.A/ Brk.B) Yearly Return*, [Online], [Cited October 2021], Available from https://tradingninvestment.com/berkshire-hathawaybrk-brk-b-yearly-return/.

[62] KINGMA D & BA J, 2015, *Adam: A method for stochastic optimization*, Proceedings of the 3rd International Conference on Learning Representations, San Diego (CA), pp. 1419–1432.

[63] KIRCHGASSNER G, WOLTERS J & HASSLER U, 2013, *Introduction to Modern Time Series Analysis*, 2nd Edition, Springer, Germany.

[64] KOHZADI N, BOYD M, KERMANSHAHI B & KAASTRA I, 1996, *A comparison of artificial neural network and time series models for forecasting commodity prices*, Neurocomputing, **10(2)**, pp. 169–181.

[65] KRANTZ M, 2016, *Fundamental analysis for dummies*, John Wiley & Sons.

[66] L M, PHAM H & MANNING CD, 2015, *Effective approaches to attention-based neural machine translation*, arXiv preprint arXiv:1508.04025.

[67] LAURA M, 2021, *What is Cryptocurrency: Your Complete Crypto ABC*, [Online], [Cited July 2021], Available from https://www.bitdegree.org/crypto/tutorials/what-is-cryptocurrencyso-what-is-cryptocurrency-mining-for.

[68] LAWRENCE S & LEE GC, 2000, *Overfitting and neural networks: conjugate gradient and backpropagation*, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, pp. 114–119.

[69] LAWTON G, 2020, *8 types of bias in data analysis and how to avoid them*, [Online], [Cited August 2021], Available from https://searchbusinessanalytics.techtarget.com/feature/8-types-of-bias-in-data-analysis-and-how-to-avoid-them.

[70] LEAL-TAXE L & NIESSER M, 2018, *Introduction to deep learning*, Lecture Notes, Department of Informatics at the Technical University of Munich, Munich.

[71] LEWIS N, 2016, *Deep Time Series Forecasting with Python: An Intuitive Introduction to Deep Learning for Applied Time Series Modeling*, 3rd Edition, CreateSpace Independent Publishing Platform, Cambridge (MA).

[72] LEWIS ND, 2015, *Applied Predictive Modeling Techniques in R: With step by step instructions on how to build them FAST*, North Charleston: CreateSpace Independent Publishing Platform.

[73] LI X, ZHANG W & DING Q, 2019, *Understanding and improving deep learning-based rolling bearing fault diagnosis with attention mechanism*, Signal processing, **161**, pp. 136–154.

[74] LOBEL B, 2019, *A Guide to the S&P 500 Index*, [Online], [Cited July 2021], Available from https://www.dailyfx.com/sp-500/what-is-sp-500.html.

[75] LOPEZ-IBANEZ M, DUBOIS-LACOSTE J, CACERES L & BIRATTARI M, 2016, *The irace package: Iterated racing for automatic algorithm configuration*, Operations research perspectives, **3**, pp. 43–58.

[76] MAITRA S, 2021, *Take Time-Series a Level-Up with Walk-Forward Validation*, [Online], [Cited October 2021], Available from https://sarit-maitra.medium.com/take-time-series-a-level-up-with-walk-forward-validation-217c33114f68.

[77] MAKRIDAKIS S & HIBON M, 1998, *ARMA Models and the Box–Jenkins Methodology*, International Journal of Forecasting, **16(3)**, pp. 147–163.

[78] MALKIEL B & BURTON G, 2019, *A random walk down Wall Street: the time-tested strategy for successful investing*, WW Norton & Company.

[79] MANNING C & SCHUTZE H, 1999, *Foundations of statistical natural language processing*, MIT press.

[80] MICROSOFT, 2020, *Microsoft time series data*, [Online], [Cited October 2021], Available from https://www.investing.com/equities/microsoft-corp.

[81] MIRJALILI S, FARRIS H & ALJARAH I, 2020, *Evolutionary machine learning techniques: Algorithms and applications*, Springer, Princeton.

[82] MITCHELL T, 1997, *Machine Learning*, McGraw-Hill Science/Engineering/Math, New York (NY).

[83] MLADJENOVIC P, 2018, *Precious Metals Investing for Dummies*, John Wiley & Sons.

[84] MUNOZ A, 2014, *Machine Learning and Optimization*, Courant Institute of Mathematical Sciences.

