FACULTEIT ECONOMIE
EN BEDRIJFSKUNDE

DYNAMICS OF FINANCIAL CRASHES AND STATISTICAL MECHANICS: AN ATTEMPT TO FIND A PREDICTOR OF MARKET INSTABILITY WITH LSTMS

Aantal woorden / Word count 16082

William Florin
Stamnummer / student number : 000130349004

Promotor / supervisor: Prof. Dr. Koen Schoors

Co-promotor / Co-supervisor: Prof. Dr. Jan Ryckebusch

Masterproef voorgedragen tot het bekomen van de graad van:
Master's Dissertation submitted to obtain the degree of:

Master in Business Engineering: Data Analytics

Academiejaar / Academic year: 2018-2019

UNIVERSITEIT
GENT

Foreword

"How we can predict the next financial crisis"

That is the title of a TED talk that somehow popped up on my Linkedin feed in July 2016. Being intrigued by the TED talk of Didier Sornette, I finished his book by the end of summer. Later that year, I reached out to Peter Cauwels who connected me with Ken Bastiaensen, which marked the beginning of my master thesis. I realize that this dissertation could not have been written without the persistent support of several people.

First of all, I want to express my greatest gratitude to my supervisor and mentor, Ken Bastiaensen, for his continuous support, constant assistance and willingness to answer my text messages at any time of the day. His practical approach, in-depth comments, easy communication and flexibility have allowed me to plan and organize my master thesis in a way that suits me best. He was the architect of this master dissertation idea, and pushed me to bring the execution to an as high level as possible. Further, I am obliged to thank him to let me discover my appreciation for quantitative finance and machine learning. I often realised how fortunate I was, having Mr. Bastiaensen as a mentor. I think few have experienced a mentor so committed to the research of a thesis student.

Secondly, I would like to express my deepest appreciation to Koen Schoors, for giving me the freedom to pursue this master dissertation. I truly admire his gift to explain the most complex matter in such a simple way. He assured that I always kept reflecting in an economic and financial context rather than just number crunching.

Further, I would like to thank Peter Cauwels for putting me in touch with Ken and for the useful insights that shaped this master disertation. Many thanks also go to Jan Rykenbusch, Andres Belaza and Corneel Casert for explaining me the ins and outs of

the magnificent world of statistical mechanics. Thank you Benjamin Vandermarliere for saving this thesis, as I would not have managed to write it in time without your golden tip of writing a technical report first.

A last but equally important thank you goes to my family and girlfriend for raising me with an open and curious mindset and supporting me in every thing I do: personally, academically and professionally.

Contents

List of used abbreviations

AI	Artificial intelligence
LPPL	Log-periodic power-law
JLS	Johansen-Ledoit-Sornette
US	United States
CDO	Collateralized debt obligation
S&P 500	Standards and poors 500
SSE	Shanghai Stock Exchange Index
BSEN	Bombay Stock Exchange Index
BVSP	Brazil Stock Exchange Index
BEL20	Brussels Stock Exchange Index
US10Y	US 10 Year Treasury
KCc1	Coffee Front Month Futures
LCOc1	Brent Crude Energy Future
ACF	Autocorrelation function
MSE	Mean square error
NN	Neural network
RNN	Recurrent neural network
LSTM	Long short-term memory
FCN	Fully Convolutional Network
ROC	Receiver operating characteristic
AUC	Area under the curve
TP	True positives
FP	False positives
TN	True negatives

FP	False positives
FN	False negatives

List of Figures

List of Tables

Chapter 1

Introduction

Financial speculative bubbles have a long history and can be at least traced back to ancient Rome (Chancellor, 1999). As society always tries to find solutions to complex problems, multiple attempts have been made to model and predict financial bubbles. With the increase of computer power and the enhanced accessibility of data in the 1980s, an enormous wave of theories arose. Unfortunately, all models were mainly descriptive rather than predictive. Until Sornette, Johansen, and Bouchaud (1996) proposed a model that was capable of predicting when a crash would occur. The model has evolved a lot since 1996 but still has difficulties to predict crashes with high precision. The superior performance of artificial intelligence offers renewed hope of finding a model capable of predicting crashes and preventing them from happening through arbitrage.

This master dissertation will attempt to build an AI model capable of predicting crashes in the financial markets. The biggest downside of AI is that much data is needed to make precise predictions, data which is not available as crashes rarely occur. This thesis will try to overcome this problem by training the model on synthetic data that resembles real financial data. The synthetic data will be generated through a financial agent model, originating from statistical mechanics, which attempts to grasp the herding behavior present in financial actors.

This master dissertation will begin with a concise literature review where the objective of this work is contextualized and theoretically framed (chapter 2). It consists of two main building blocks: first, explaining the dynamics behind crashes and second, discussing crash detection theories with a special emphasis on the log-periodic power-law. Subsequently, the research question, the accompanying hypotheses and the methodology for testing these hypotheses will be outlined (chapter 3.1). Once there, the experiment starts which consists of three central phases. The first phase is devoted to finding the phase transition model capable of generating series that resemble real financial time series (chapter 4). The second phase involves training the machine learning model on the synthetic in order to predict crashes (chapter 5). Lastly, that model is applied and evaluated on real financial data (chapter 6).

Chapter 2

Literature

This chapter provides a comprehensive overview of what has been written in the literature about bubbles and crashes. The chapter is structured into two main segments. The first segment tries to cover the characteristics of a crash, while the second segment covers theories of crash detection. At the end of this chapter we go in more depth about log-periodic power-law, which is considered as the highest regarded bubble detection theory of the last two decades.

2.1 Dynamics of financial crashes

From a miner in Katanga-Congo to a small business owner in Deinze-Belgium, everyone felt the effects of the financial meltdown in 2008. To prevent such a global catastrophe, the consensus among economists was that new bubble detecting models are needed. Before evaluating these models, it is important to have a grasp of the dynamics behind crashes.

2.1.1 The five stages of a bubble

Even tough economists agreed on the need for new mechanisms to detect bubbles, they unfortunately don't agree on what a bubble is (Geraskin, 2013). Minsky (1982) provided an early, informal characterization of bubbles and the associated busts. According to

Minsky, capitalism has an inherent tendency to financial instability: periods of boom are followed by financial crashes. In what will follow is an explanation throughout the five stages of a bubble according to Charles P. Kindleberger (1978) and Minsky (1982).

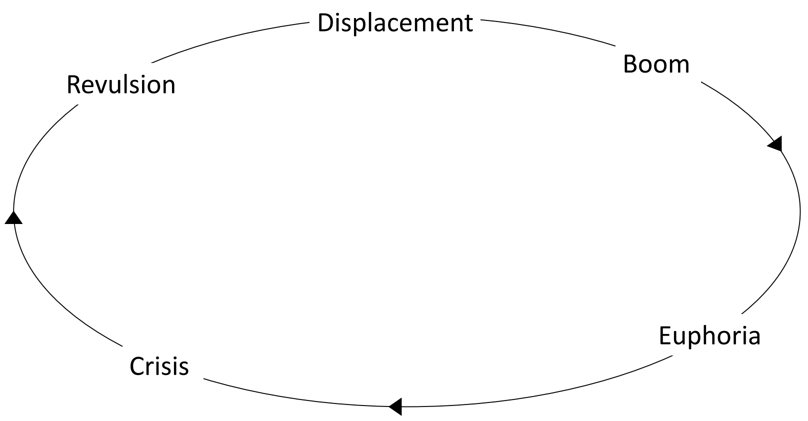


Figure 2.1: Kindleberger-Minsky Model

Stage 1 Displacement: All bubbles start with a fundamental change in the economy. It is often stated that a disruptive technology starts the bubble years before it crashes. The breakthrough of the internet led to the Dot-com bubble in the 2000s. Kindleberger says it does not necessary needs to be technology that is at the roots of a bubble. For example, the opening of Russia in 1990 led to the crash in 1998. In the displacement phase, the attention of investors is drawn upon the fundamental change in the economy and they start to invest in it. The first air into the financial bubble balloon is blown.

Stage 2 Boom: The price of certain assets gain momentum in this phase. That momentum is driven by two main dynamics. The first dynamic is the widespread media coverage of the fundamental change in the economy. People start talking about it, they have a fear of missing out and start to invest in the financial asset. The historical example of this phase is the Tullip Mania in the 17^{th} century. The enormous media attention Bitcoin and Blockchain received in autumn of 2017 can be considered as a more recent example. This dynamic is often referred in the literature as herding behaviour or positive feedback (section 2.2.3.2). The second dynamic that drives the boom phase is the credit creation by central banks globally. This will cause banks and financial intuitions to write out more loans to both private individuals and businesses. A money flow towards the booming assets is generated. Those two dynamics combined will generate the balloon to inflate.

Stage 3 Euphoria: In this phase asset prices are going through the roof. Speculation is becoming extremely excessive. To meet the speculation, financial institutions are inventing new financial products that are meeting the exact needs of speculators. Further, there's a general belief that everyone can make money. In the sub prime boom, hunderds of TV Channels had programs about people who became house flippers (Depken et al., 2011). Lastly, the valuation of the asset reaches extreme levels. They are inventing new measures to adapt to the new reality. In the analogy of our balloon, the balloon is extending and stretching its limits.

Stage 4 Crisis: This phase is characterized by the fact that insiders are cashing out, this is followed by a financial distress. Insiders realize that the growth of the asset is not sustainable and decide to sell. The price might still increase as long as outsiders are willing to purchase the assets. This is followed at some point with a period of financial distress. Investors and companies come to the realization that it is time to become more liquid. The consequence is that they sell more assets which lower the prices. Some highly leveraged investors may tumble into bankruptcy because the decline in the prices is so large that the market value of their securities and real estate decline below their indebtedness. This causes the general public to lose their trust and they try to sell all their assets before it is too late. The balloon has reached its maximum capacity and is deflating rapidly now.

Stage 5 Revulsion: In the final stage, investors are so scared that they don't participate in the market at all. Banks are cautious to provide credit as their collateral has decreased significantly. Further, there's overall panic that decreases the trading volume enormously. After the failure of Lehman Brother in 2008, the trust was so damaged that even the trading volume in the interbank market decreased to unseen levels which endangered the world economy. This phase can be seen as the balloon popping.

The panic feeds itself until the confidence is restored. This can mainly happen through one of three possible dynamics. First, prices are so low beneath their fundamental value that investors start investing again. The second dynamic is that exchanges place limits on the maximum daily decline. Last and most used dynamic is the lender of last resort who steps in to provide liquidity in the market that increases the credit creation.

2.1.2 Endogenous versus exogenous crashes

Two types of crashes can be classified according to Johansen and Sornette (2002). On one hand events of an endogenous origin which are associated with preceding speculative crashes. Secondly, events of exogenous origin associated with the markets response to external shocks. The housing market crash in 2008 can be classified as endogenous while the crash in the airline industry following the 9/11 attacks can be considered as an exogenous crash. Of the 47 outliers in financial drawdowns found by Johansen and Sornette (2010) 25 were classified as endogenous crashes and 22 were a response of an exogenous shock.

2.1.3 Heterogeneity of traders

An important concept that has been increasingly researched is the heterogeneity of traders. Heterogeneity of traders means that not all the agents have the same expectations for assets. E. M. Miller (1977) states that the heterogeneity of traders together with the restriction of short selling is the reason for security overvaluation. During the internet bubble, there were very optimistic agents who believed in internet stocks. On the other hand there were pessimistic agents but they were limited in trading as short selling is restricted. Those two factors combined result in the price being overvalued and leading to a bubble as pessimistic traders aren't able to express their pessimism sufficiently (Huang et al., 2010). The two main agents who are reluctant to short stocks: are mutual funds (Chen et al., 2002), and hedge funds (Shleifer and Vishny, 1997) as they may avoid risk-adjusted excess return trades in highly volatile settings.

Harrison and Kreps (1978) build further upon the model of Miller, they claim that the optimistic agent is willing to pay a premium on their valuation of the price as the agent anticipates that an even more optimistic agent will buy the asset in the future. This behaviour is described by Keynes (1936) as speculative behaviour as the agent is willing to pay more for the asset than they would pay if they would be obliged to hold it forever. Such speculative behavior leads to a crash component in asset prices. Harrison and Kreps (1978) further state that crashes come after a period of high trading volume and high price volatility. The latter has been proven wrong by Sornette, Cauwels, and Smilyanov (2017), they proved that volatility is not a good predictor for an upcoming crash.

2.2 Theories in crash detection

Given the significant impact of financial crashes on people's lives, continuous attempts are made to prevent financial crashes from happening. To take the right preventive measures, one must be able to determine when assets are in the crash-danger zone. Several efforts are made to model crashes with statistical models. The theories can be classified in two main categories: rational crash theories (2.2.1) and behavioural crash theories (2.2.2). In what will follow is a short explanation of the theories and statistical models in each field. The following section is inspired by two research papers: the one of Forró (2015) and of Yang (2006).

2.2.1 Rational crash theories

Rational crash models are based on the assumption of full rationality of investors and are therefore formed in the context of the present value of dividends models (M. Miller and Modigliani, 1961). This concept states that the present value of C in t years with an annual return of r is:

$$PV = \frac{C}{(1+r)^t} \tag{2.1}$$

2.2.1.1 The variance bound test

The variance bound test introduced by Shiller (1981), states that the variability of the dividend sets an upper bound to the variability of the stock price. p_t is defined as the stock price at time t and d_t as the dividend during period t.

$$p_t = \frac{E(d_t)}{1+r} + \frac{E(d_{t+1})}{(1+r)^2} \tag{2.2}$$

Define p_t^* as the present value of the actual dividends,

$$p_t^* = \frac{d_t}{1+r} + \frac{d_{t+1}}{(1+r)^2} \tag{2.3}$$

Economists make a difference between the ex-ante and the ex-post price. The ex-ante price p_t is the current price of the asset based on the present value of the expected dividend streams while the ex-post price p_t^* is based on the actual dividends. We write

$$p_t^* = p_t + e_t \tag{2.4}$$

with e_t the forecast error.

In a rational forecast the prediction error should have the following properties: (i) the error should have mean zero (ii) the error should be uncorrelated with the forecast.

As p_t and e_t are uncorrelated and the variance of the sum of two uncorrelated variables is the sum of the variances we can write:

$$Var(p_t^*) = Var(p_t) + Var(e_t) \tag{2.5}$$

The variance of the ex-post rational price can be split into the variance of the ex-ante forecast and of the forecast error. By consequence the variance of the ex-post real price should be an upper bound for the ex-ante forecast.

$$Var(p_t^*) \geq Var(p_t) \tag{2.6}$$

If the upper bound of the variance of the forecast price is violated, Tirole (1985) has suggested that it is caused by a crash. The problem with the variance bound test is that it can only be executed ex-post and can't be used as a predictor for upcoming crashes. Further, a lot of aggregated data over a long period of time is needed to avoid small sample bias.

2.2.2 Behavioural Bubble Theories

Behavioural finance is based upon human psychology and tries to explain why people make certain investment decisions. Behavioural financial theories have been introduced in the 1970s by the "economic superstars" such as Daniel Kahneman, Amos Tversky and Richard Thaler. The main characteristic of behavioral finance is that agents are not as rational as is claimed in traditional economic theories. Agents are driven by certain biases and heurestics.

2.2.2.1 Simple behavioural model

Delong et al. (2009) composed a really simple behavioural financial bubble model based upon the maniac, panics and crashes of Charles P. Kindleberger (1978). The model makes the assumption that there aren't any rational agents who understand the price dynamics and attempt to profit from them. An agent can decide to either buy stocks or bonds with p_t be the total amount of wealth invested in stocks. A stock pays a dividend d_t with probability π. By consequence there's a 1 - π probability that a stock doesn't pay dividends. The bond pays a fixed rate of return of r. Each investor does the exact same thing as he did in the previous period except a number equal to λ times the difference in rates of return switch from the lower to the higher-performing strategy for the next period. As a result if an agent owns a stock that doesn't pay a dividend and he encounters an agent that owns bonds he would switch to bonds. Taken all the assumptions into consideration, the mathematical equation of the model is then:

$$p_{t+1} = p_t + \lambda p_t(1 - p_t)\left(\frac{p_t - p_{t-1} + d_t}{p_{t-1}} - r\right) \tag{2.7}$$

Taking unconditional expectations:

$$E(\Delta p_{t+1}) = \lambda p_t(1 - p_t)\left(\frac{E(\Delta p_t) + \pi\delta}{p_{t-1}} - r\right) \tag{2.8}$$

When $E(\Delta p_t) = 0$ then $E(\Delta p_{t+1}) = 0$. Then p_{t-1} is

$$p_{t-1} = \frac{\pi\delta}{r} \tag{2.9}$$

This is the 'fundamental' value of p, denoted as p^*.

Table 2.1: Parameters of the simulation of the simplest behavioural financial bubble model

Parameter	π	$1-\pi$	δ	r	λ	p^*	p_0
Value	0.5	0.5	0.05	0.05	1.5	0.5	0.25

The simulation has been done with t = 100 which can be considered as years. The stocks are considered to be overvalued (bubble behaviour) when the price is higher than

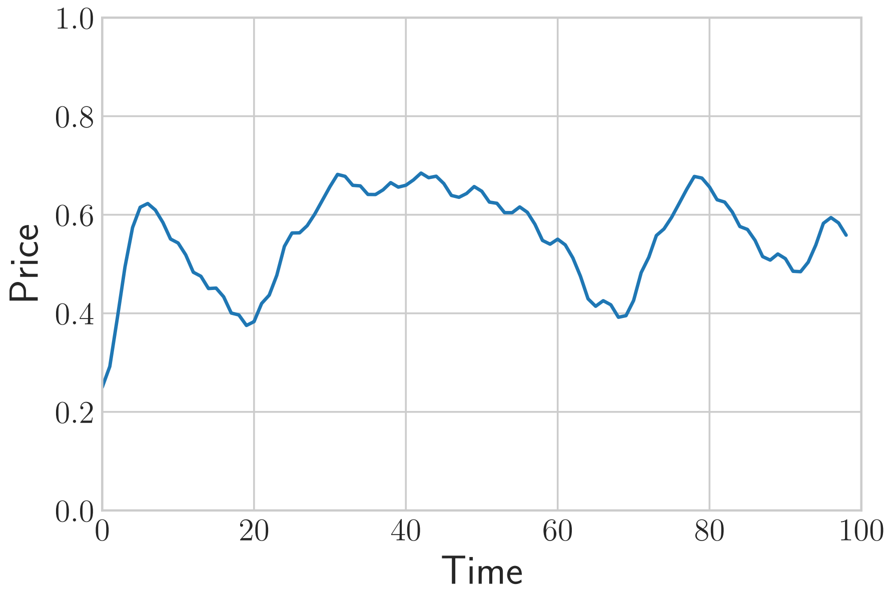


Figure 2.2: Simulation of the simplest behavioural financial bubble model

the fundamental value, 0.5 and undervalued when it's lower than 0.5. If the price is too high, the yields go down and the bonds are more interesting. If there aren't any dividends paid for two consecutively years, the price tumbles and a crash occurs.

2.2.3 Log-periodic power-law

Although many models attempt to describe bubbles, explaining and predicting remains a big problem (Gürkaynak, 2008). The JLS-model, which was first proposed by Sornette, Johansen, and Bouchaud (1996) and later formalized by Johansen, Ledoit, et al. (1998), distinguish itself by explaining the dynamics behind a bubble and proposing a functional form for the price dynamics up to a crash. The theory starts from applying the science of complexity on the stock market. The science of complexity studies a wide variety of field. Examples of complex systems can be found in the human body with the dynamics of neurons in the brain (Kinouchi and Copelli, 2006), military conflicts with the Syrian war (Parens and Bar-Yam, 2016) and business with the dynamics behind teams (Bar-Yam and Kantor, 2018).

2.2.3.1 General idea

The model is built upon the behavioural phenomenon of positive-feedback (subsection 2.2.3.2). The positive feedback leads to a super-exponential growth instead of a fixed growth rate. As the price increases exponentially, it should go to infinity. This is unsustainable and not realistic and therefore bound to undergo a regime change.

The Johansen-Ledoit-Sornette model (JLS Model) aims to describe the evolution of the pattern of stock prices in an unsustainable regime. It describes two main dynamics, the first is the super-exponential growth(supra). The second are log-periodic oscillations of the price with decreasing amplitude that reflect human grouping patterns (Sornette and Cauwels, 2014). Everything comes together in the log-periodic power-law(LPPL) where the expected value of the price until t_c is:

$$E[Log(p_t)] = \underbrace{A + B(t_c - t)^m}_{\text{Super exponential growth}} + \underbrace{C(t_c - t)^m \cdot cos(\omega log(t_c - t) - \phi)}_{\text{Log-Periodic Oscillations}} \tag{2.10}$$

where:

A	Log-price at time of crash
B	Magnitude of the power law
C	Amplitude of log-periodic Oscillations
m	Super-exponential growth with $0 < \text{m} < 15$
ω	Angular frequency of social hierarchies
ϕ	Oscillation time scale

2.2.3.2 Positive feedback mechanism

Sornette and Johansen (2002) claim that there are rare but anomalous large drawdowns in all markets. Since investors dictate the prices through supply and demand, crashes occur when too many investors sell at the same time. As any move in the stock market needs to be traced back to the behaviour of the investors the question arises: 'What mechanism causes such a coordinated sell-off' (Johansen and Sornette, 1999).

This section does an attempt in answering this question. We will start of with explaining the difference between positive and negative feedback mechanisms. Next, we sketch a more general mechanism for positive feedback which is called 'herd' or 'crowd' behaviour.

We end by explaining forces of imitation which is the driver behind herding.

The concept of "positive feedback" can be associated to the idea of dominant design. The more people use a certain product the more useful it becomes (cfr. Whatsapp). Positive feedback can cause bubbles to grow and eventually crash, while negative feedback tend to move prices to the fundamental value. For example, in population dynamics, the larger the population of rabbits in a valley, the less grass there is per rabbit. If the population grows too much, the rabbits eventually die of hunger, slowing down their reproduction rate, which reduces their population later. Consequently, negative feedback means that the higher the population, the slower the growth rate, which leads to a spontaneous regulation of the population size; negative feedback tends to regulate growth in the direction of balance. Positive feedback, by contrast, claims that the higher the price or price return in the recent past, the higher the future price growth (Sornette, 2003).

Herding behaviour is an example of a positive feedback mechanism. It is often said to occur when many people take the same actions. There are different dynamics behind herding behaviour. Humans are sociable creatures and have a great desire to be accepted by their peers. Imitating the actions of a larger group seems to be a natural way of becoming a member of that group. The main forces of imitation are explained in the next paragraph. A second dynamic behind herding is that the more people buy into a certain idea, the less likely the idea is wrong (Shiller, 2015). Bouri et al. (2018) found empirical evidence of herding behaviour in the cryptocurrency crash of December 2017.

Each agent is influenced by two types of information. Information of the agent's local network like family, friends, media (a). Secondly, an idiosyncratic signal that only the agent receives (b). (a) will create order as agents imitate each other while (b) will create disorder as agents make independent decisions. In normal market conditions, the fight between the two forces results in an equal amount of people who want to buy and sell. When order (a) wins the fight, too many agents sell or buy at the same time which results in a bubble regime.

2.2.3.3 Short derivation of the log-periodic power-law

The JLS model is built on the rational expectation theory of Blanchard and Watson (1982) which states that an asset's observed price p_0 can be written as:

$$p_o = p^* + p \tag{2.11}$$

with p^* the fundamental value and p the bubble component. The JLS model assumes that the market does not pay dividends and that interest rates, aversion to risks and market liquidity are negligible (Johansen, Ledoit, et al., 1998). Therefore, the fundamental price p^* is always 0 and the JLS model focus solely on the bubble component.

The model assumes that the bubble component p follows a stochastic process:

$$dp_t = \mu(t)p_t dt + \sigma(t)p_t dW_t - \kappa p_t dj \tag{2.12}$$

with:

$\mu(t)$	Drift component
$\sigma(t)$	Volatility
dW_t	Increment of a standard Wiener process with zero mean and unit variance
dj	Discontinuous jump with $dj = 0$ before the crash and $dj = 1$ after the crash
κ	Amplitude of the jump dj

$$dj = \begin{cases} 1, \; with\, probability\; h(t)dt \\ 0, \; with\, probability\; (1 - h(t))dt \end{cases} \tag{2.13}$$

The dynamics of the jump dj are driven by $h(t)$, the crash hazard rate, which indicates the probability of a crash occurring per unit time if it has not yet occurred and is defined by Johansen, Sornette, and Ledoit (1999) as:

$$h(t) \approx \beta(t_c - t)^{m-1} + C(t_c - t)^{m-1} \cdot cos[\omega log(t_c - t) + \phi] \tag{2.14}$$

The hazard rate portrays the interactions between an investor network that exhibits herding behaviour. Equation 2.14 contains two main characteristics. Firstly, it models a hyperbolic power law growth, which ends in a finite-time singularity t_c. This is a reflection of the positive feedback mechanism discussed in section 2.2.3.2. Secondly, the cosine parts portraits the accelerating panic punctuating the growth of the bubble (Sornette, Woodard, et al., 2013a). The reason why 2.14 is an approximation is because it is derived from a Taylor expansion. The expectation of a crash happening can be written as following:

$$E_t[dj] = 1 \times h(t)dt + 0 \times (1 - h(t)dt) = h(t)dt \tag{2.15}$$

The expectation of the price can be derived out of equation 2.12:

$$\begin{aligned} &E_t[\tfrac{dp_t}{p_t}] = \mu(t)dt + \sigma(t)E_t[dW] - \kappa E_t[dj] \\ &= \mu(t)dt + \sigma(t)(0) - \kappa E_t[dj] \\ &= \mu(t)dt - \kappa E_t[dj] \end{aligned} \tag{2.16}$$

Plugging equation 2.16 in 2.16 and taking the assumption of no-arbitrage into account, $E_t[dp_t] = 0$ (Blanchard and Watson, 1982) we become:

$$\begin{aligned} &E_t[\tfrac{dp_t}{p_t}] = \mu_t dt - \kappa h(t)dt = 0 \\ &\Longleftrightarrow \\ &\mu(t)dt = \kappa h(t)dt \end{aligned} \tag{2.17}$$

meaning that the drift $\mu(t)$ is a function of the crash hazard rate $h(t)$. This can be understood intuitively, as the the higher the likelihood of a crash, the higher the return of the asset need to be, in order to compensate an investor for taking more risks. Equation 2.12 can be rewritten as conditional on the fact that a crash has not happened yet:

$$\begin{aligned} &\tfrac{dp_t}{p_t} = \mu t + \sigma(t)dW_t \\ &= \kappa h(t)dt + \sigma dW_t \end{aligned} \tag{2.18}$$

Its conditional expectation results in:

$$E_t[\frac{dp_t}{p_t}] = \kappa h(t)dt \tag{2.19}$$

$$E[ln(\frac{dp_t}{p_t})] = \kappa \int_0^t h(t')dt' \tag{2.20}$$

Substitute equation 2.14 in 2.20 and integrating results in the log-periodic power law

$$ln(E[p_t]) = A + B(t_c - t)^m + C(t_c - t) \cdot cos(\omega ln(t_c - t) - \phi) \tag{2.21}$$

which is the same as equation (2.10)

2.2.3.4 Fitting procedure

Fitting equation 2.10 on financial data requires 7 parameters to be fitted: $A, B, C, m, \omega, \phi, t_c$. Local optimization algorithms are often trapped in local minima. To overcome this problem, Filimonov and Sornette (2013) proposed a robust form to fit the JLS-model. It reduces it to a function of only three non-linear parameters(t_c, ω, m).For a robust and realistic fit, there are some constraint on the parameters that emerged from the analysis of historical bubbles.

- $0.1 \leq m \leq 0.9$
- $6 \leq \omega \leq 13$
- $|C| < 1$
- B <0

Constraint on m ensures that the probability of a crash remains finite and smaller than 1 for t $\leq t_c$. Further, it reduces type I errors (rejecting the LPPL hypothesis when it is true). The limitations on ω prevents the log-periodic oscillations from being too frequent, thus fitting random noise, nor being too rare to contribute to the trend.

Figure 2.3 shows the an LPPL fit on BTC on the time period from 2017-03-20 till 2017-11-25, the fit predicted a crash on 2017-12-10 while the actual crash happened on 2017-12-18. The LPPL was fitten with the code provided by Yichuan (2014).

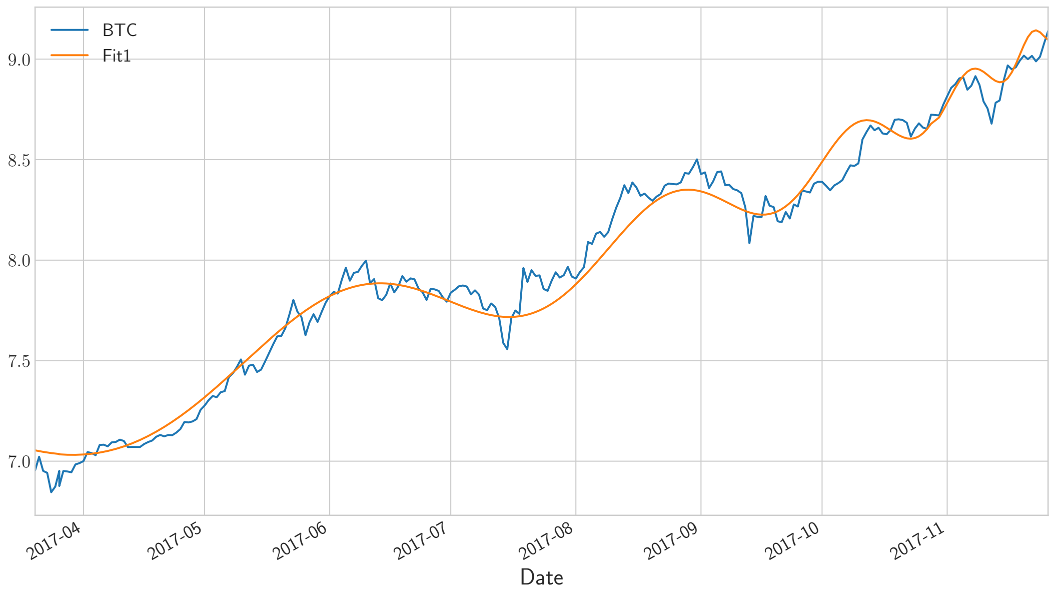


Figure 2.3: LPPL fit on BTC with estimated T_c 10/12/2017

2.2.3.5 Criticism on LPPL

The most significant and thorough critique against the LPPL was made by Chang and Feigenbaum (2006), who studied the mechanism underlying the LPPL using Bayesian techniques applied to the returns. They showed that a model withouth log-periodical oscillations outperfroms the JLS-model.

A second flaw of the LPPL is that it only can predict endogenous bubbles (2.1.2). As approximately 40% of the bubbles are exogenous (Johansen and Sornette, 2002) the predictiveness of general bubbles is limited.

Bree and Lael Joseph (2010) has found that many published predictive LPPL's had variables that violated the hard constraints set for the JLS-model (2.2.3.4). They either allowed the fitted LPPL to decrease at some point or fitted on the raw data rather than the log. This suggests that the LPPL's predictability is less accurate than shown.

Chapter 3

Research question and methodology

3.1 Research question and hypotheses

Until now, the reader has been introduced to the topic of financial crashes by discussing the dynamics behind crashes and reviewing methods that attempt to predict crashes. So far, the conclusion can be drawn that few methods are successful in predicting crashes. The only method, that realized to predict crashes ex-ante is the log-periodic power-law discussed in section 2.10. However, many critics have shown that the precision of the LPPL is doubtful and the fitting procedure is cumbersome (Brée et al., 2013).

However, this work believes in the underlying foundation of the LPPL, namely that prices are formed by interacting agents, who have a certain influence on each other. This idea is based on the premise that people are social animals and demonstrate herding behaviour as outlined in 2.2.3.2. Zhou and Sornette (2007b) translate this concept to the Ising model, a popular model in statistical mechanics used to model phase transitions.

This work will depart from the roots of the LPPL, the concept of herding behavior modeled by a phase transition model, and will attempt to compose a predictive model capable of predicting crashes. More concrete, synthetic financial data will be generated by a phase transition model. A machine learning model will be trained on the synthetic data, which is then applied on financial time series to evaluate its predictive power. This is academically formalized in the following research question:

"Can a machine learning model that is trained on synthetic data, generated by a phase transition model, predict crashes in the financial market?"

The research question will be answered by two main evaluation criteria. First, a confusion matrix will be composed to verify how precise and accurate the model performs on a test set. This is grasped in the following hypotheses set:

- H_0 The efficient market hypothesis holds, and the accuracy and precision of the model is no higher than a randomly guessing model
- H_1 The efficient market hypothesis does not hold, and the accuracy and precision of the model outperform a randomly guessing model

Secondly, the model will be evaluated through a simple trading strategy, which is translated into the following hypotheses set:

- H_0 The efficient market hypothesis holds, and the returns generated by the trading strategy do not outperform a buy and hold strategy
- H_1 The efficient market hypothesis does not hold, and the returns generated by the trading strategy do outperform a buy and hold strategy

3.2 Methodology

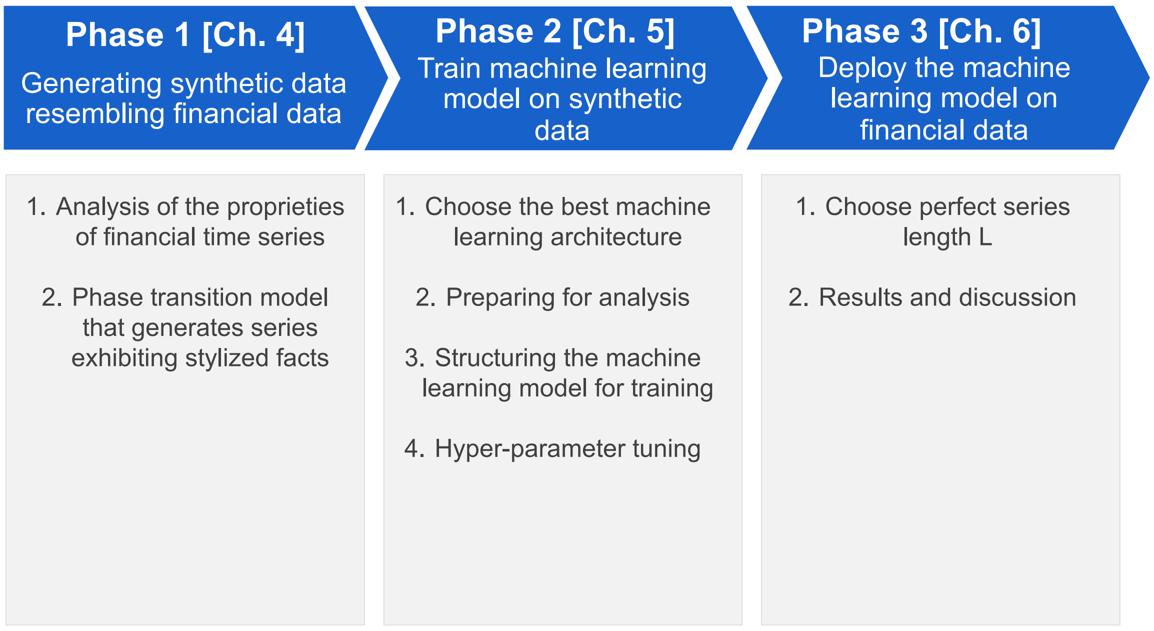

Phase 1 [Ch. 4]
Generating synthetic data resembling financial data
Phase 2 [Ch. 5]
Train machine learning model on synthetic data
Phase 3 [Ch. 6]
Deploy the machine learning model on financial data
1. Analysis of the proprieties of financial time series
2. Phase transition model that generates series exhibiting stylized facts
1. Choose the best machine learning architecture
2. Preparing for analysis
3. Structuring the machine learning model for training
4. Hyper-parameter tuning
1. Choose perfect series length L
2. Results and discussion

Figure 3.1: Methodology

As illustrated in figure 3.1, the methodology consists of three main phases. The first phase is devoted to finding the phase transition model capable of generating series that resemble real financial time series (chapter 4). The second phase consists of training a machine learning model on the synthetic data (chapter 5). The last phase is deploying and evaluating the machine learning model on real financial data (chapter 6). Each phase is structured as a chapter, while each bullet point is structured as a section.

3.3 Computational setup

Most code was executed in Python except the generation of the synthetic data, which was written in Julia, because of Julias superior loop performance. All machine learning models were trained using the Keras (Chollet et al., 2015) library with the TensorFlow (Martın Abadi et al., 2015) backend. The scripts of phase 1 were run on my local computer on a 4 core cluster while phase 2 and 3 were run on two NVIDIA Tesla P100 GPUs on Google cloud. The machine learning models have ran for 91.8 hours on the two GPU's.

Chapter 4

Generating synthetic data resembling financial data

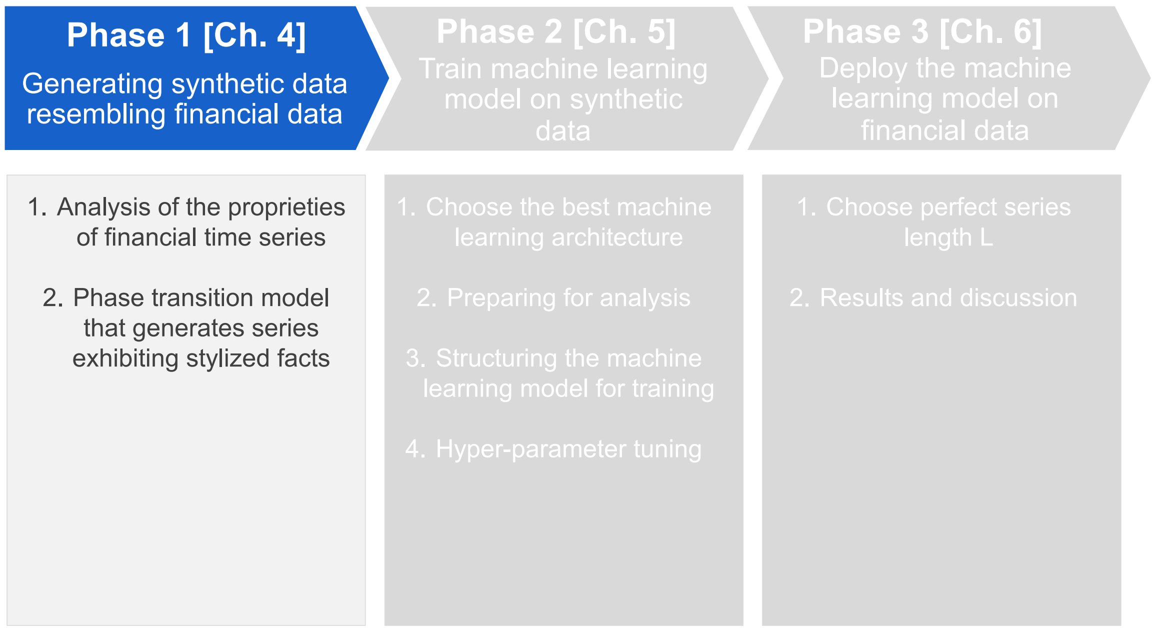

Phase 1 [Ch. 4]
Generating synthetic data resembling financial data
Phase 2 [Ch. 5]
Train machine learning model on synthetic data
Phase 3 [Ch. 6]
Deploy the machine learning model on financial data
1. Analysis of the proprieties of financial time series
2. Phase transition model that generates series exhibiting stylized facts
1. Choose the best machine learning architecture
2. Preparing for analysis
3. Structuring the machine learning model for training
4. Hyper-parameter tuning
1. Choose perfect series length L
2. Results and discussion

Figure 4.1: Phase 1: Generating synthetic data resembling financial data

The ultimate goal of this thesis is to deploy a machine learning model that can predict financial crashes. The machine learning model will be initially trained on synthetic data. This chapter is devoted to generating the synthetic data that resembles financial data. This chapter consists of two main sections. Section 4.1 analyses the proprieties of financial assets. Section 4.2 is devoted to finding the best phase transition model that suits our goal.

4.1 Analysis of the properties of financial time series

The goal of this section is to analyze the characteristics of financial time series. This section is structured as follows. First, appropriate assets need to be identified to derive the features from (subsection 4.1.1). Secondly, those assets are prepared for analysis in subsection 4.1.2. Thereafter, proprieties that are most important for the problem of crash detection are selected in subsection 4.1.3. The remainder of the subsection is then devoted to quantifying those proprieties.

4.1.1 Selection of representative assets

It is essential to choose the correct assets to quantify the proprieties. The assets must meet two main conditions: they must be collectively exhaustive and they must also prove to be helpful for answering our research question.

To begin, this research opted for extremely liquid assets as they are valuable for the experiment for two central reasons. First, it is hard to act upon a possible bubble in an illiquid market (cfr. 2008 US house market bubble, where it was extremely hard to get out of illiquid CDO's). Secondly, it has been proven that liquid assets are more efficient (Chordia et al., 2008), which will help us to receive a strong crash signal as no anomalies will be found in the price.