[85] MURPHYL J, 1999, *Study Guide to Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*, Penguin.

[86] MUSHTAQ R, 2014, *Augmented dickey fuller test*.

[87] NAMDARI A & DURRANI T, 2021, *A multilayer feedforward perceptron model in neural networks for predicting stockj market short-term trends*, Operations research forum, **2(1)**, pp. 2–38.

[88] NAMOUS F, FARIS H, HEIDARI A, KHALAFAT M, ALKHAWALDEH R & GHATASHEH N, 2020, *Evolutionary and swarm-based feature selection for imbalanced data classification*, Evolutionary Machine Learning Techniques, pp. 231–250.

[89] NEG A, 2020, *Introduction to deep learning*, Department of Computer Science at Stanford University, Stanford (CA).

[90] OLIVIER C & SCHOLKOPF B, 2006, *Semi-supervised learning*, (Unpublished) Technical Report ISBN 9780262033589, MIT Press, Cambridge (NY).

[91] PAGANO M, 1993, *Financial markets and growth: an overview*, European Economic Review, **37(3)**, pp. 613–622.

[92] PAOLELLA M, 2019, *Linear Models and Time-Series Analysis*, 3rd Edition, John Wiley & Sons, Switzerland.

[93] PARDO R, 2008, *The evaluation and optimization of trading strategies*, John Wiley & Sons.

[94] PEDRYCZ W & CHEN S, 2014, *Information granularity, big data, and computational intelligence*, Springer.

[95] PEDRYCZ W & CHENI S, 2014, *Information granularity, big data, and computational intelligence*, Springer, Princeton.

[96] PRITRAM A, 2020, *Top 10 Countries with the Highest Currency Value in the World: 2021 Edition*, [Online], [Cited June 2021], Available from `https://www.bookmyforex.com/blog/highest-currency-in-the-world/`.

[97] QIAN Y, RALESCU D & ZHANG B, 2019, *The analysis of factors affecting global gold price*, Resources Policy, **64**, pp. 679–688.

[98] R HYNDMAN GA, 2018, *Forecasting: Principles and practise*, OTexts, Monash University, Australia.

[99] RAGANA D, CHANDRASEKARA N & TILAKARATNE C, 2011, *Comparison of support vector regression and artificial neural network models to forecast daily Colombo Stock Exchange*, Proceedings of the International Statistics Conference, Institute of Applied Statistics, Sri Lanka.

[100] RUMELHART D, HINTON G & WILLIAMS R, 1986, *Learning representations by backpropagating errors*, Nature, **1**, pp. 533–536.

[101] RUSSEL S & NORVIG P, 2010, *Artificial intelligence: A modern approach*, (Unpublished) Technical Report ISBN 9780136042594, Prentice Hall, Paris.

[102] SAMIR M, 2016, *Machine learning theory — Part 2: Generalization bounds*, [Online], [Cited August 2021], Available from `https://mostafa-samir.github.io/ml-theory-pt2/`.

[103] SEZER OB, GUDELEK MU & OZBAYOGLU AM, 2020, *Financial time series forecasting with deep learning : A systematic literature review: 2005–2019*, Applied Soft Computing, **90(4)**, pp. 1249–1268.

[104] SHACHMUROVE Y & WITKOWSKA D, 2000, *Utilizing Artificial Neural Network model to predict stock markets*, CARESS Working Paper, **1**, pp. 1–11.

[105] SHAFIEE S & TOPAL E, 2010, *An overview of global gold market and gold price forecasting*, Resources policy, **35(3)**, pp. 178–189.

[106] SIMONIS J, 2019, *A generic framework for predicting the severity of roa accidents: A data mining approach*, BSc Thesis, Stellenbosch University, Stellenbosch.

[107] SMOLA A & SCHOLKOPF B, 2004, *A tutorial on support vector regression*, (Unpublished) Technical Report 14, Statistics and Computing, Netherlands.

[108] SMOLA A & VISHWANATHAN A, 2008, *Introduction to machine learning*, Cambridge University Press, College of Engineering and Computer Science Australian National University.

[109] SNOEK J, LAROCHELLE H & ADAMS RP, 2012, *Practical bayesian optimization of machine learning algorithms*, Advances in neural information processing systems, **25**.