Secondly, the choice was made to diversify over different asset classes (equities, bonds, currencies, commodities) while keeping some affinity with the Belgian market. Further, an attempt was made to be diversified over the developed and emerging markets. Table 4.1 outlines the assets on which the analysis will be performed. They are the liquidest assets over the emerging and advanced economies of each asset class. In order to keep the body of this work somewhat dense, only one asset of each asset class will be discussed in the corpse. The assets that will be examined in the main text are: (i) equities: S&P 500 (ii) bonds: US10Y (iii) currency pairs: EURUSD (iv) commodities: KCc1. The analysis of the other assets can be found in the appendix and will be referred to in square brackets.

The daily close price was downloaded from Thomas Reuters from the date the asset was listed on a daily basis until Friday 3/05/2019. This work opted for the daily close price as hourly ticks are too volatile while monthly data is too smooth.

Asset class	Asset name	Asset acronym	Start year
Equities	S&P 500 [USA]	S&P 500	1950
	Nikkei 225 [JPN]	N225	1965
	Shanghai Stock Exchange Index [CHN]	SSE	1996
	Bombay Stock Exchange Index [IND]	BSEN	1997
	Brazil Stock Exchange Index [BRA]	BVSP	2002
	Brussels Stock Exchange Index [BEL]	BEL20	1991
Bonds	US 10 Year Treasury [USA]	US10Y	1980
	Belgian 10 Year Bond Yield [BEL]	BEF10Y	1993
Currencies	EURUSD [EUR - USA]	EURUSD	1975
	USDJPY [USA - JPN]	USDJPY	1987
Commodities	Coffee Front Month Futures	KCc1	1980
	Brent Crude Energy Future	LCOc1	1970

Table 4.1: Assets covered in this work

To have a clean view of the accuracy, which is calculated on these assets in chapter 6, the correlation between the different assets should be negligible. This is the case for the selected assets since the correlation between the different assets, as shown in figure 4.2 [A.1 in appendix], is small . The following subsection is dedicated to preparing the data for analysis.

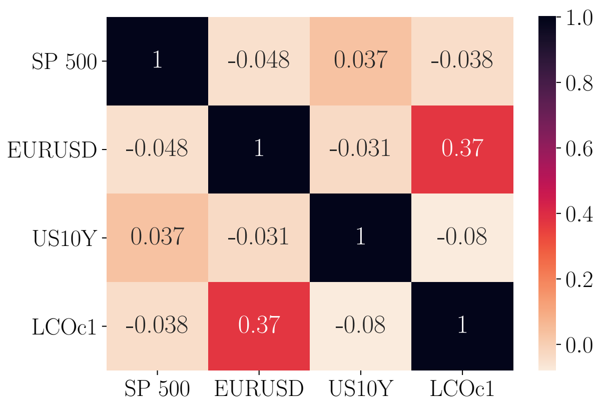


Figure 4.2: Correlation between the various assets

4.1.2 Data preparation: from prices to standardized returns

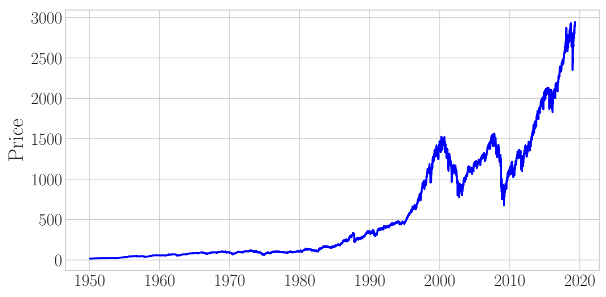
Campbell et al. (1997) makes the claim to use returns rather than prices as they have more attractive statistical proprieties and are scale free. Prices always depend on the previous price while returns are stationary. However, there are two main definitions of returns: raw returns and log-returns. Log-returns are more useful for our analysis as multi period returns are calculated through addition rather than multiplication. The main advantage is that statistical proprieties are easier derived from additive processes than from multiplicative processes. The main drawback of log-returns is calculating the weighted returns of a portfolio, which is not necessary in this case.

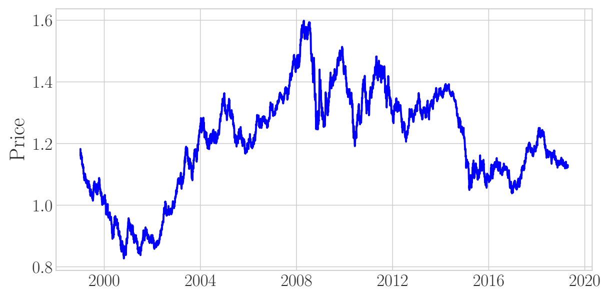
$$R = \log(\frac{P_t}{P_{t-1}}) = log(P_t) - log(P_{t-1}) : \tag{4.1}$$

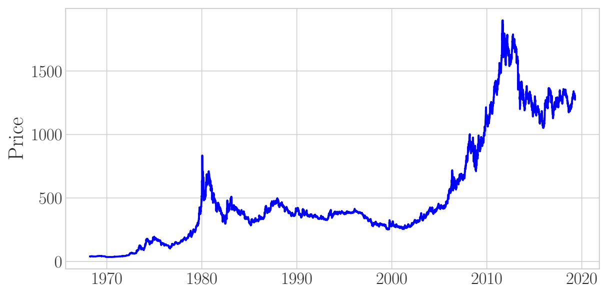
In chapter 5, a model will be trained both on synthetic and real financial data. Therefore the returns should be transformed so that both synthetic and real financial data are similar. This can be achieved by transforming the log-returns to standardised returns as following:

$$R_{standardised} = \frac{R - R_\mu}{R_\sigma} \tag{4.2}$$

From each return the mean is subtracted which is then divided by the standard deviation of the series. These resulting returns tell us how many standard deviations the price moved in regards to the mean. Figure 4.3 shows the trading price of the assets from the day they were listed till 3/05/2019 on a daily basis. The standardised returns are shown in figure 4.4. The most remarkable crashes can be easily identified: f.e. Black Monday, which is the largest daily drawdown in history of the S&P 500 or the housing crash in 2008.


(a) S&P 500


(b) EUR USD


(c) LLoC1

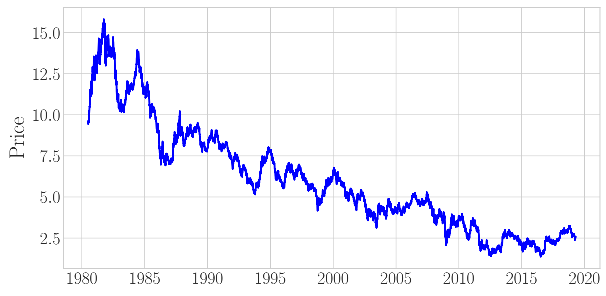
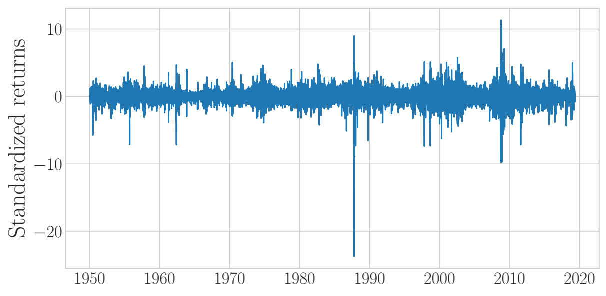
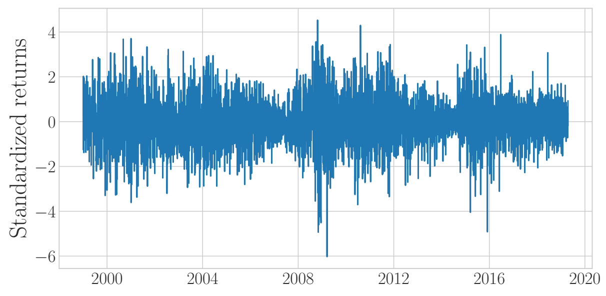
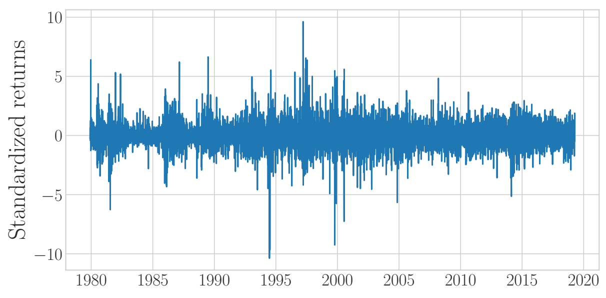

(d) US10Y

Figure 4.3: Price evolution of various assets


(a) S&P 500


(b) EUR USD


(c) LLoC1

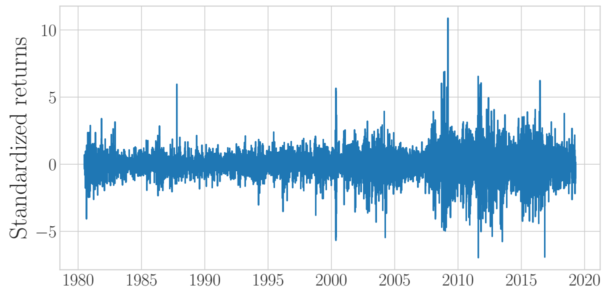

(d) US10Y

Figure 4.4: Daily log standardised returns

4.1.3 Selecting relevant stylised facts

As the previous subsection succeeded in selecting appropriate assets and transforming those for analysis, the third step is to select the relevant characteristics. All the characteristics, or stylised facts as they are called for financial time series, are nicely outlined by Aste (2013).

From all these stylized facts, we select those that are dependent on variables that can be calculated from the price. The reason for this is that this work seeks to construct a model that can predict crashes only using a price-related measure as input. Four stylised facts meet this requirement, the first, and most important, is the presence of fat tails (Mandelbrot, 1963) which will be discussed in paragraph 4.1.4.1. Horák and Smid (2009) claim that the tails of falls are fatter than of rises. However we will not examine this as it does not hold for currency pairs since the rise of the base currency is the loss of the quote currency. For the purpose of this thesis we will solely stick to the presence of fat tails. Secondly, Fama (1965) has found that the returns do not exhibit autocorrelation (paragraph 4.1.4.2), while the absolute and squared returns do. The latter is evidence for what is called Volatility Clustering (Mandelbrot, 1963) which is discussed in paragraph 4.1.4.3. Scaling, found by Cont (2001b), is solely price dependent, but it is not relevant as we previously assumed to work only with daily prices. As a result, three main stylised facts will be quantified: fat tails, absence of autocorrelation and volatility clustering. These stylised facts will be used to evaluate the representativeness of the generated series by the phase transition model (section 4.2).

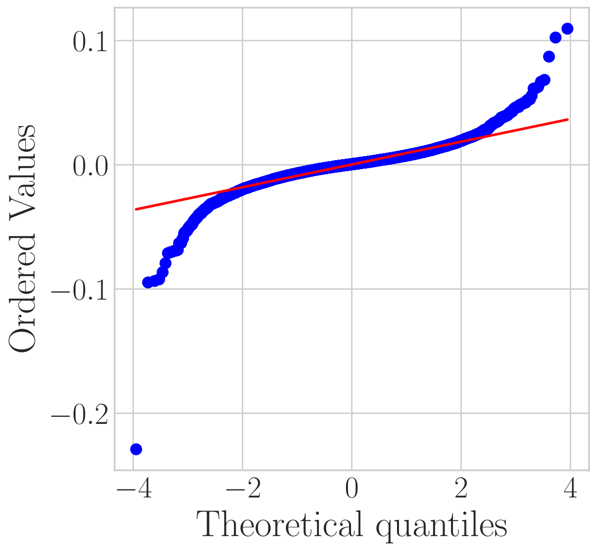
4.1.4 Quantifying the stylised facts

4.1.4.1 Fat Tails

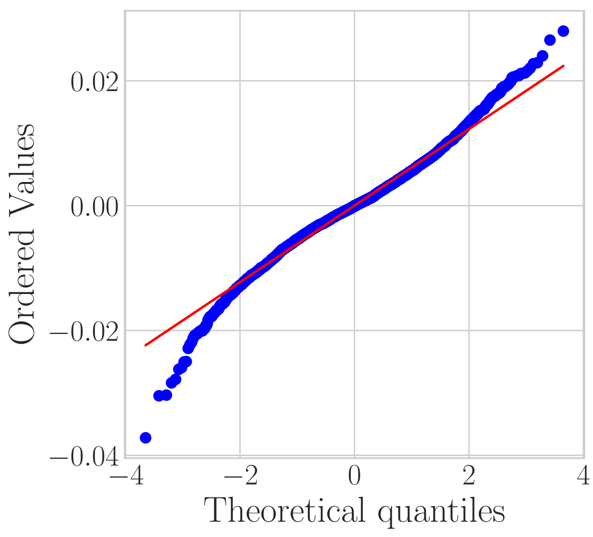
A random variable is said to have a fat tail when it exhibits more extreme outcomes than a normally distributed random variable with the same mean and variance. Mandelbrot (1963) was the first to find fat tails in financial time series. One can see in figure 4.5 that the probability of large outcomes is higher than described by the normal distribution. Cont (2001a) proposes Kurtosis as a measure for fat tails. Kurtosis measures the degree of peakedness of a distribution relative to the tail and is a strong signal that a return series has a fat tail. Table 4.2 shows the calculated Kurtosis for different indices and currency pairs using the following formula

$$\kappa = \frac{\langle (r(t,T) - \langle r(t,T) \rangle)^4 \rangle}{\sigma^2} \tag{4.3}$$

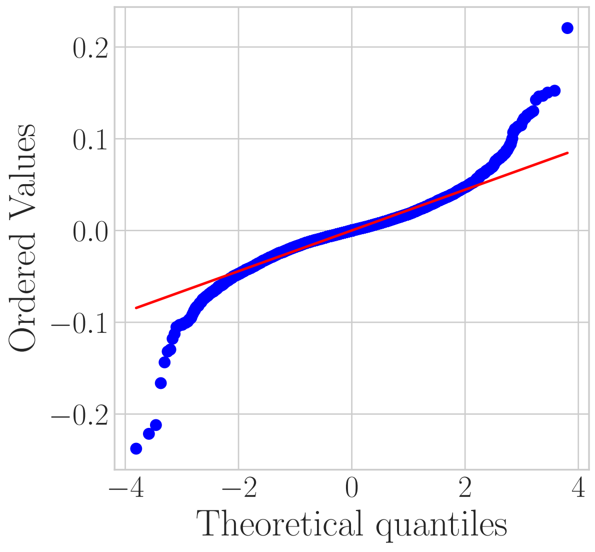
where σ^2 is the variance of the log returns r(t,T) = x(t+T) - x(t). Kurtosis is nevertheless not a robust metric of fat tailedness as a fat tail and high kurtosis do not go necessarily hand in hand. Multiple counter examples can be found in Balanda and Macgillivray (1988). Further, the standard error is enormous as the definition of the kurtosis consists of the fourth moment. This makes the Kurtosis an unworkable measure for fat tailedness.



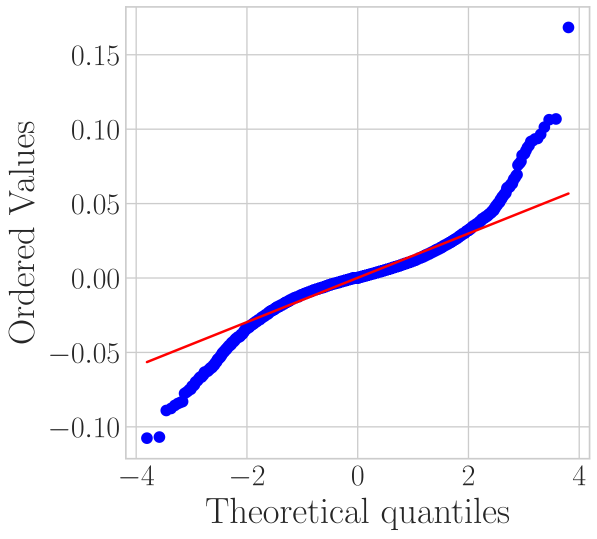
(a) S&P 500



(b) EUR USD



(c) LLoC1



(d) US10Y

Figure 4.5: QQ Plot for daily returns

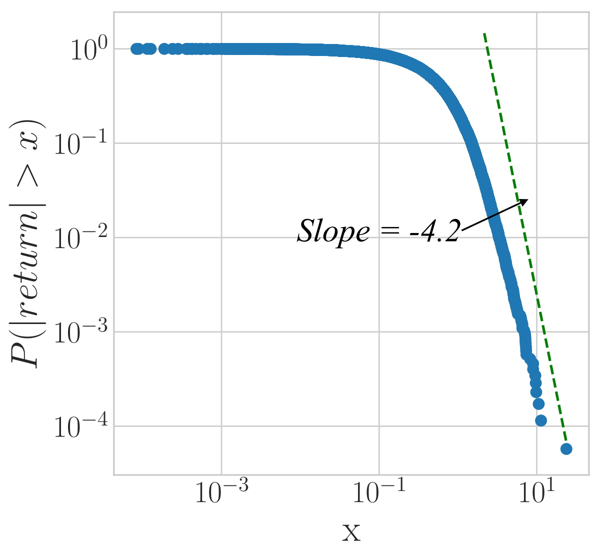
Asset	Kurtosis
S&P 500	27.20
US10Y	6.65
EURUSD	2.94
LCOc1	10.83

Table 4.2: Kurtosis of daily returns

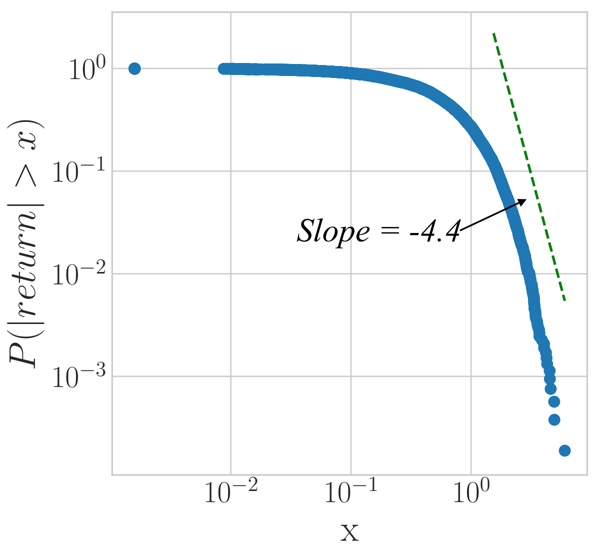
A better measure to quantify fat tailedness is to calculate the exponent of a power law as they do not exhibit the disadvantages of the Kurtosis. Mandelbrot (1963) and Fama (1965) found that stock return distributions exhibited power tails given by:

$$Pr(|r| > x) = Bx^{-\alpha} \tag{4.4}$$

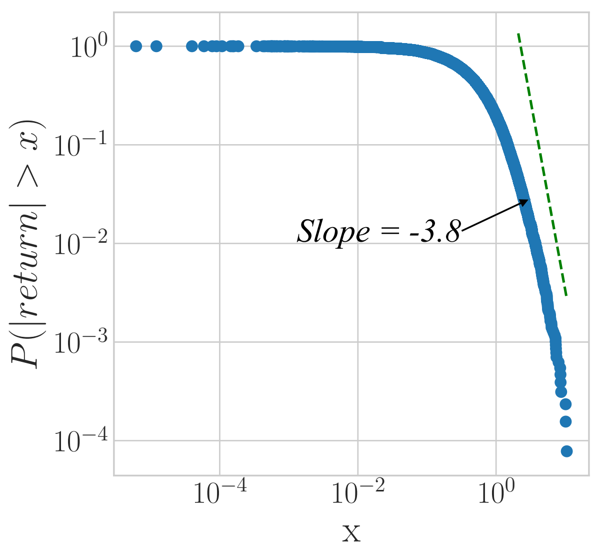
With r being the demeaned return of the asset, B is the scaling coefficient and α the tail power coefficient, also referred to as the tail index. This can be illustrated by plotting $Pr(|r| > x) = Bx^{-\alpha}$ versus x on a double logarithmic plot. When the returns obey a power law the plot will appear as a straight line for large x, with the slope of the line being an estimate for the tail index α. The power laws were fitted on the absolute log returns using a python package developed by Alstott et al. (2014) and based on the theory of Clauset et al. (2009). The power law is plotted on figure 4.6 [table A.1] together with the daily returns distribution. The calculated tail indices are in line with what found in the literature (Warusawitharana, 2018).