[110] STUTZLE T, 1999, *Local search algorithms for combinatorial problems — Analysis, improvements, and new applications*, PhD Thesis, Technical University of Darmstadt,, Darmstadt.

[111] SURESH A, 2013, *A study of fundamental and technical techniques*, International Journal of Marketing, Financial Services & Management Research, **2(5)**, pp. 44–59.

[112] TASHI Q, RAIS H, ABDULKADIR S, MIRJALILI S & ALHUSSIAN H, 2020, *A review of grey wolf optimizer-based feature selection methods for classification*, Evolutionary Machine Learning Techniques, pp. 273–286.

[113] TASHMAN L, 2000, *Out-of-sample tests of forecasting accuracy: an analysis and review*, International Journal of Forecasting, **16(1)**, pp. 437–450.

[114] TATIBOUET M, 2021, *What Are The Key Factors Influencing The Price Of Bitcoin*, [Online], [Cited July 2021], Available from https://www.finance-monthly.com/2019/07/what-are-the-key-factors-influencing-the-price-of-bitcoin/.

[115] THAHER T, HEIDARI A, MAFARJA M, DONG JS & MIRJALILI S, 2020, *Binary Harris Hawks optimizer for high-dimensional, low sample size feature selection*, Evolutionary Machine Learning Techniques, pp. 251–272.

[116] TRUJILLO L, GONZÁLEZ E, ERNESTO A, GALVÁN E, TAPIA J & PONSICH A, 2020, *On the analysis of hyper-parameter space for a genetic programming system with iterated F-Race*, Soft Computing, **24(19)**, pp. 14757–14770.

[117] TSAY R, 2010, *Analysis of Financial Time Series*, 3rd Edition, John Wiley & Sons, Chicago (IL).

[118] UNIVERSITY S, 2018, *Convolutional neural networks for visual recognition: Image classification*, Department of Computer Science at Stanford University, Stanford (CA).

[119] VALDEZ S & MOLYNEUXI P, 2010, *An introduction to global financial markets*, Macmillan International Higher Education, Bangor University (UK).

[120] VAN HEERDEN SA, 2018, *A framework for conducting road accident risk evaluations using network spatial analysis and machine learning techniques*, MSc Thesis, Stellenbosch University, Stellenbosch.

[121] VASWININ A, SHAZEER N & PARMAR N, 2017, *Attention is all you need*, Advances in Neural Information Processing Systems, pp. 5998–6008.

[122] WANG W, 2021, *Bayesian optimisation concept explained in layman terms*, [Online], [Cited August 2021], Available from https://towardsdatascience.com/bayesian-optimization-concept-explained-in-layman-terms-1d2bcdeaf12f.

[123] WIJAYASEKARA D, MANIC M, SABHARWALL P & UTGIKAR V, 2011, *Optima artificial neural network architecture selection for performance prediction of compact heat exchanger with the EBaLM-OTRl*, Nuclear Engineering and Design, **241(7)**, pp. 2549–2557.

[124] WINSTON W & GOLDBERG J, 2004, *Operations Research: Applications and Algorithms*, 4th Edition, Brooks/Cole, Belmont (CA).

[125] WU W, AN S, GUAN P, HAUNG D & ZHOU B, 2019, *Time series analysis of human brucellosis in mainland China by using Elman and Jordan recurrent neural networks*, BMC infectious diseases, **19(1)**, pp. 1–11.

[126] XU Y, 2020, *Financial signal processing*, [Online], [Cited October 2021], Available from https://towardsdatascience.com/financial-signal-processing-part-1-69c20fd5ad6d.

[127] YAO J & TAN CL, 2000, *A case study on neural networks to perform technical forecasting of forex*, Neurocomputing, **34**, pp. 79–98.

[128] ZUCKERMAN G, 2019, *The Man Who Solved the Market: How Jim Simons Launched the Quant Revolution*, Portfolio/Penguin.

APPENDIX A

Cleaned financial asset time series data set extract

An extract of the various financial asset data sets, after the execution of Modules 1.0 to 4.1, is provided in Table A.1. For each asset the '*Date*', '*Close*' and '*Change %*' features are tabulated. The TS data are utilised in the study conducted in Chapter 5.