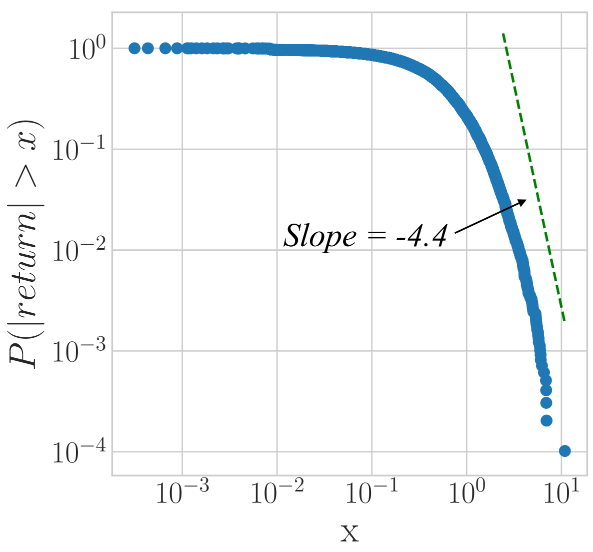
(a) S&P 500



(b) EUR USD



(c) LLoC1



(d) US10Y

Figure 4.6: Tail distribution of stock returns

4.1.4.2 Autcorrelation

Fama (1965) discovered that returns do not exhibit any significant autocorrelation. This conceptually means that a positive return today does not imply a positive return tomorrow. The absence of correlation is intuitively easy to understand: if returns exhibit significant correlation, this correlation may be used to conceive a simple strategy with positive expected earnings. Such strategies, termed statistical arbitrage, will therefore tend to reduce correlations except for very short time scales, which represent the time the market takes to react to new information Cont (2001a). Autocorrelation is calculated as follows:

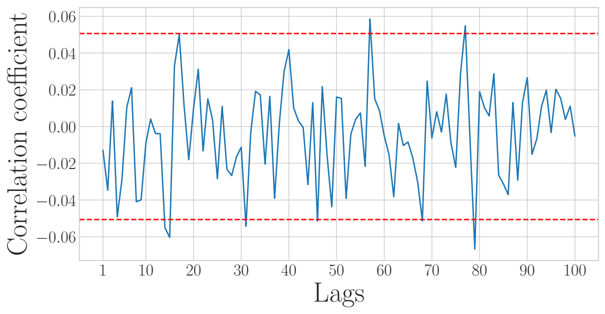
$$C_\tau = corr(r(t,t), r(t+\tau,t)) \tag{4.5}$$

with τ being the amount of lags. Figure 4.7 plots the autocorrelation of standardised daily returns up to 100 lags, along with a 95% confidence interval. The confidence interval is calculated according to the method of Olver et al. (2010) as $\frac{1.96}{\sqrt{N}}$, with N being the sample size. Visually, one can see that on most assets, the autocorrelation stays within the confidence interval. Aditionally, a more analytical method is used to supplement these results in order to decide the amount of autocorrelation present in a time series.

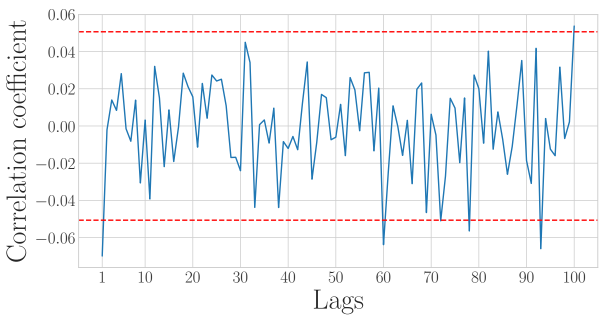
The Ljung-Box test (Ljung and Box, 1978) can be used to statistically test for autocorrelation in the returns. It is a Portmanteau statistical test for the H_0: the data are independently distributed where the observed correlation results from randomness. Against H_1: the data are not independently distributed; they exhibit serial correlation. The test statistic is given by:

$$LB(m) = n(n+2)\sum_{k=1}^{m} \frac{\hat{C}_\tau^2}{n-k} \tag{4.6}$$

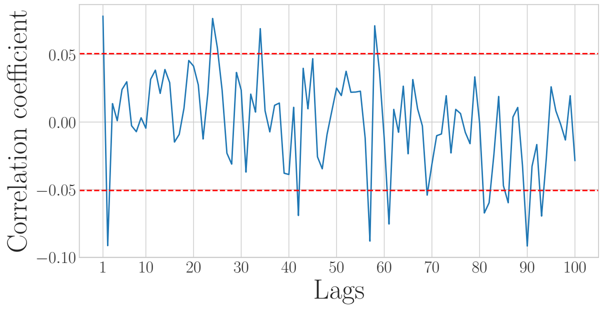
with n the sample size (or the number of trading days), $\hat{C}_\tau^2$ is the autocorrelation at lag k, and m is the number of lags that are tested. The principle is that H_0 is rejected if the p-value is equal to or less than the significance level. The absence of autocorrelation does not always appear to hold at larger time lags, as a week or a month. This is due to large estimation errors in bigger lags and error accumulation issues in summation. It is therefore wise to use a small m, as serial correlation at small lags is often at its strongest. In addition, the Ljung-Box test is for large nor small T samples robust (Fan and Yao, 2017). To counteract these problems, 100 random samples of 500 timesteps of the standardized returns are taken from each time series. The Ljung-Box test is applied on each sample with m = 1, as autocorrelation is most likely to occur at low lags, at a 5 % significance level. Table 4.3 [table A.2] outlines the percentage of cases in which the test proves absence of autocorrelation. One can see that most assets exhibit no autocorrelation on a one day lag. The S&P 500 displays autocorrelation in 38 % of the samples. The reason for that acccording to Jim Simons of Renaissance technology is that autocorrelation used to be present until the 1980s, from then it got arbitraged out (Weatherall, 2013).



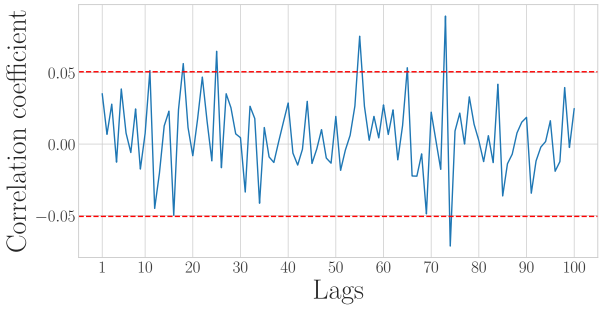
(a) S&P 500



(b) EUR USD



(c) LLoC1



(d) US10Y

Figure 4.7: Autocorrelation plots of daily returns, along with a 95 % confidence interval

4.1.4.3 Volatility Clustering

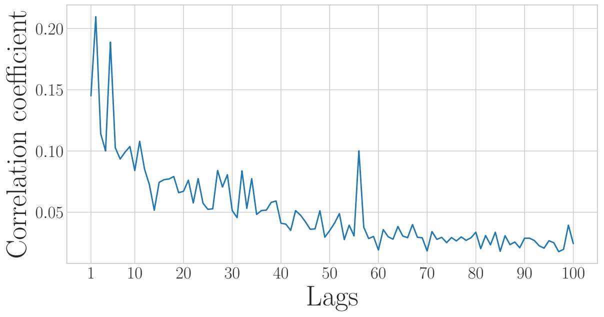
Volatility clustering discovered by Mandelbrot (1963), implies that large price variations are more likely to be followed by large price variations while small price variations are more likely to be followed by small price variations. The main driver behind volatility clustering is leverage of investment firms (Thurner et al., 2012). Bouchaud et al. (2001) and Taylor (2011) nevertheless claim that volatility clustering is partly caused by herding behaviour. As herding behaviour, is according to Sornette, Woodard, et al. (2013a) the main driver for crashes, this is an extremely important stylised fact in the context of our research.

The quantitative demonstration of this fact is that the squared returns r_t^2 show a positive, significant and slowly declining autocorrelation function given by:

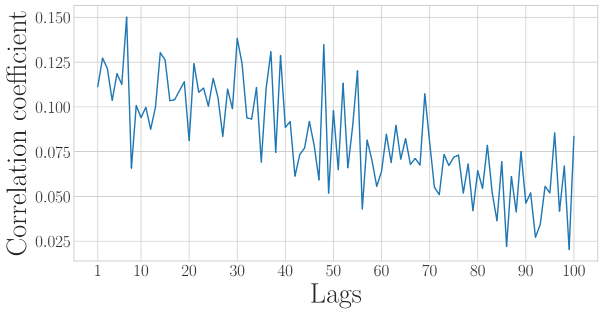
$$C_2(\tau) = Corr(|r(t+\tau, \Delta t)|^2, |r(t, \Delta t)|^2) \tag{4.7}$$

This behaviour can visually be observed in figure 4.8. Observations of this type in financial time series go against simple random walk models and have led to the use of GARCH models and mean-reverting stochastic volatility models in financial forecasting and derivatives pricing.

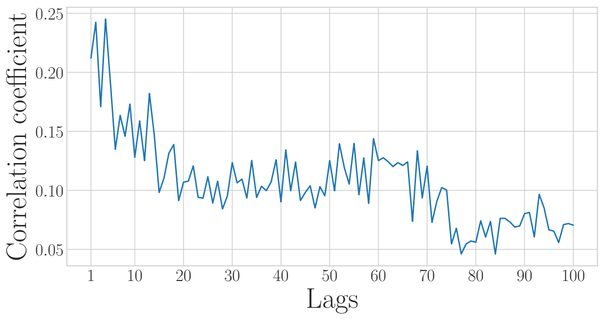
This phenomenon is analytically tested with a Ljung-Box test (equation 4.6), but this time we will test for the presence of correlation on the squared returns instead of testing for the absence of correlation on the returns. Again, 100 samples of 500 trading days are taken. The Ljung-Box test is applied on each sample with m = 21, as volatility clustering should still be present on a longer time frame, at a 5 % significance level. Table 4.3 [table A.2] outlines the percentage of case where autocorelation can be observed on the squared returns, which implies volatility clustering. Most volatility clustering can be observed for the S&P 500, while the currency pair EUR-USD and the 10 Year Treasury only exhibit volatility clustering in 50 % of the cases.



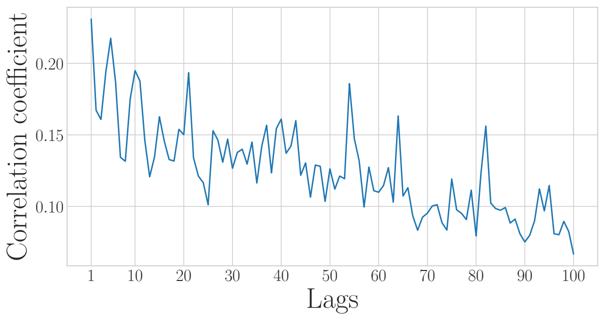
(a) S&P 500



(b) EUR USD



(c) LLoC1



(d) US10Y

Figure 4.8: Autocorrelation of the squared returns,along with a 95% confidence interval, for the first 100 lags

Assets	Absence of autocorrelation	Presence of volatility clustering
S&P 500	59 %	83 %
US10Y	79 %	54 %
EURUSD	91 %	47 %
LCOc1	75 %	67 %

Table 4.3: Fraction of the time series with absence of autocorrelation and presence of volatility clustering [Real financial data]

4.1.5 Summary of section

This section was devoted to deriving the stylised facts of financial time series. First, representative assets were selected that are relevant for deriving the characteristics. This work ended up with 12 assets that are outlined in table 4.1. Secondly, three stylised facts were selected and quantified. The finding is that: financial assets exhibit power law behaviour on their absolute returns, with a tail index between 3 and 4.5. The returns do not display autocorrelation in 60 % to 90 % of the cases and volatility clustering is present between 50 % and 80 % of the cases. This means that in the next section a configuration of a phase transition model needs to be found that realizes at generating series that display those stylized facts.

4.2 Phase transition model that generates series exhibiting stylised facts

This section searches for a setup of a phase transition model which arrives at creating series that express the stylised facts discussed in the previous section. In order to achieve this, two successive steps must be taken. First, we must try to find the most appropriate phase transition model for our cause (section 4.2.1). Secondly, the parameters both inherent and external to the model will be tweaked to achieve the highest level of similarity between the synthetic data and the real financial data (section 4.2.3).

4.2.1 Choice of phase transition model

Three models are considered as phase transition model. The first spin model used for financial markets was introduced by Bornholdt (2001) and is based on the Ising model. In the Bornholdt model the spin configuration represent 'buy' or 'sell'. Those two conflicting interactions cause a complicated dynamics, i.e. there is no single stable phase, rather the model exhibits metastable phases. This results in the fat-tailed distribution of the price returns and clustering of the volatility. Properties that are also displayed in financial time series (Cont, 2001a). This model is explained by Zhou and Sornette (2007b) as the basis of the LPPL.

The second spin model considered in this work is the Potts model (Potts, 1952). The Potts model has more complexity than the Ising model as it can span more than two states, allowing the model to encompass first-order and second-order phase transitions while the Ising model can only model second-order phase transitions. This means that, financially speaking, the q-state Potts model can represent,'buy', 'hold' and 'sell' whereas the Ising model can only represent 'buy' and 'sell'.

The last examined spin model is the Kuramoto model (Kuramoto, 1975). The primary benefit of the Kuramoto model is that time is a true model-dependent parameter, whereas for the Potts and Ising model time is an arbitrarily selected Monte Carlo update step. Secondly, each cell has its own weight, which is a better economic representation of reality. For instance, the trade advice of Warren Buffet is given more importance than the trade advice of the writer.

This work has chosen to continue its analysis with the Potts model since it offers more space for complexity and flexibility than the Ising model. The Kuratmoto model approaches reality closer, but it transcends this introductory work in terms of sophistication. What follows is a more in-depth discussion of the q-states Potts model.

4.2.2 The Potts model

4.2.2.1 Physical description

The q-state Potts model can be considered as a system of spins confined in a lattice, which each spin pointing to one of the q equally spaced directions. Consider a two-dimensional integer lattice $G = (V, E)$ with vertex set V and edge set E. Vertici i, j are considered nearest neighbours if they have a common edge $e = (i, j) \in E$. Each vertex V is initially assigned with a randomly chosen value θ_n which takes one of q possible values, distributed uniformly about the circle, at angles:

$$\theta_n = \frac{2n\pi}{q} \tag{4.8}$$

where n = 0,1,...,q-1. A pair of nearest neighbours gets energy $-J$ if they have an identical spin. This specific model configuration is known as the vector Potts model or the clock model. *The Hamiltonian* (dimensionless energy functional) of the Potts model is thus:

$$H = J \sum_{(i,j)\in E} \cos(\theta_{si}, \theta_{sj}) \tag{4.9}$$

with the Kronecker delta function being:

$$\delta(\sigma_i, \sigma_j) = \begin{cases} 1 & \text{if} s_i = s_j \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

Then the *partition function* of the q-state Potts model is:

$$Z = \sum e^{-\beta H} \tag{4.11}$$

where $\beta = \frac{1}{k_b T}$, being the inverse temperature. The ferromagnetic context $J > 0$, the spins tend to align at low temperature($J >> 1$), describing a phase of ferromagnetic order. By contrast, the spins are almost independent at high temperatures ($J << 1$), leading to a paramagnetic phase in which entropic effects prevail. On a physical basis, it is anticipated that the two phases will be divided by a critical value β_c. For q = 2, the clock model relates to the familiar Ising model with $J_i = 1/2J_p$. For q > 4, the model has more entropy with an accompanied first-order transition. The magnetisation M of the system is given by:

$$M(t) = \frac{1}{N} \sum_{i}^{b} s_i \tag{4.12}$$

4.2.2.2 Financial agent model

A financial-agent based model is developed upon the q-state Potts model on a NxN grid. The assumption is made that the stock market consists out of one single stock or commodity traded by N^2 agents of equal size. The traders are located on a 2D NXN square grid and can trade at each time t. It is important to note that time is not an inherent parameter of the physical model. Time is defined here as an arbitrarily chosen Monte Carlo update step. We believe that the q-state Potts model can imitate the interaction between financial agents in the market adequately. The price resulting from the interaction of agents is defined as the magnetisation M(equation 4.12).

4.2.2.3 Monte carlo update

The algorithm for updating the state of the agents is based on the hit-and-miss method outlined by Enric (2017) and is as following:

1. Generate a random configuration of the N^2 spins which each spin pointing to one of the q equally spaced directions.
2. Loop over each agent
   (a) Calculate the Hamiltonian H_{old} of the agent from equation 4.9
   (b) Replace the agent with a random state and calculate the new Hamilitonian H_{new}
   (c) Generate a uniform random number r in [0,1]
   (d) If $r \leq exp(\beta(H_{old} - H_{new}))$ update the agent with the new state
3. Calculate the magnetism/price of the whole system from equation 4.12 . From now on we will refer to the magnetisation as price.
4. Go back to step 1 unless the required timeframe t is reached

5. Cut off the burn-in period b from the generated timeserie which acts as thermalisation

4.2.3 Choosing the best Potts model configuration

This section is about searching the Potts model configuration, which produces series that exhibit comparable stylized facts as discussed in section 4.1. This work will tweak three parameters that are inherent to the model and two parameters that are external to the model. The parameters inherent to the Potts model are: (i) the number states q (ii) the grid size N (iii) the inverse temperature or beta. Beta is defined as how closely the system operates towards β_c. Therefore, for each Potts configuration the critical β should be found. The algorithm to do so is outlined in box 1. The parameters external to the Potts model are the series length T, the burn-in period b, and the sub sampling step p.

	Parameters	Value
Inherent	Amount of states q	2
	Grid size $N \times N$	100×100
	Ratio $\frac{\beta}{\beta_c}$	1
External	Series length T	10 000
	Thermalisation/burn-in period	2 000
	Subsample step	1

Table 4.4: Benchmark Potts model configuration

For the parameter search, a benchmark configuration as outlined in table 4.4 is utilized. More specifically, the benchmark is the Potts implementation of the configuration outlined by Zhou and Sornette (2007a). In order to find the optimal parameter configuration, each parameter will be tweaked, while keeping the other parameters constant. Further research could fine tune configuration by analyzing the interactions between the different parameters. For each parameter search the following is done:

1. For each configuration 100 sequences are generated, with each sequence representing price points

2. The prices of each sequence are transformed to log standardised returns as outlined in section 4.1.2.

3. The configuration is assessed to show stylized facts in line with the ones exhibited by real financial time series, outlined in section 4.1.5. This is evaluated analytically as follows:

    (a) **Fat tails**: For each of the 100 series the tail index is calculated according to the method of Clauset et al. (2009). Then for each configuration the mean is calculated, along with a 95 % confidence interval [1]. The aim is to find a tail index that is within the range exhibited by real financial time series, outlined in section 4.1.5.

    (b) **Autocorrelation**: Similar as in section 4.1.4.2, for each serie 100 samples of 500 time steps are selected. For each sample the Ljung-Box test for correlation(equation 4.6) for m =1 is performed on the returns. Then, the percentage where autocorrelation is absent on a 5 % significance level is calculated.

    (c) **Volatility clustering**: Again, for each sequence 100 samples of 500 time steps are selected. On each sample the Ljung-Box test for correlation is performed on the squared returns with m = 21. Then, the percentage where autocorrelation (volatility clustering) is present on a 5 % significance level is calculated.

4. The value of the parameter that generates sequences most similar to real financial time series is selected

```
1. Loop over different possible $\beta_c$ values
    (a) Loop 10 times
        i. Update the grid 4000 times, and calculate the Energy of the system
        ii. Calculate the median of the Energy of the last 1000 observations
    (b) Take the average of the 10 medians
2. Plot the median Energies in function of $\beta$
3. Derive the median Energy
4. $\beta_c$ is the $\beta$ for which the derivation of the energy is the highest
```

: Algorithm for determining the β_c

[1]In the literature there is discussion between the Bayesian statistici and Frequentist school whether the mean can be pinpointed

The coming subsections are devoted to the optimal parameter configuration. To keep the corpse of the document somewhat dense, the figures will only be plotted for the first paragraph.

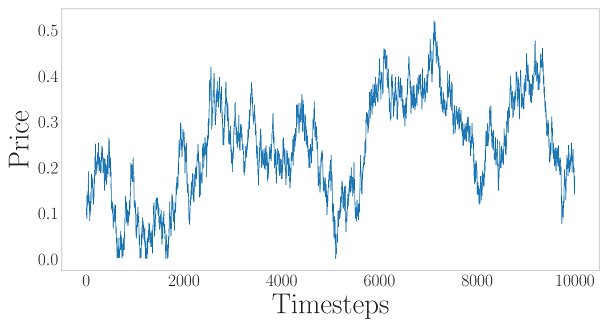
4.2.3.1 Parameters inherent to the Potts model

4.2.3.1.1 Number of states q This work has decided to examine the model's performance for the states 2,3,4,5. This choice is justified by the fact that a 2,3 or 4 state Potts model displays a second-order phase transition, while a 5-state Potts model displays a first-order phase transition (Arisue and Tabata, 2001). This allows us to capture the whole spectrum of phase transitions and the possible nuances that accompany them. The tested configuration is outlined in table 4.5.

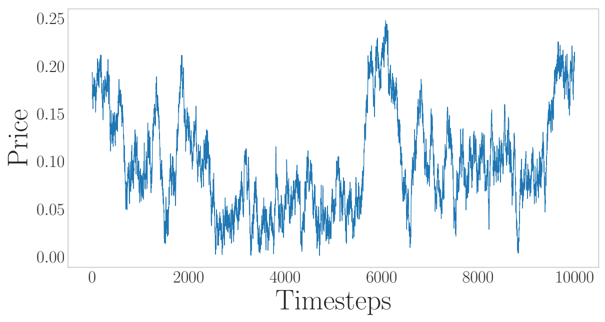
	Parameters	Value
Inherent	**Amount of states q**	**2 3 4 5**
	Grid size $N \times N$	100×100
	Ratio $\frac{\beta}{\beta_c}$	1
External	Series length T	10 000
	Thermalisation/burn-in period	2 000
	Subsample step	1

Table 4.5: Configuration set-up to find the best amount of states q

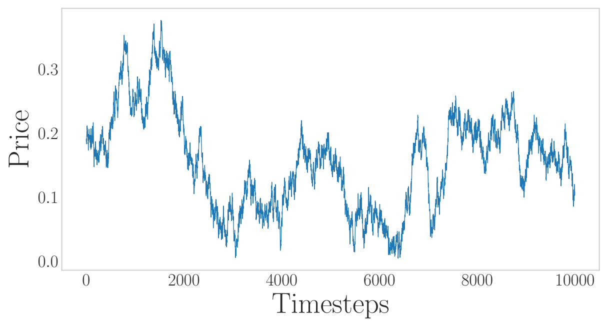
Figure 4.9 plots the price in function of the time steps for the different configurations outlined in table 4.5. Visually, the similarity with financial time series can be noted. Moreover, it is difficult to distinguish between the four configurations on sight.



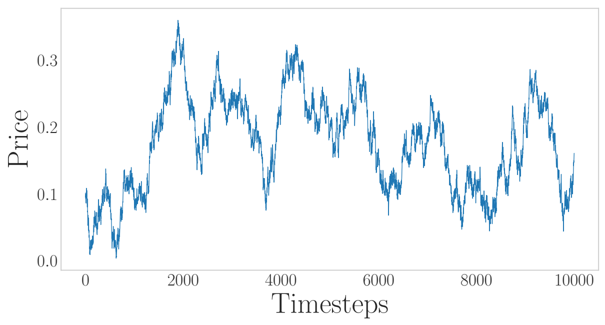
(a) 2 states Potts model



(b) 3 states Potts model



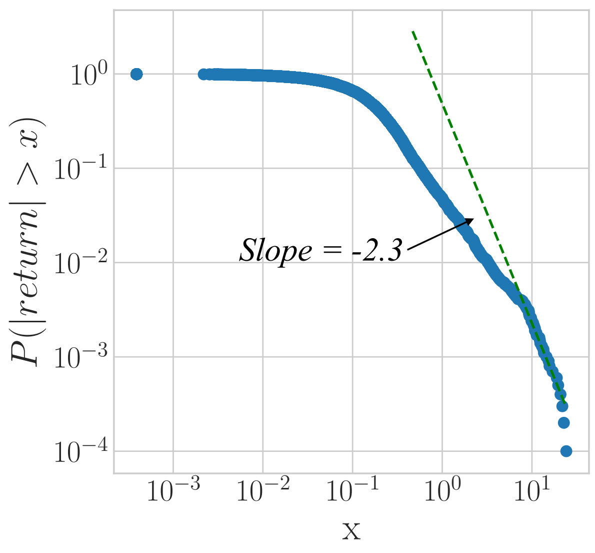
(c) 4 states Potts model



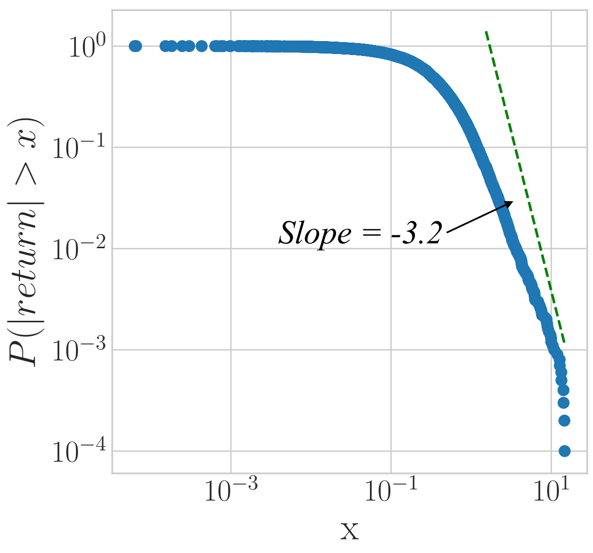
(d) 5 states Potts model

Figure 4.9: Price evolution for the different configuration of the Potts model outlined in table 4.5

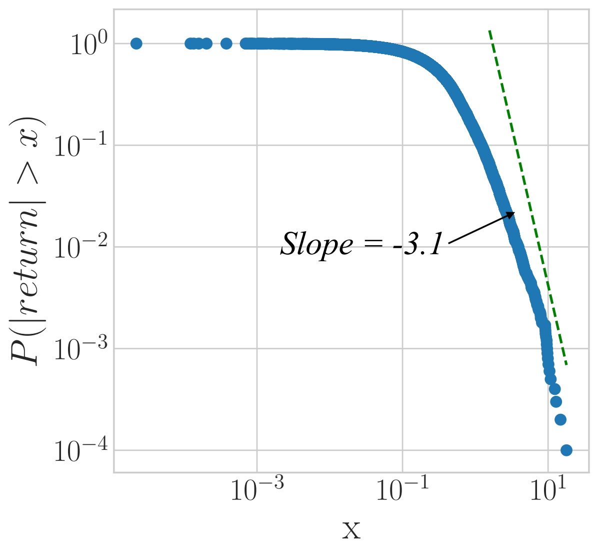
We will now evaluate the stylised facts of each configuration on 100 samples. As outlined in 4.1.5, we need to aim to find a configuration that generates series with a tail index between 3 and 4.5, absence of autocorrelation in 60 % to 90 % of the cases and presence of volatility clustering in 50 % to 80 % of the cases. The stylised facts will be derived visually for one sample of each configuration to reinforce our conclusion. To keep the corpse somewhat dense this will only be done for the amount of states q.



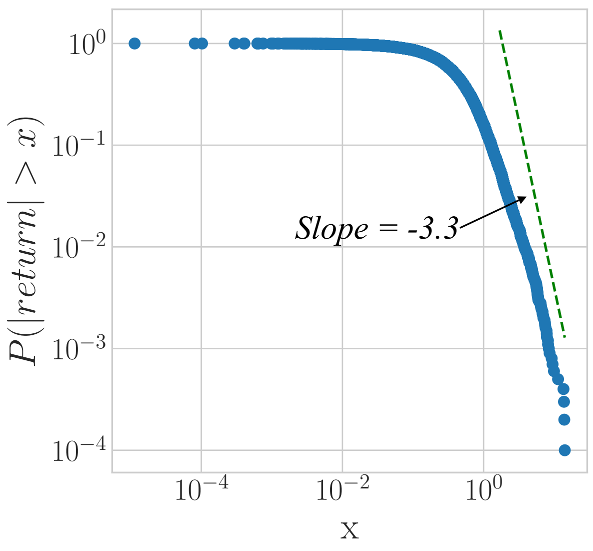
(a) 2 states Potts model



(b) 3 states Potts model



(c) 4 states Potts model



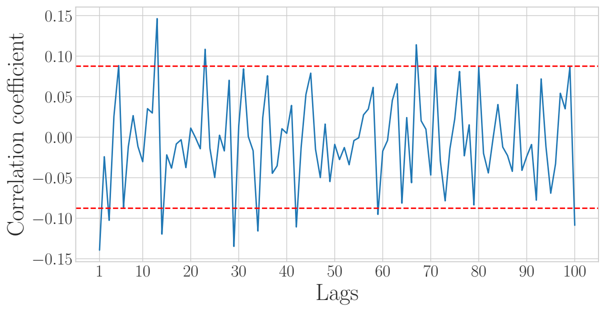
(d) 5 states Potts model

Figure 4.10: Tail index for different configuration of the Potts model outlined in table 4.5

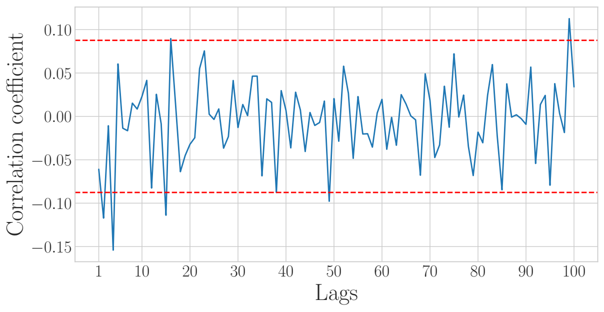
The slope of the fitted power law on the absolute returns is shown in figure 4, which demonstrates that the slope seems to increase with more states. Table 4.6 quantifies the mean of the tail index, along with a 95 % confidence interval. It partially confirms the conclusion drawn from the plot, the tail index increases for a q up to 4 and then appears to decrease again for q=5. A feasible explanation could be the shift from second order phase transition to first order phase transition. The configuration for q = 4 seems to correspond best with the range of power laws exhibited by real financial time series.

This can be interpreted as possible evidence that the Ising model is not the best phase transition model to propose as foundation for the LPPL.

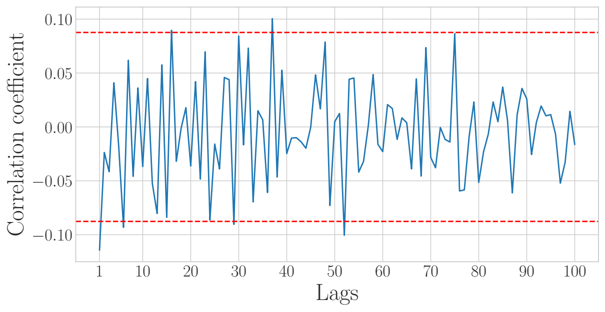
The ACF of the returns is shown in figure 4.11 for the different configuration, along with a 95 % confidence interval calculated as described in section 4.1.4.2. Table 4.7 shows that the absence of autocorrelation seems to decrease as volatility clustering increases. It can be concluded that the 4-5 states Potts model is within the pre-set range, whereas the 2-3 state model Potts is not.



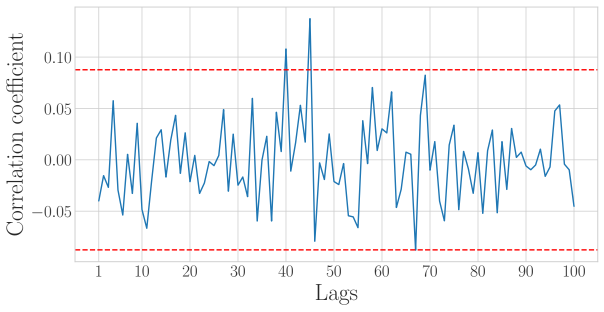
(a) 2 states Potts model



(b) 3 states Potts model



(c) 4 states Potts model



(d) 5 states Potts model

Figure 4.11: ACF of the returns for different configuration of the Potts model outlined in table 4.5

Figure 4.12 shows the ACF of the squared return. From table 4.7 it can be concluded that 4-5 states Potts model is within the pre-set range for volatility clustering, whereas the 2-3 state model Potts is not.

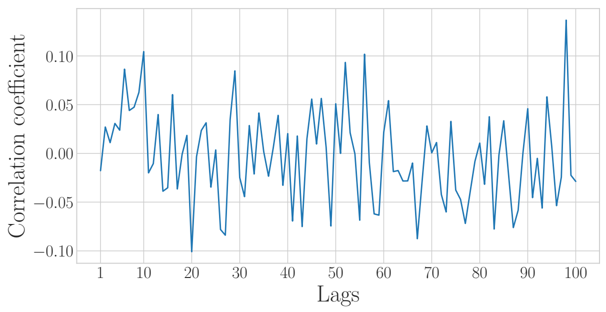
This work chooses for the 4-state Potts model as for q = 4 it seems to correspond best with the range of power laws, and shows sufficient absence of autocorrelation and presence of volatility clustering.

States	Confidence Interval
q = 2	2.06 ± 0.02
q = 3	3.13 ± 0.08
q = 4	3.48 ± 0.21
q = 5	3.09 ± 0.10

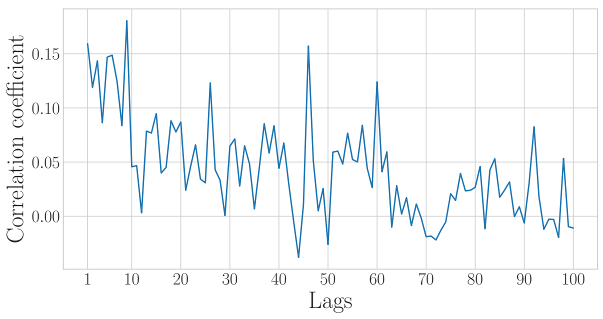
Table 4.6: The average tail index for different ratios, along with a 95 % confidence interval [States q]

States	No autocorrelation	Volatility clustering
q = 2	51.41 %	82.11 %
q = 3	54.15 %	80.59 %
q = 4	71.87 %	61.64 %
q = 5	79.65 %	64.58 %

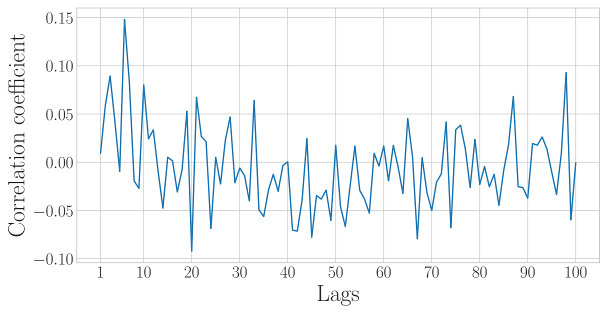
Table 4.7: Fraction of the time series with absence of autocorrelation and presence of volatility clustering [States q]



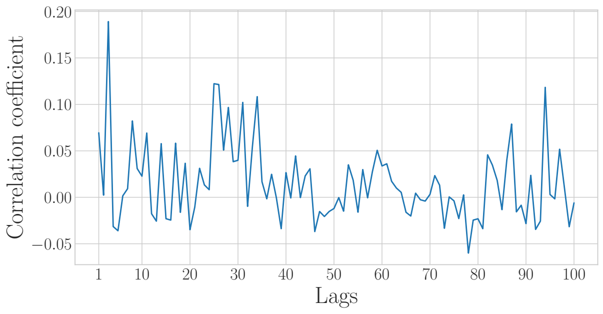
(a) 2 states Potts model



(b) 3 states Potts model



(c) 4 states Potts model



(d) 5 states Potts model

Figure 4.12: ACF of the squared returns for different configuration of the Potts model outlined in table 4.5 [States q]

	Parameters	Value
Inherent	Amount of states q	2
	Grid size NxN	**50x50 100x100 200x200 500x500**
	Ratio $\frac{\beta}{\beta_c}$	1
External	Series length T	10 000
	Thermalisation/burn-in period	2 000
	Subsample step	1

Table 4.8: Configuration set-up to find the best grid size NxN

4.2.3.1.2 Grid size N This subsection will try to find the optimal balance between computational effort and finite size effects, with the aim of finding the best stylised facts for the lowest grid size. Table 4.9 shows that the tail index increases gradually, for an increasing grid size. However, this trend is accompanied by an increasing standard deviation, which is not always desirable. From table 4.10 we see that a 50x50 grid shows a low absence of autocorrelation, whereas the other grid sizes are showing an acceptable absence of autocorrelation and presence of volatiltiy clustering. We will therefore proceed with a 100x100 gird, which is the lowest grid size that provides sensible stylised facts.

N	Confidence Interval
50x50	2.09 ± 0.06
100x100	2.13 ± 0.08
200x200	2.36 ± 0.19
500x500	2.51 ± 0.39

Table 4.9: The average tail index for different ratios, along with a 95 % confidence interval [Grid size]

N	No autocorrelation	Volatility clustering
50x50	37.51 %	95 %
100x100	53.42 %	82.80 %
200x200	59.11 %	74.04 %
500x500	56.89 %	74.78 %

Table 4.10: Fraction of the time series with absence of autocorrelation and presence of volatility clustering [Grid size]

	Parameters	Value
Inherent	Amount of states q	2
	Grid size $N \times N$	100×100
	Ratio $\frac{\beta}{\beta_c}$	**0.95 0.98 0.99 1 1.01 1.02 1.05**
External	Series length T	10 000
	Thermalisation/burn-in period	2 000
	Subsample step	1

Table 4.11: Configuration set-up to find the best ratio $\frac{\beta}{\beta_c}$

4.2.3.1.3 Critical Beta Two main trends can be observed: table 4.15 indicates that the tail index appears to be U-shaped in relation to the ratio. Secondly, in table 4.16, the absence of autocorrelation increases with an increasing ratio, while the presence of volatility clustering decreases with an increasing ratio. Only the configurations with a ratio of 1.01 and 1.02 display stylised facts which fall within the pre-set range. As it is not clear which ratio is superior and how they will interact with the other parameters, a further analysis will be performed in the next section.

Grid size	Confidence Interval
$0.95 \cdot \beta_c$	2.46 ± 0.24
$0.98 \cdot \beta_c$	2.16 ± 0.10
$0.99 \cdot \beta_c$	2.09 ± 0.07
$1 \cdot \beta_c$	2.17 ± 0.09
$1.01 \cdot \beta_c$	2.57 ± 0.48
$1.02 \cdot \beta_c$	3.84 ± 0.80
$1.05 \cdot \beta_c$	4.53 ± 1.05

Table 4.12: The average tail index for different ratios, along with a 95 % confidence interval $[\frac{\beta}{\beta_c}]$

β	No autocorrelation	Volatility clustering
$0.95 \cdot \beta_c$	10.45 %	99.35 %
$0.98 \cdot \beta_c$	26.55 %	96.05 %
$0.99 \cdot \beta_c$	43.75 %	91.40 %
$1 \cdot \beta_c$	45.15 %	88.20 %
$1.01 \cdot \beta_c$	59.55 %	75.00%
$1.02 \cdot \beta_c$	77.20 %	53.75 %
$1.05 \cdot \beta_c$	76.20 %	32.65 %

Table 4.13: Fraction of the time series with absence of autocorrelation and presence of volatility clustering $[\frac{\beta}{\beta_c}]$

4.2.3.1.4 Analyzing the interaction effect of the inherent parameters This subsection combines the parameters resulting from the analysis conducted in the previous subsections. Since there are still two distinct beta values, two configurations must be compared (table 4.14). We will refer to the configuration with ratio $\frac{\beta}{\beta_c} = 1.01$ model 1 and $\frac{\beta}{\beta_c} = 1.02$ model 2.

	Parameters	Value
Inherent	Amount of states q	4
	Grid size $N \times N$	100×100
	Ratio $\frac{\beta}{\beta_c}$	1.01 1.02
External	Series length T	10 000
	Thermalisation/burn-in period	2 000
	Subsample step	1

Table 4.14: Configuration set-up to find the best inherent parameters

Both models exhibit tail indices of equal size, the first model has a slightly lower standard deviation which is preferential. However, the second model exhibits low level of volatility clustering. For those reasons the final configuration of the model will be a 100x100 4-states Potts model with a $\frac{\beta}{\beta_c} = 1.01$

Model	Confidence Interval
Model 1	4.07 ± 0.24
Model 2	3.93 ± 0.41

Table 4.15: The average tail index for different ratios, along with a 95 % confidence interval

Model	No autocorrelation	Volatility clustering
Model 1	75.3 %	50.7 %
Model 2	74.4 %	31.65 %

Table 4.16: Fraction of the time series with absence of autocorrelation and presence of volatility clustering

4.2.3.2 Parameters external to the Potts mode

After selecting parameters inherent to the Potts model that generate sequences exhibiting stylized facts most equal to those found in financial time series, we will now tweak parameters which are external to the Potts model. Three main parameters will be tweaked: the choice of the sequence length T, the burn-in period b and the subsampling step p.

4.2.3.2.1 Choice of sequence length T The ideal sequence length T is evaluated according to its stylized facts, on the one hand, and the features of its crashes, on the other. The tail index is too large for a T smaller than 5 000 and too small for a sequence length greater than 50 000 as shown in table 4.17. Because autocorrelation and volatility clustering do not differ significantly, a sequence length of 10 000 appears to be the best option from a stylized facts point of view.