TABLE A.1: *Time series extract for each financial asset.*

Original time series extract for each financial asset										
Assets	**S&P 500 index**		**Bond yield**		**USD/ZAR**		**Gold**		**Bitcoin**	
Date	**Close**	**% Change**	**Close**	**% Change**	**Close**	**% Change**	**Close**	**% Change**	**Close**	**% Change**
2021-07-23	4411.79	1.015	1.281	0.313	14.843	1.090	1801.8	-0.199	33597.1	3.951
2021-07-24	4422.30	0.238	1.283	0.156	14.807	-0.243	1799.2	-0.144	33845	0.738
2021-07-25	4401.46	-0.471	1.295	0.935	14.779	-0.189	1804	0.267	35407	4.615
2021-07-26	4400.64	-0.019	1.239	-4.324	14.775	-0.027	1804.6	0.033	37296	5.335
2021-07-27	4419.15	0.421	1.238	-0.081	14.549	-1.530	1835.8	1.729	39563	6.078
2021-07-28	4395.26	-0.541	1.266	2.262	14.562	0.089	1817.2	-1.013	40004	1.115

APPENDIX B

Original financial asset time series data set extract

An extract of the various financial asset data sets, after the execution of Modules 4.2 to 4.3, is provided in Tables B.1 to B.5. The features x_{t-1} and x_{t-2} are employed as the input features, while '*Change %*' is set as the output (target) feature. The TS are utilised in the study conducted in Chapter 5.

TABLE B.1: *Feature engineered S&P 500 index time series data set extract.*

S&P 500 index index index time series extract			
Date	x_{t-2}	x_{t-1}	**Change (%)**
2001-01-02	-	-	-
2001-01-03	-	-0.010	-0.007
2001-01-04	0.050	-0.011	-0.026
2001-01-05	-0.011	-0.026	-0.002
2001-01-08	-0.026	-0.002	0.004
2001-01-09	-0.002	0.004	0.010
2001-01-10	0.004	0.010	0.010
2001-01-11	0.010	0.010	-0.006
2001-01-12	0.010	-0.006	0.006

TABLE B.2: *Feature engineered bond yield time series data set extract.*

Bond series data set extract			
Date	x_{t-2}	x_{t-1}	**Change (%)**
2001-01-02	-	-	-
2001-01-03	-	-0.022	-0.002
2001-01-04	4.735	-2.154	-2.261
2001-01-05	-2.154	-2.261	0.548
2001-01-08	-2.261	0.548	0.807
2001-01-09	0.548	0.807	2.082
2001-01-10	0.807	2.082	0.235
2001-01-11	2.082	0.235	2.797
2001-01-12	0.235	2.797	-0.400

TABLE B.3: *Feature engineered USD/ZAR time series data set extract.*

USD/ZAR series data set extract			
Date	x_{t-2}	x_{t-1}	**Change (%)**
2001-01-02	-	-	-
2001-01-03	-	-0.007	-0.002
2001-01-04	0.159	-0.699	-0.159
2001-01-05	-0.699	-0.159	0.998
2001-01-08	-0.159	0.998	1.910
2001-01-09	0.998	1.910	2.560
2001-01-10	1.910	2.560	-1.135
2001-01-11	2.560	-1.135	-0.230
2001-01-12	-1.134	-0.229	0.192

TABLE B.4: *Feature engineered gold time series data set extract.*

Gold series data set extract			
Date	x_{t-2}	x_{t-1}	**Change (%)**
2001-01-02	-	-	-
2001-01-03	-	-0.168	-0.214
2001-01-04	-0.149	-0.261	0.262
2001-01-05	-0.261	0.262	0.000
2001-01-08	0.262	0.000	-0.187
2001-01-09	0.000	-0.187	-1.047
2001-01-10	-0.187	-1.047	-0.264
2001-01-11	-1.047	-0.264	-0.038
2001-01-12	-0.264	-0.038	-0.227

TABLE B.5: *Feature engineered Bitcoin time series data set extract.*

Bitcoin series data set extract			
Date	x_{t-2}	x_{t-1}	**Change (%)**
2001-01-02	-	-	-
2001-01-03	-	-1.015	-1.754
2001-01-04	-1.639	-1.667	-3.390
2001-01-05	-1.667	-3.390	-5.263
2001-01-08	-3.390	-5.263	5.556
2001-01-09	-5.263	5.556	-1.754
2001-01-10	5.556	-1.754	3.571
2001-01-11	-1.754	3.571	1.724
2001-01-12	3.571	1.724	-5.085