T	Confidence Interval
2500	4.68 ± 0.47
5000	4.64 ± 0.56
10000	3.98 ± 0.43
20000	3.64 ± 0.37
50000	3.10 ± 0.11

Table 4.17: The average tail index for different ratios, along with a 95 % confidence interval [T]

T	No autocorrelation	Volatility clustering
2500	75.2 %	57.3 %
5000	77.0 %	47.15 %
10000	73.9 %	48.75 %
20000	73.8 %	50.9 %
50000	74.3 %	47.93 %

Table 4.18: Fraction of the time series with absence of autocorrelation and presence of volatility clustering [T]

Secondly, we try to achieve features of crashes that resemble those of real financial time series as close as possible. For this test, we define a crash as the 1 % heaviest drawdowns, with a drawdown being consecutive losses ignoring half a standard deviation increase. More information on the definition of drawdowns can be found in section 5.2.1. Three main features will be compared (i) the average drawdown of the crash (ii) the average duration of the drawdowns in trading days (iii) the average duration between two crashes. Those features are quantified for financial assets in table 4.19 [table A.3 in appendix], we notice that the average drawdown ranges from -9 to -16 standard deviations from the mean, the average duration from 6 to 7 trading days and a crash occurs every 480 to 650 days.

T	Avg. drawdown	Avg. duration	Frequency
2500	-8.92	6.98	523
5000	-9.52	6.39	531
10000	-9.78	6.11	545
20000	-10.1	5.06	568
50000	-10.7	4.27	580

Table 4.20: Crash features for various values of T

Asset	Avg. drawdown	Avg. duration	Frequency
S&P 500	-13.28	6.59	545
US10Y	- 11.8	6.3	515
EURUSD	-9.3	7	480
LCOc1	-9.7	7	520

Table 4.19: Crash features of various assets

Table 4.20 shows those features for different values of T, one can remark that the average drawdown of T=2500 does not fall in the range stated in the previous paragraph. Crashes seem to occur in the synthetic data as frequent as crashes in financial series for all values of T. T = 5000, 10000 are the only values that arrive at having average durations within the range found in financial series. Taking the stylized facts and the features of crashes into account, it seems us most appropriate to continue our analysis wit a T = 10 000.

4.2.3.2.2 Burn-in period It is clear out of table 4.21 and table 4.22 that the stylized facts do not differ significantly. However, to play it safe we will still opt for a burn-in period of 1000.

Burn-in	Confidence Interval
500	3.72 ± 0.33
1000	3.57 ± 0.35
2000	3.67 ± 0.33
4000	3.98 ± 0.41
8000	3.89 ± 0.34

Table 4.21: The average tail index for different ratios, along with a 95 % confidence interval [Burn-in]

Burn-in	No autocorrelation	Volatility clustering
500	75.9 %	49.7 %
1000	73.0 %	51.6 %
2000	75.3 %	55.7 %
4000	76.6 %	54.7 %
8000	74.3 %	52.4 %

Table 4.22: Fraction of the time series with absence of autocorrelation and presence of volatility clustering [Burn-in]

p	Average 0.01 % drawdown	Average duration	Days in between
1	-9.3	6.0	552
2	-8.7	5.8	545
4	-8.3	6.1	543
8	-7.7	6.0	582
16	-6.9	5.7	625

Table 4.25: Fraction of the time series where respectively no autocorrelation, volatility clustering occurs [p]

4.2.3.2.3 Smoothing parameter The smoothing parameter p, is the parameter that tells how much the series are batched. For example if p=4, every fourth observation is taken, as a result a serie of 10 000 is then reduced to one of 2500. Table 4.23 shows no significant difference for the mean of the power law. Auto correlation is at its most optimal value for a smoothing parameter of 1, with volatility clustering remaining in the relevant range. The 1 % average drawdown is most optimal for a p=1. Taking those three facts into account, this work has decided to not smooth the series.

p	Confidence Interval
1	3.78 ± 0.24
2	3.74 ± 0.24
4	3.68 ± 0.24
8	3.61 ± 0.26
16	3.46 ± 0.22

Table 4.23: The average tail index for different ratios, along with a 95 % confidence interval [p]

p	No autocorrelation	Volatility clustering
1	75.7 %	52.3 %
2	65.8 %	68.3 %
4	53.6 %	83.3 %
8	49.6 %	93.2 %
16	35.6 %	96.3 %

Table 4.24: Fraction of the time series with absence of autocorrelation and presence of volatility clustering [p]

Chapter 5

Train machine learning model on synthetic data

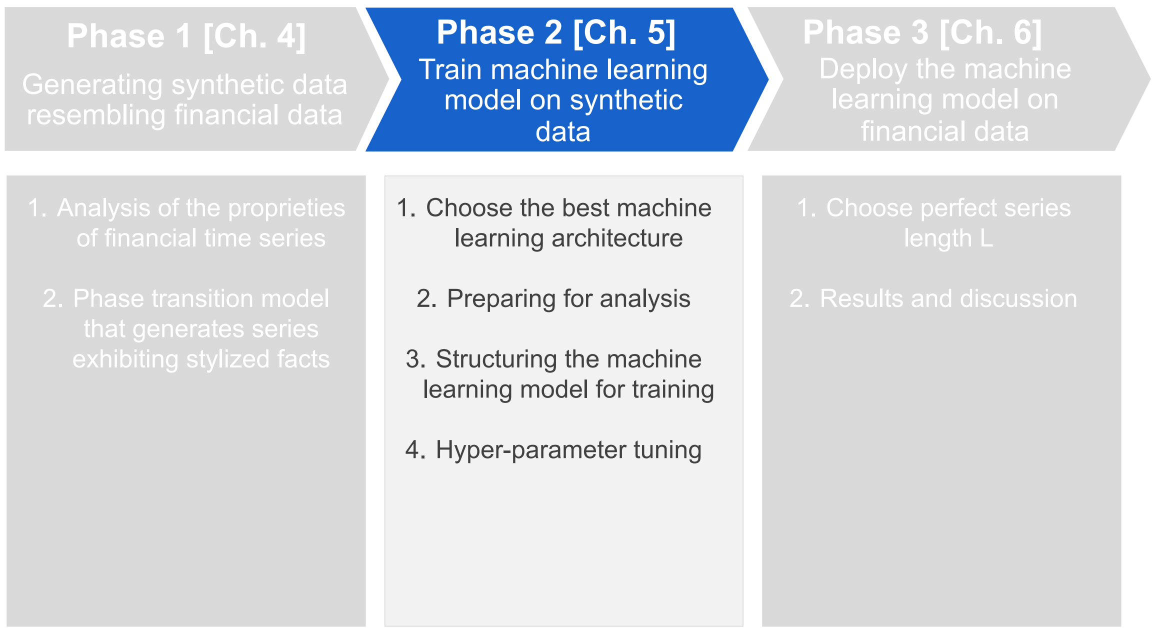

Phase 1 [Ch. 4]
Generating synthetic data resembling financial data
Phase 2 [Ch. 5]
Train machine learning model on synthetic data
Phase 3 [Ch. 6]
Deploy the machine learning model on financial data
1. Analysis of the proprieties of financial time series
2. Phase transition model that generates series exhibiting stylized facts
1. Choose the best machine learning architecture
2. Preparing for analysis
3. Structuring the machine learning model for training
4. Hyper-parameter tuning
1. Choose perfect series length L
2. Results and discussion

Figure 5.1: Phase 2: Train machine learning model on synthetic data

As the perfect phase transition model configuration was found in phase 1, the next step is to train a machine learning model on the synthetic data. This chapter consists of four sections, in the first section the machine learning architecture will be chosen. The second section is dedicated to preparing the data and the model for analysis. Thirdly, the approach and structure of training the ML model will be discussed. In the fourth section the hyper-parameters are tuned in order to achieve superior performance.

5.1 Choose best machine learning algorithm

This section is devoted to find the best machine learning algorithm that suits our problem. It constructively takes the reader to the best algorithm by explaining the different components and theories that make up the final architecture. The first subsection introduces the reader to the basic concepts of neural network. The second subsection further elaborates on it by explaining recurrent neural networks (RNNs), which are a type of neural networks capable of handling sequential data. The third subsection explains LSTM's that are a type of RNN that perform better for time series with longer dependencies. The final subsection is devoted to our final architecture choice.

5.1.1 Neural networks

McCulloch and Pitts (1943a) created a computational model, that tries to mimic the functioning of the human brain. In the 20th century, most of neural network theory was created, however, owing to an increasing data capture and stronger computing facilities, it has become increasingly popular in recent years. They have found many application in a broad spectrum of fields, due to their capacity to model nonlinear systems. This subsection is intended as introduction to the main principles of neural networks and is mainly based on the books of Goodfellow et al. (2016) and Nielsen (2017). We will start with explaining the concept of a perceptron, which is based on threshold logic and is the foundation of any neural network. Secondly, we will explain the basics of neural network architecture.

5.1.1.1 Perceptron

The idea of a perceptron (artificial neuron) was first introduced by Rosenblatt (1962) inspired by the work of McCulloch and Pitts (1943b). A perceptron has as input several binary variables x_1, x_2, x_3, which take the values of either 0 or 1 and give as output a singular binary variable. To give each input a certain importance, Rosenblatt proposed to introduce weights.

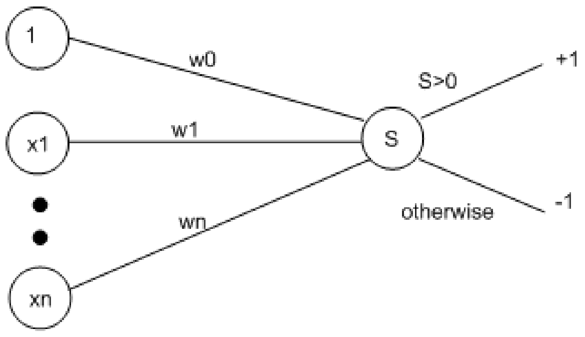


Figure 5.2: Model of perceptrons

The value of the output is dependent on the fact whether the weighted sum , $\sum_j w_j x_j$ is greater or smaller than a threshold value. The threshold value is another parameter of the model.

This concept can be applied to model everyday decision. Imagine a person who is wondering whether he should go to Benidorm for a week. The answer to that question is dependent on three other questions that each have a certain weight.

- Is the weather going to be nice? : x_1 and $w_1 = 0.5$
- Is the food good? : x_2 and $w_2 = 0.5$
- Are there mainly people of my age? : x_3 and $w_3 = 0.5$

The person will decide to go to Benidorm if the sum of the weighted variables is greater than the threshold value 1. Imagine that the outcome of the questions are all positive. The weighted sum is then 1.5, which means that the person will decide to spend the week in Benidorm. Deep learning uses this exact model as basis with the only difference that it tweaks it weights and biases of each perceptron itself without any intervention of a programmer. This results in an algorithm that can solve complex problems, without the need of someone to lay out a theory.

5.1.1.2 Architecture of a neural network

Neural networks are composed of three main components. The input layer consists of the neurons that are fed by the data. The output layer gives the final output of the entire system. The neurons that are neither output nor input are called hidden layers. The construction of the output and input layers are straightforward as they are dependent on the model. The choice of the hidden layers is not that trivial. Researchers have developed many heuristics that allow to play with the trade-off between the number of hidden layers against the time to train the network.

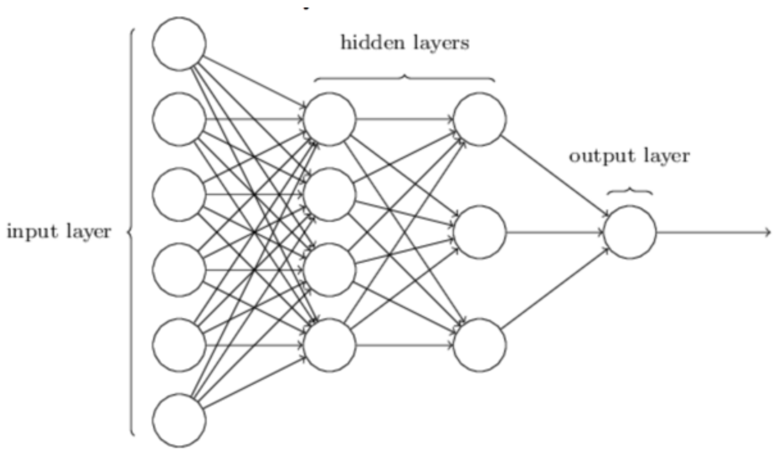


Figure 5.3: Architecture of a neural network (Nielsen, 2017)

As mentioned earlier the deep learning algorithm consists of adapting the biases and weights to improve the accuracy of the model. The model is evaluated each step against a certain minimization function, e.g. minimizing the difference between output and results, MSE,... The algorithm tweaks the weights and biases so that they can calculate with gradient descent the minimal value for C(w,b). This optimization process eventually results in the most accurate model.

5.1.2 Recurrent neural networks

Regular neural networks assume that inputs and outputs are independent from each other. This assumption is however not useful if one wishes to forecast if a crash will occur in the next few time steps. Recurrent neural networks, or RNNs (McClelland et al., 1986), solve this problem as they are neural networks that can process sequential data. They take the previous outputs or hidden states as input. It can be thought of as multiple copies of the same network, each passing information to a successor.

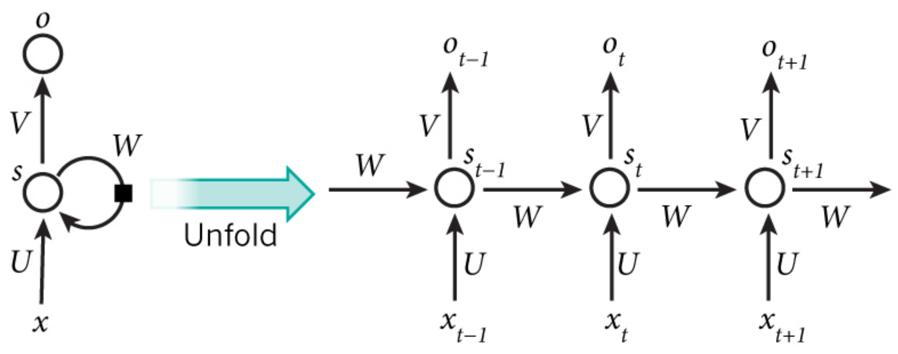


Figure 5.4: The unfolding of a RNN in it's different sequential components. Source: Nature

Figure 5.4 outlines an example of a RNN structure that consists of three layers. The algorithm behind a RNN goes as follows:

- x_t being the input vector at time t. In the case of financial time series this could be a bundle of features like volume, return, open-close gap at time t.
- s_t is the hidden state at time t. It is often referred to as the memory of the network and is dependent upon the input vector x_t and the previous hidden state s_{t-1} as following: $s_t = f(Ux_t + W_{s_{t-1}})$. With f being a non-linear function such as *tanh* or *relu*.
- o_t is the output at step t

Although, an RNN is extremely useful, it suffers from the *vanishing gradient problem* (Hochreiter 1991; Bengio et al. 1994). This means that when additional layers are added using certain activation functions, the gradient of the loss function approaches zero, making it difficult for the network to train. For example, when using the sigmoid function, a big input space is squeezed into a tiny input space between 0 and 1. By consequence, a major change in the input of the sigmoid function will result in slight change in the output. The derivative therefore becomes minimal. The practical implication of this is that the RNN has difficulties storing more memory than 3-4 past instances. To overcome this issue the LSTM was introduced by Hochreiter and Schmidhuber (1997).

5.1.3 LSTM

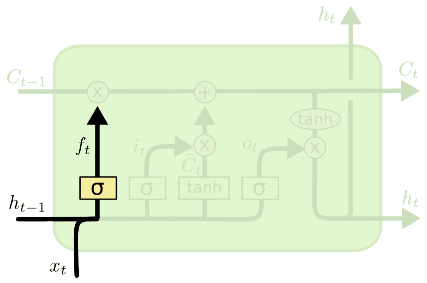
Long short-term memory (Hochreiter and Schmidhuber, 1997), often referred to as LSTM, is an RNN architecture that is much better at capturing long-term dependencies. They do not have a fundamentally different architecture from vanilla RNNs, but a different function is used to calculate the hidden states. To add new data to an RNN, the information as a whole is modified, there is no examination for 'crucial' and 'not so crucial' information. On the other hand, LSTMs make small changes to the information by multiplications and additions through a structure known as cell states. Following, an explanation of the core ideas behind an LSTM which is based on the work of Olah (2015), is provided. First, the components that make up the LSTM cell will be explained. Secondly, this work will discuss the iterative process the LSTM cell undergoes. An LSTM cell consists of the following elements

- *Cell state*: can be seen as a conveyor belt that transport information along the sequence chain
- *Forget gate*: decides which information is thrown away from the cell state
- *Input gate*: decides which new information is kept in the cell state
- *Output gate*: decides the content of the next hidden state

The previous stated elements make use of the following activation functions:

- *Sigmoid*: squishes values between 0 and 1
- *Tanh*: squishes values between -1 and 1

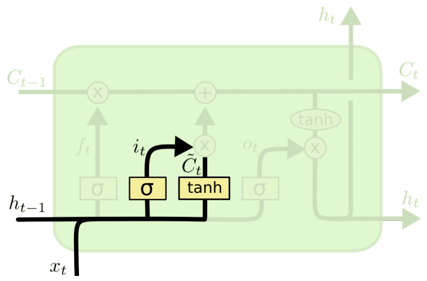
In the first step the LSTM needs to decided which information is kept and which is thrown away from the cell state. It takes h_{t-1}, the previous hidden state, and x_t, the input at time t into account. It passes through *the forgot gate*,which is a sigmoid function, and outputs a number between 0 and 1. A value close to zero indicates that it should be forgotten and a value closer to 1 implies that it needs to be retained.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Figure 5.5: Step 1: Passing information through the forgot gate (Olah, 2015)

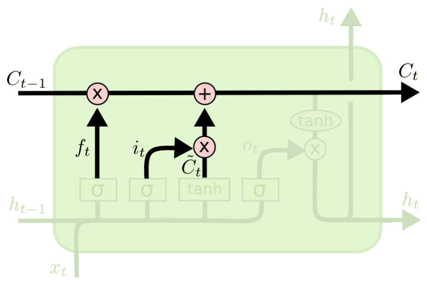
The second step, has as goal to 'decide' which information needs to be stored in the cell. This process consist of two main components, which both again have h_{t-1} and x_t as input: (i) sigmoid layer which decides which values will be updated (ii) tanh layer that creates a vector of new candidate values.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Figure 5.6: Step 2: Decide which information needs to be stored (Olah, 2015)

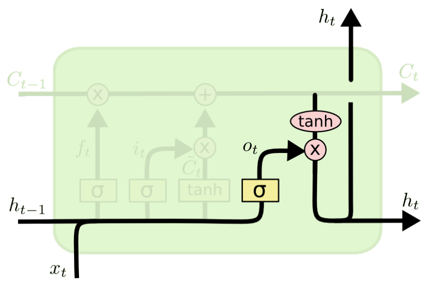
Thirdly, the old cell state C_{t-1} is 'converged' into the new cell state C_t as following: the old state is multiplied by f_t, while the two components calculated in step 2 are multiplied.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 5.7: Step 3: Converge the old and new cell (Olah, 2015)

Lastly, the output is calculated based on a filtered version of the cell state. First, a sigmoid function is run to decided which parts of the cell state are relevant. Secondly, the cell state is pushed through a tanh and multiplied by the output of the sigmoid, this results in an output of only relevant information. The new cell state generated in *step 3* and the hidden state (*step 4*) are carried over to the next iteration.



$$o_t = \sigma\left(W_o \, [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 5.8: Step 4: Calculate the output (Olah, 2015)

5.1.4 LSTM-FCN

Having explained the theory of the different components necessary to derive a quality NN architecture, we will now set-up our final machine learning architecture. The architecture should be capable of learning the nuances between series that lead to a crash and series that will not lead to a crash. The architecture this work is looking for needs to have superior performance in binary classification, capable of handling unbalanced data (as crashes rarely occur) and performing well for univariate series (as the only input variable will be standardised returns).

The architecture that satisfies those conditions is an LSTM - Fully Convolutional Networks (LSTM-FCN), proposed by Karim et al. (2018). The main reason for choosing this architecture is that it is the most accurate architecture for univariate time series classification. For unbalanced data sets, Geng and Luo (2019) also proved its superior performance. The key idea behind the architecture, outlined in figure 5.9, is to compensate with the addition of LSTM blocks for the disadvantages of temporal convolutions. Expanding the details behind this architecture would go beyond the complexity of this introductory work, therefore we refer to the paper of Karim et al. (2018) for more information. Rather, we will focus on tweaking the proposed parameters to fit our goal of successfully identifying time series leading to a crash.

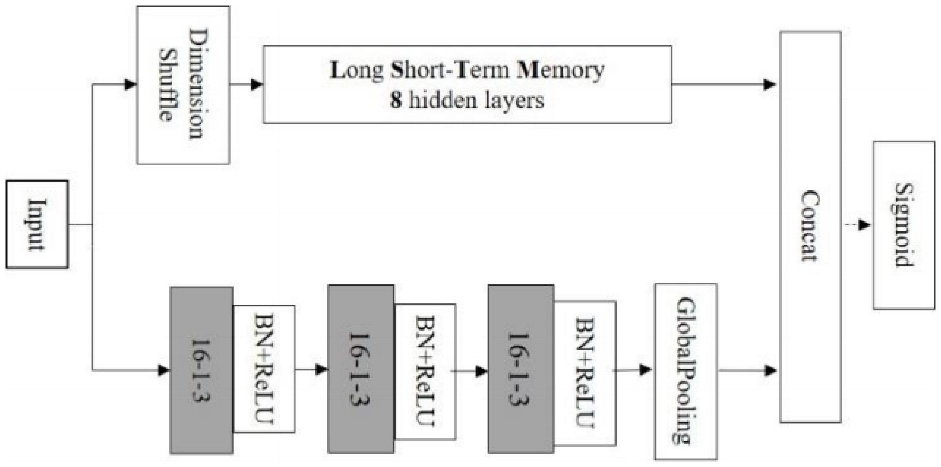


Figure 5.9: The architecture of LSTM-FCN (Geng and Luo, 2019)

5.2 Preparing for analysis

5.2.1 Definition drawdown

Before being able to predict crashes, one needs to define a crash. Unfortunately, an objective definition has not been agreed upon by the financial community. Johansen (2004) has shown that the largest ϵ-drawdowns have been positive identified as outliers and by consequence as "crashes". ϵ-drawdowns are defined as as a persistent decrease in the price over consecutive days from a local maximum to the next local minimum ignoring price increases in between the two of relative size less than ϵ.

The final definition of a crash is as a result dependent on two parameters: (i) The percentile of the worst drawdowns κ (ii) the maximum price increase in between the local maximum and minimum ϵ. This work applies the following methodology to come to a consistent choice for both parameters. First, we will look what the literature suggest for the choice of those parameters. Secondly, the features of the crashes will be calculated for the different parameters. Thereby, we will evaluate which parameter configuration produces features closest to those observed in financial time series.

5.2.1.1 Kappa

Johansen and Sornette (2004) identified the most extreme cumulative losses (i.e. drawdowns) in a large set of financial assets and showed that they belong to a probability density distribution, which is distinct from the distribution of the 99% of the smaller drawdowns which represent the normal market regime. Jacobsson (2009) is sceptical about this reasoning as the fitting procedure is questionable according to her. We will nevertheless continue with the definition set by Johansen and Sornette (2004), as no constructive alternatives are offered. However, we encourage further research to experiment with how different *kappa* values might affect the performance of the model (section 6.2.3)

5.2.1.2 Epsilon

Johansen and Sornette (2004) have suggested that ϵ should be bounded by $0 < \epsilon \leq \sigma$ where σ is the historical volatility. They further claims that as the distribution of noise in financial time series on short timescales is not know it is hard to set a more narrow band. As in our work the returns are normalized, the ϵ should be between 0 and 1. In table 5.1, for different epsilon values, the crash characteristics[1] of the synthetic and financial data are compared. This work has selected an epsilon value of 0.5 as the relative differences are smallest for average drawdown and average duration. Once again, we urge further research to experiment with this parameter.

ϵ	**Avg. Drawdown**		**Avg. Duration**		**Frequency**	
	Real	*Synt*	*Real*	*Synt*	*Real*	*Synt*
0	-10.8	-8.2	4.1	3.4	406	386
0.25	-11.7	-9	5.4	4.3	471	469
0.5	-12.6	-9.9	6.1	5.3	548	553
0.75	-13.4	-10.6	7.1	6.2	617	616
1	-14	-11	8.4	7	666	666

Table 5.1: Comparison of crash characteristics for different values of ϵ

[1]The crash characteristics are similar to those in section 4.2.3.2

5.2.2 Preparing series and labels for the machine learning model

5.2.2.1 Preparing the synthetic data

The purpose of this subsection is to prepare the data for the machine learning model. In chapter 4, a phase transition model is found that succeeds in generating series exhibiting stylized facts similar to those found in financial time series. This model is a 4-state Potts model with a temperature of $1.01\beta_c$ and a grid size of 100x100. The model thermalizes for 1000 steps and each sequence is 10 000 'time' steps long. Having generated series, our next step is subselecting sequences and label them whether they end up crashing or not which is outlined in box 2. After the series are processed, the synthetic data is split 60-40 into training set and validation set.

1. Iterate over the generate sequences
   - (a) Identify for each sequence all the drawdowns, with a drawdown being consecutive losses ignoring for a 0.5 standard deviation increase (as outlined in subsection 5.2.1)
   - (b) Define the 1 % worst drawdowns as crashes (subsection 5.2.1)
   - (c) For each sequence generate rolling windows of size 100[a] and increments of 1, e.g. the first rolling window contains observations for time step 1 through 100, the second rolling window contains observations for time step 2 through 100 + 1, and so on
   - (d) Check for each rolling window whether it will result in a crash within 5 trading days[b], with a crash being labeled as 1 and a no-crash labeled as 0.

[a]This is a parameter that is tuned in subsection 4.2.3.2
[b]See further work 6.2.3

: Algorithm for preparing series and labels for the machine learning model

100 series of 10 000 time steps result in 1 000 000 rolling windows, with about 990 000 rolling windows that do not result in a crash and 10 000 rolling windows that result in a crash. It is important for the reader to be aware that each crash will cause five rolling windows to be labeled as true. When evaluating the final model, the reader should keep this in the back of his head, as the windows that lead to the same crash can be correlated to some extent.

5.2.2.2 Preparing the real financial data

The real financial data (table 4.1) is prepared for analysis in the same way as explained in box 2. Once the series are processed, the data will be split in different subsets as outlined in table 5.2. For example, the first 20 % will be the first validation set, it is important to note that the first 20 % is the cumulation of the first 20 % of each asset. Which means that each split does not necessary contain time series from the exact same time period, as the start date of the different assets is different. We argued that this is not a problem given that the assets are uncorrelated (figure 4.2).

Subset name	Distribution
Validation set 1	[0%:20%[
Validation set 2	[20%:40%[
Train set	[40%:80%[
Test set 1	[80%:100%[

Table 5.2: Different subsets of real financial time sequences

5.2.3 Unbalanced problem

The presence of crashes implies a high imbalance between sequences that will lead to a crash and that will not lead to a crash. This subsection is devoted to solve this unbalance problem inherent to predicting crashes in financial time series. The first paragraph explains why a *traditional cost-insensitive model* does not seem fit for our unbalance problem. Thereafter, different solutions are proposed. The second paragraph explains the introduction of *cost-sensitive learning*. The following paragraph explains *thresholding* and how it can be a solution for the unbalance problem. The fourth paragraph thematises *other solutions* explained in the literature. Lastly, a test is executed to decide upon the best solution to solve the unbalance problem.

Training a traditional (cost-insensitive) classification model on a highly imbalanced dataset will lead to predicting everything being the majority class. As the model strives to increase the accuracy, it is the right thing to do, to label everything as the majority class (Drummond and Holte, 2000).

Solving the imbalance problem thus becomes only meaningful when either the distribution of the training and test set is unequal or if the cost of different types of error (false positive and false negative in the binary classification) is not the same. The first is here not the case as the distribution crash-series – no-crash-series are equal.The latter is however more relevant, the cost of misclassifying a major crash is insanely high, by consequence labelling crash correctly should not be weighted equally as labelling a non-crash correctly. One could distribute the weight over the two classes to resemble the actual economic cost of a major crash. This is not evident as the cost is not easily estimated. The literature proposes more pragmatic solutions. Japkowicz and Stephen (2002) suggest to choose the weights proportional to the number of positive and negative training cases so that:

$$\omega_0 n_0 = \omega_1 n_0 \tag{5.1}$$

Another solution to the imbalance problem is called thresholding (Buda et al., 2017). The idea is that one adjusts the decision threshold of a classifier to minimize an arbitrary criterion. Essentially, we will determine the threshold value for a certain recall on a 'validation' portion of financial data. That threshold value is then applied on the test set of the financial data to determine its final performance.

Some authors (H. He and Garcia 2009; Buda et al. 2017) propose a more pragmatic solution to the imbalance problem. A widely used method is undersampling. It consists of taking a subsample of the majority class to match the minority class. The main disadvantage of this method is that it discards potentially useful data. As there is a thin line between bubbles that result in crashes and bubbles that eventually do not crash this method does not seem fit for our purpose. Oversampling is another widely suggested method to fight imbalance problems. As in our case, the training data is anyway produced synthetically, generating extra cases of the minority class or deleting the majority class can be considered similar. By result, this technique is dismissed for the same reasons as described in the event of undersampling.

Cruz et al. (2016) claims most techniques are insignificant to tackle the imbalance problem. He further states that a post-processing step in selecting effective thresholds is the single most efficient measure. Therefore, this work will only introduce class weights and do a post-processing step for selecting the best performing thresholds on real financial data. The best performing class weights are determined by a hyper-parameter search outlined in subsection 5.4.

5.2.4 Performance measure

A classifier's performance is usually measured by *the predictive accuracy* derived from the confusion matrix (figure 5.11), defined as:

$$acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

Accuracy as a performance measure has limitations with an unbalanced dataset, as the measure is difficult to interpret given the enormous amount of TN in our case. What follows is a discussion of alternative performance measures that better fit the purpose of this thesis.

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

Figure 5.10: Confusion Matrix

The Receiver Operating Characteristic (ROC) curve is a standard technique for summarizing classifier performance over a range of trade-offs between true positive and false positive error rates (Swets, 1988). ROC curves can be thought of as representing the family of best decision boundaries for relative costs of TP and FP. On a ROC curve the X-axis represents %FP = FP/(TN + FP), which is equal to 1- specifity. The Y-axis represents %TP = TP/(TP + FN), which is the sensitivity or recall(infra) . The ideal point on the ROC curve would be (0,100), that all positive examples are classified correctly and no negative examples are misclassified as positive. The line y = x represents the scenario of randomly guessing the class (Maimon and Rokach, 2005). A single operating point of a classifier can be chosen from the trade-off between the %TP and %FP, that is, one can choose the classifier giving the best %TP for an acceptable %FP (Neyman-Pearson method) (Egan, 1975).

Area Under the ROC Curve (AUC) (Bradley, 1997) is a useful metric for the performance of the classifier as it is independent of the selected decision criterion and prior

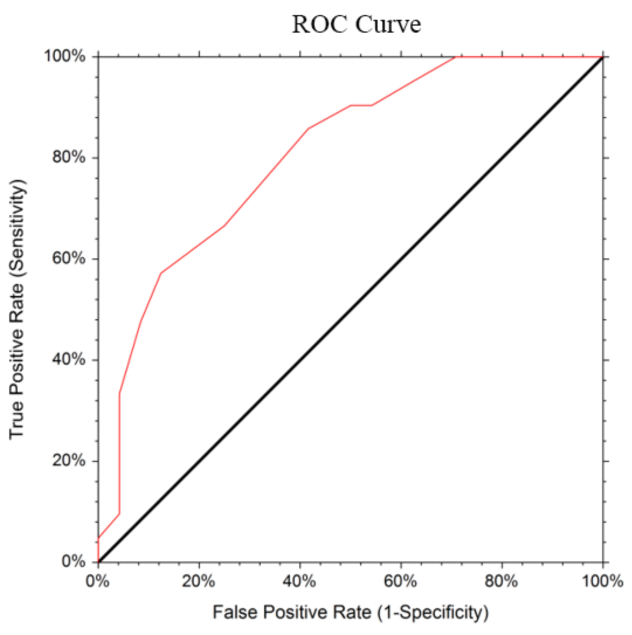


Figure 5.11: ROC Curve

probabilities. It tells how much the model can distinguish between classes. An excellent model has AUC close to 1 which indicates it has good separability measurement. When the AUC is 0.5, it means model has no class separation capacity. The useful aspect about AUC is that it is class balance insensitive (Fawcett, 2006) and it does not require the choice of a decision threshold.

However, shown in the paper by Davis and Goadrich (2006), ROC and the accompanied AUC can be misleading in highly unbalanced situations. Moreover, the paper suggests that the precision-recall curve (Buckland and Gey, 1994) is more fit for the detection of rare events. It can be derived from the confusion matrix (figure 5.11), and are defined as:

$$precision = \frac{TP}{TP + FN} \tag{5.3}$$

$$recall = \frac{TN}{TN + FP} \tag{5.4}$$

The main objective, in our case, is to improve the precision without hurting the recall. Recall and precision goals, however, can often be conflicting, as the number of false positives can also be increased by increasing the true positive for the minority class ; this will reduce the accuracy. Typically these two metrics are combined together in a metric called F_β (i.e., harmonic mean of precision and recall), which eases comparison of different systems, and problems with many classes. The expression for the F-value is as follows:

$$F_\beta = \frac{(1+\beta^2) * recall * precision}{(\beta^2 * recall) + precision} \tag{5.5}$$

β is a parameter that defines the preference between recall and precision. If β is 1 precision and recall are equally weighted. This work will opt for a β equal to one as changing the β endangers the interpretation of the F-score.

Given the discussed information, we will structure the performance measure and evaluation as follows. First, we will define the loss function as the F_1 derived metric $1 - F_1$, this stimulate the model while training to reduce the loss function and by consequence to increase F_1. The different hyper-parameters will be evaluated by a precision-recall curve on a validation portion of the real financial data. Lastly, the final model will be performed through a confusion matrix

5.3 Structuring the machine learning model for training

All the puzzle pieces are now in place to structure our machine learning model for training. Thereby, we will deploy the machine learning set-up discussed in subsection 5.1.4, and use the series prepared in subsection 5.2.2 as input. We customized the loss function to tackle the unbalance problem discussed in subsection 5.2.4, by taking $1 - F_1$ as the loss function. The model will attempt to decrease the loss function and by consequence increase the F_1 score, which is the weighted average between recall and precision. The hyper-parameters are tuned by evaluating them against validation set 1(table 5.2), unless specified otherwise. On each iteration the hyper-parameter is selected resulting in the most favorable precision-recall curve (section 5.2.4).

Algorithm 1: Machine learning algorithm

```
Data: Synthetic generated data
Initialise($model_{weights}$,lr,batch size)
while $i < K$ do
    Train($Data_{train}$, $model_{weights}$,lr,batch size)
    validateLoss = Validate($Data_{Validate}$)
    if $validateLoss < bestValidateLoss$ then
        Save $model_{weights}$
        bestValidateLoss = validateLoss
    else
        k = k +1
        if $k == 5$ then
            lr = updateLearningRate(lr, i)
            k = 0
        end
    end
    i = i +1
end
```

The mechanism behind the neural learning model is described in algorithm 1. The model runs for a certain amount of epochs K. One epoch is when the whole data set is passed through the neural network, both forward and backward. At every epoch, the model is assessed by the synthetic validation set. If the validation loss (paragraph 5.2.4) has not decreased for 5 epochs, the learning rate will be reduced by a factor $\frac{1}{\sqrt[3]{2}}$. It is important to note here that the validation loss is calculated on the validation set of the synthetic data. The drop-out rate is set at 50% instead of the paper's proposed 80%. We will run each revision for 20 epochs due to computational limitations, and have found through an iterative process that 20 epochs show superior performance. All convolution kernels were initialized with K. He et al. (2015) proposed initialization. The optimizer of choice is the Adam optimizer proposed by Kingma and Ba (2014) .

As we have defined our framework, the next section is dedicated to tuning the hyper-parameters to achieve the highest performance.

5.4 Hyper-parameter tuning

This section is devoted to tuning five main parameters: (i) the amount of data (ii) the class weights (iii) the learning rate (iv) the batch size (v) the amount of cells. This part of the thesis is designed in such a way that when a parameter is found to be superior it is immediately deployed in the architecture to search for the next parameter.

5.4.1 Amount of data

Because of the set-up of this work, we have the privilege of generating as much training data as wanted. As insufficient data leads to poor performance and excessive data slows the algorithm excessively, it is important to select the right amount of data. To decide the amount of data needed, we follow an approach suggested by Cho et al. (2015). Thereby, the idea is to train the model on different amounts of data and calculate for each configuration the lowest achieved loss on the validation set of the synthetic data over five epochs. 5.12 shows the result with on the x-axis the amount of data and on the y-axis the validation loss. One can see that whereas for low amounts of data the results are relatively wonky, for 200 000 data points the validation loss starts to stabilize. As there is no significant improvement observed, this work opts to train on 300 000 datapoints. One epoch on 300 000 datapoints takes about 45 seconds on the structured outlined in 3.3.

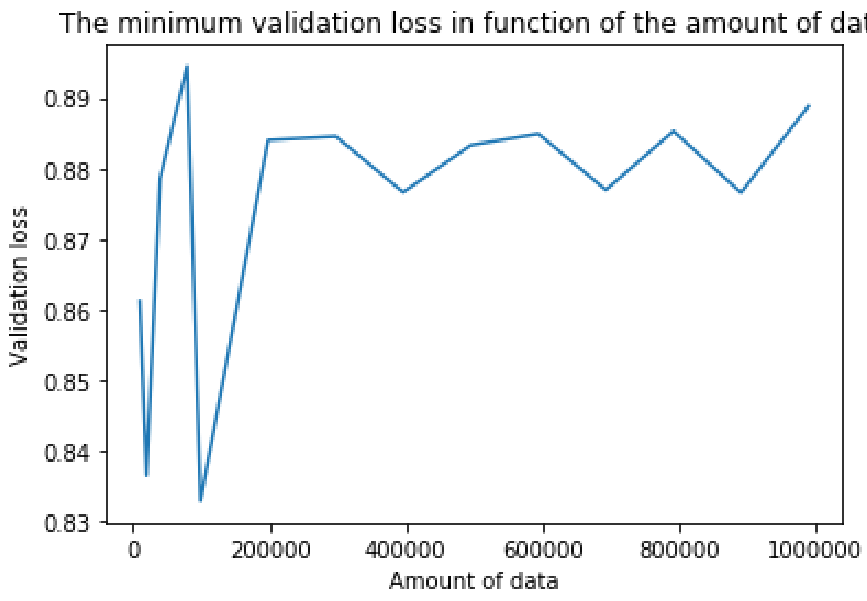


Figure 5.12: The loss on the validation set in function of the amount of data

5.4.2 Learning Rate

Through the method described in the paper by Smith (2018), the optimum learning rate range is found. The learning rate is increased from low to high over one epoch and the validation[2] loss is every time calculated. Plotting the loss in function of the learning rate, we select the minimum learning rate where the loss function begins to flatten and the maximum learning rate when the loss starts to increase again. Figure 5.13 depicts that the loss begins to flatten around -3.5, since the x-axis is logarithmic, this means that the minimum learning rate is $10^{-3.5}$. When the learning rate approaches 10^{-2} the loss begins to deteriorate, which we will take as our maximum learning rate. By consequence, this work will vary the learning rate from 10^{-2} to $10^{-3.5}$. Further, as outlined in algorithm 1, the learning rate will be lowered after 5 epochs of no performance improvement on the validation set of the synthetic data. The learning rate will be lowered with a factor $\frac{1}{\sqrt[3]{2}}$ as proposed by Karim et al. (2018).

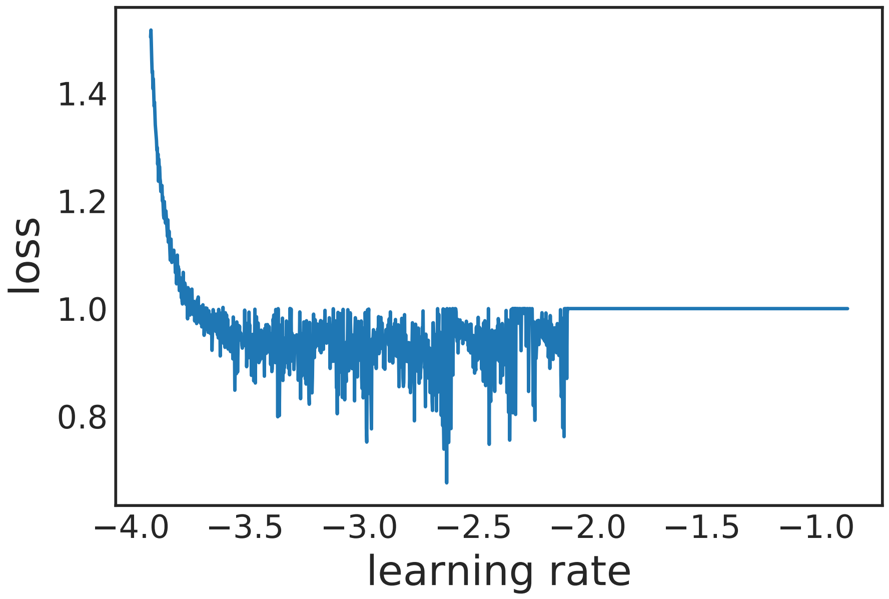


Figure 5.13: The loss on the validation set in function of the learning rate

5.4.3 Class weights

The class weights ω, used up until now, is given by the following formula: $\omega_0 n_0 = \omega_1 n_0$ and was discussed in section 5.2.3. This work wanted to examine the impact of different class

[2]Here we evaluate through the validation loss of the synthetic data, as this is required by the algorithm

weights on performance by performing a scan. The proposed class weights are arbitrarily chosen within reach of the one discussed in section 5.2.3 and are further outlined in table 5.3. The models were trained on the synthetic data and validate on validation set 1 (table 5.2). Figure 5.14 illustrates that the performance over different class weights is not significantly different. However, the performance of a class weight of 400 is slightly better in the low-medium range. Therefore we will continue with a class weight of 400.

Weight No bubble	Weight bubble
1	10
1	50
1	100
1	200
1	400
1	800

Table 5.3: Different class weights

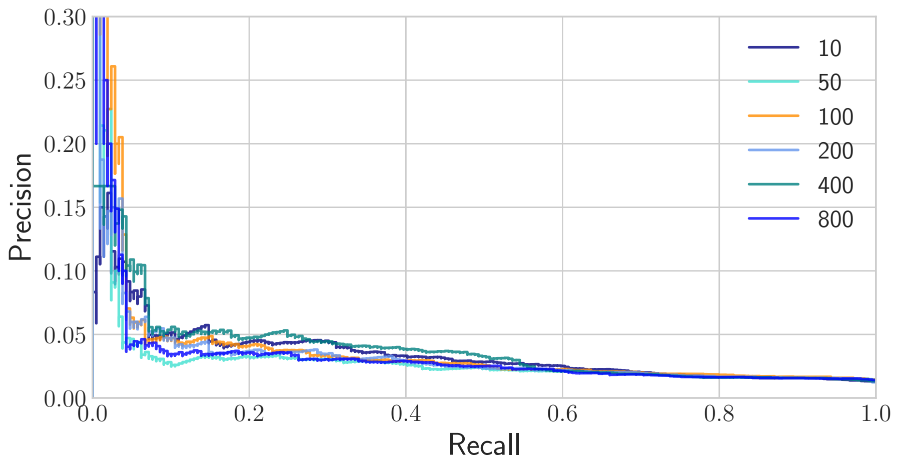


Figure 5.14: Precision recall on validation set 1 for different class weights

5.4.4 Batch size

There is not a structured approach in finding the perfect batch size. However, there are some clues to be found in the literature of better performing batch sizes. As we have an extremely imbalanced problem, the batch size needs to be taken sufficiently large, so each batch counts at least one positive example. Figure 5.16 shows results of the grid search. It is important to mention again the differences are quite insignificant, but visually it can be observed that a batch size of 256 performs slightly better.

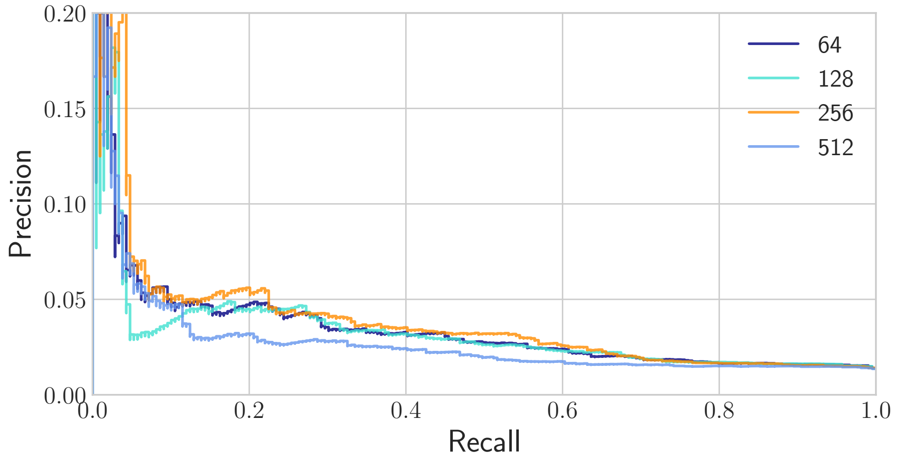


Figure 5.15: Precision recall on validation set 1 for different batch sizes

5.4.5 Cells

As suggested in the paper of Karim et al. (2018), to find the optimal number of cells for the LSTM, a parameter search must be performed. Figure (5.16) shows that the 8-cell model is superior for low recall values. Further, since we want to avoid overfitting we opt to train our model with 8 cells.

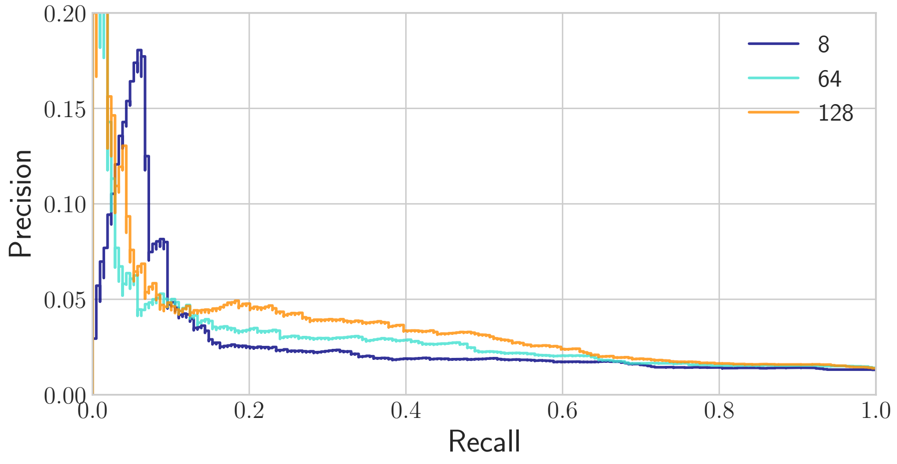


Figure 5.16: Precision recall on validation set 1 for different cells

5.4.6 Conclusion

The final model takes the model proposed by Karim et al. (2018) as its foundation, however with the following adaptions to suit our goals. Because of the problem's imbalanced nature, we choose to use the F1 score as the performance metric rather than binary cross entropy (section 5.2.4). The drop-out rate is 50% instead of 80%. Our model is ran for 20 epochs. As shown in figure 5.12 the F_1 score begins to stabilize around 300 000 datapoints, which is taken as the amount of data we will feed into the final network.

The ideal learning range, as outlined in 5.4.2, is between 10^{-2} and $10^{-3.5}$. Through a parameter search we have found that the ideal *crash - no crash* weight ratio is 400:1 (subsection 5.4.3), the best batch size is 256 (subsection 5.4.4) and the amount of LSTM cells is 8 (figure 5.4.5)

Chapter 6

Deploy the machine learning model on financial data

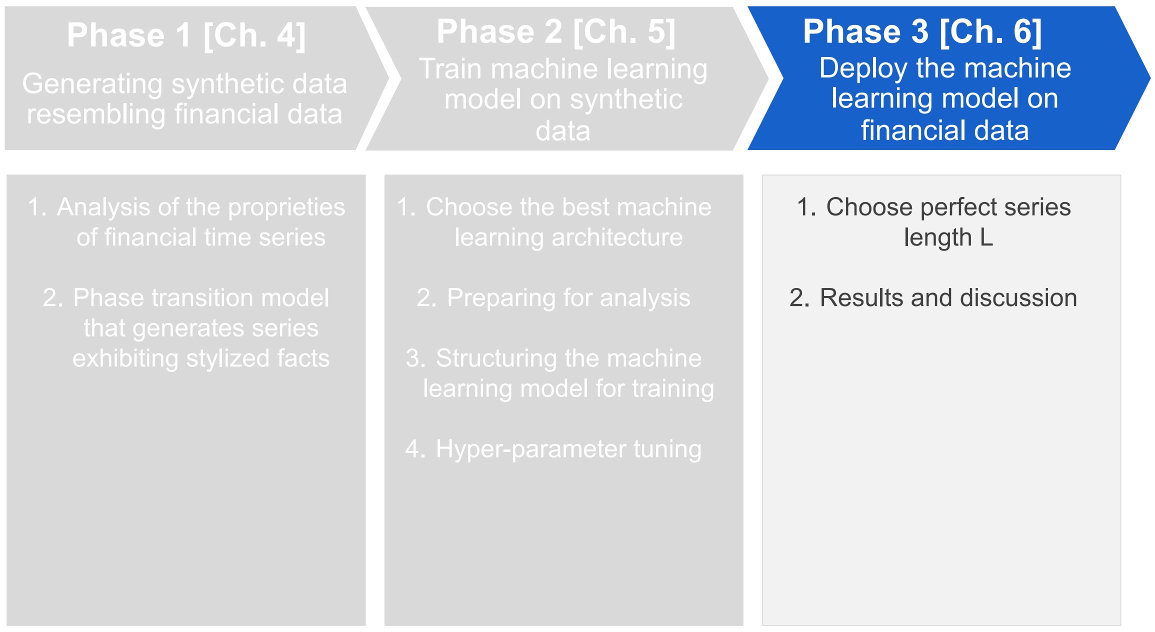

Phase 1 [Ch. 4]
Generating synthetic data resembling financial data
Phase 2 [Ch. 5]
Train machine learning model on synthetic data
Phase 3 [Ch. 6]
Deploy the machine learning model on financial data
1. Analysis of the proprieties of financial time series
2. Phase transition model that generates series exhibiting stylized facts
1. Choose the best machine learning architecture
2. Preparing for analysis
3. Structuring the machine learning model for training
4. Hyper-parameter tuning
1. Choose perfect series length L
2. Results and discussion

Figure 6.1: Phase 3: Deploy the machine learning model on financial data

Having composed the perfect model on synthetic data, our next step consists of deploying the model on real financial data. Two models are generated, starting from the model trained on the synthetic data. The first model is not subjected to post-processing and is directly implemented on the real financial data. The second model, on the contrary, undergoes first additional training on the real financial data's training set[1]. The results can be found in appendix A.1. Given the fact that the second model looks promising, we encourage further research to look more deeply into its performance. Nevertheless, this work continues with the first model to remain truthful to the concept of starting from a financial agent model and deploying it directly on real financial data.

The chapter consists of three central sections, in the first sections the perfect length L of the series are selected. The second section is devoted to discussing the results of deploying our final model on the test set. This will be done by testing the two proposed hypotheses in chapter 3.1. The third section focuses on areas of improvement where further research could concentrate on.

6.1 Choose perfect series length L

The model has so far always been trained on time series of 100 trading days. The question arises how the model performs when changing the time series length L. Figure A.4 shows the performance of the model on the second validation set1^0 for different length of time series L. It can be clearly seen that a model of 400 time steps is superior to others. This can be partly explained by the nature of an LSTM, which only takes the information into account that will arrive at solving the problem. Further, it is interesting to see that the second best performing model is one of 20 time steps, especially for low recall. This means that a lot of information is in the end of the time series, which confirms a finding by Sornette, Woodard, et al. (2013b).

[1] table 5.2

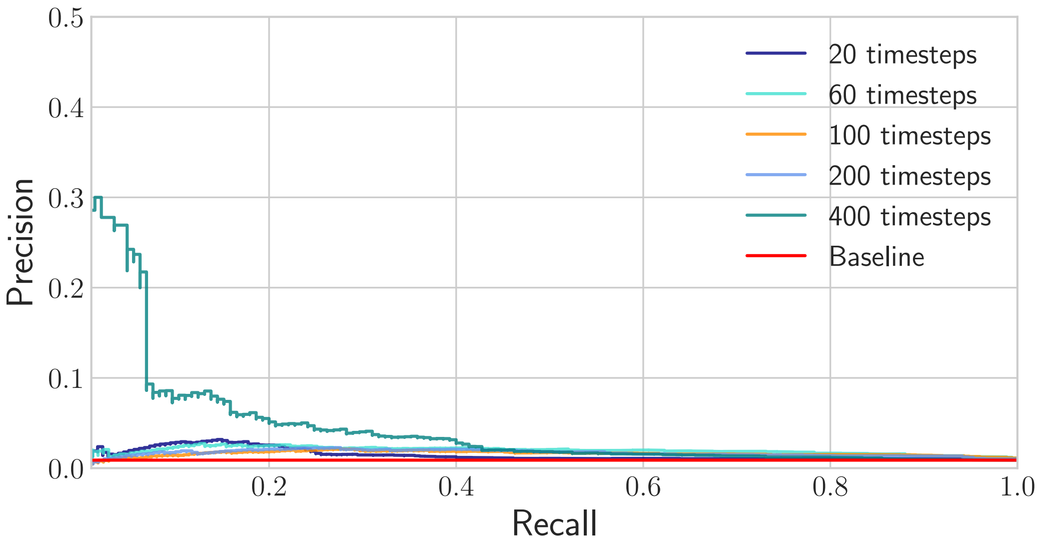


Figure 6.2: Precision-recall on validation set 2 for different length of the timeseries L

6.2 Results and discussion

Since the final model, which tries to predict whether a crash will occur within a week, has been reached, it is now time to evaluate its performance on the test set, allowing us to answer the hypotheses introduced in section 3.1. The first hypothesis is evaluated through two central methods. The first one being the precision-recall curve, which provides an overview of the performance of the model. The main drawback of this method, however, is that it estimates the thresholds on the same data set on which the analysis is performed. This does not enable us to conclude any predictive power because the performance is ex-post adjusted to the optimal thresholds. Therefore, additionally the confusion matrix will be discussed for various thresholds. The thresholds are derived from the precision-recall curve of the train set[2]. This gives us an understanding of how many crashes our model could correctly predict, how many crashes our model could not anticipate, and how much crashes it wrongly predicted. The second hypothesis is evaluated using a simple trading strategy, in order to test whether the model is economically viable, as this is can not be concluded from the confusion matrix.

[2]table 5.2

6.2.1 First hypothesis

The first hypothesis proposed in chapter 3.1 is:

- H_0 The efficient market hypothesis holds, and the accuracy and precision of the model is no higher than a randomly guessing model
- H_1 The efficient market hypothesis does not hold, and the accuracy and precision of the model outperform a randomly guessing model

will be answered through the precision-recall curve and the confusion matrix.

6.2.1.1 Precision-recall

Figure 6.3 shows the precision-recall curve on the test set, with the red line representing a randomly guessing model. It can be noticed that the first part of the curve is flat, this is because multiple windows are predicted to have a 100 % probability to crash. In further work it is relevant to tweak the model so that more nuanced scores are predicted. For low recall values the precision-recall can be interpreted as follows: imagine the test set contains 100 crashes, if 10 crashes were predicted correctly (10% recall), the model would incorrectly forecast 40 crashes (25% precision). Hence, for low recall values the conclusion can be drawn that the model performs vastly better than a randomly guessing model. For higher recall values the precision is negligible so that when most crashes are predicted, the model performs only slightly better than a randomly guessing model.

Hence, the H_0 can be rejected with the nuance that the rejection does not seem to hold for larger recall values. Concluding predictive power need to taken with certain vigilance as the precision-recall curve, estimates the thresholds on the same data set on which the analysis is conducted. Further, as outlined in 5.2.2.2 it is important for the reader to be aware of the fact that the windows leading to the same crash are correlated to some extent which can cause the precision and recall to be skewed.

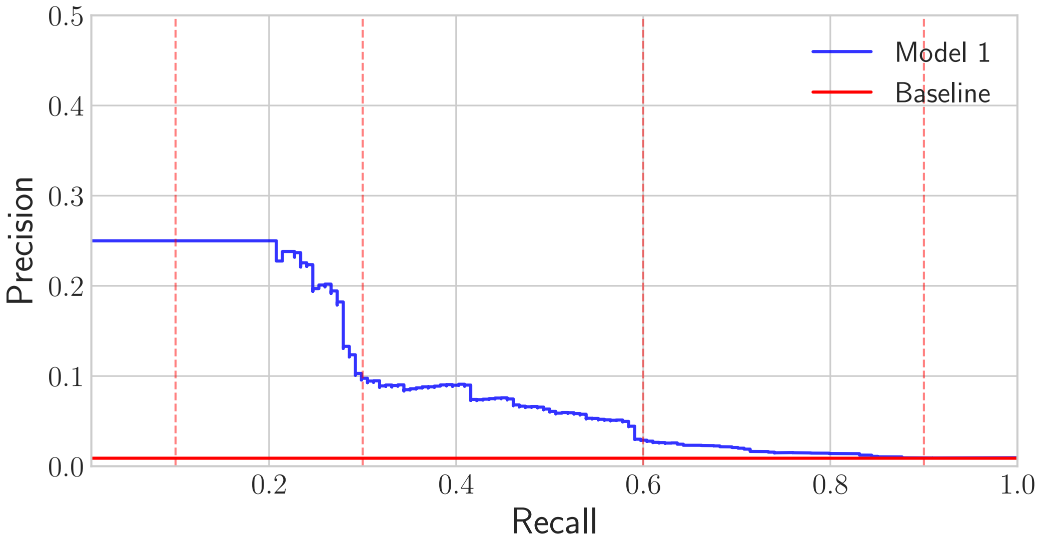


Figure 6.3: Precision-recall on the test set

6.2.1.2 Confusion matrix

The use of the confusion matrix has more conclusive power because it estimates the thresholds on a set other than the one on which the analysis is performed. Here, the confusion matrix is composed with thresholds calculated on the train set (figure 6.4). The thresholds were determined for four recall values (0.05,0.1,0.3,0.5), which were taken as low since the performance deteriorates after a recall value of 0.6 (figure 6.4)

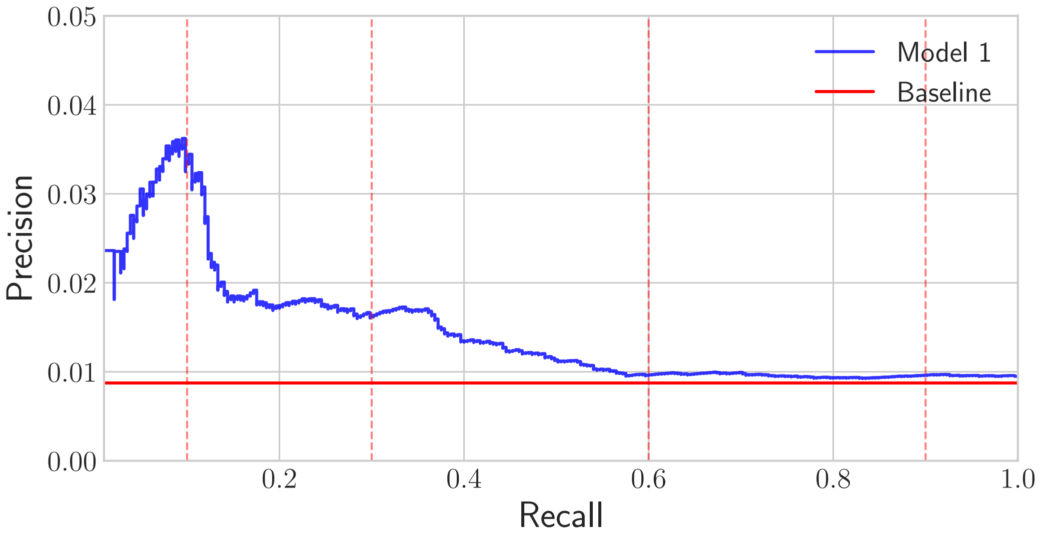
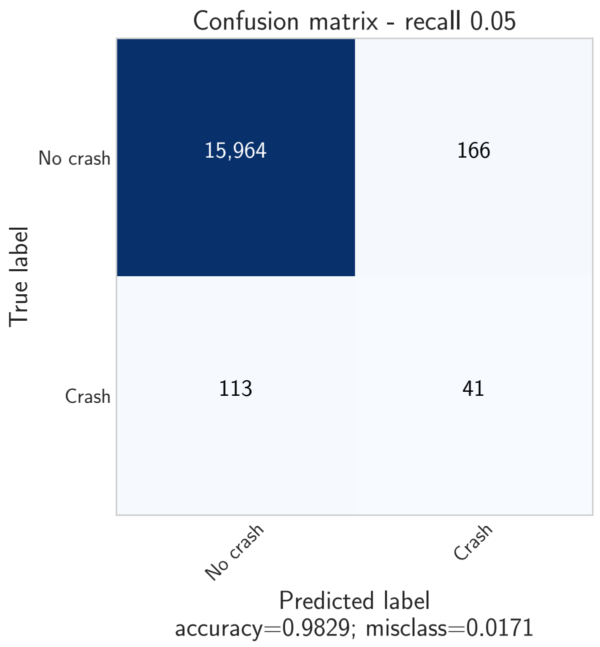
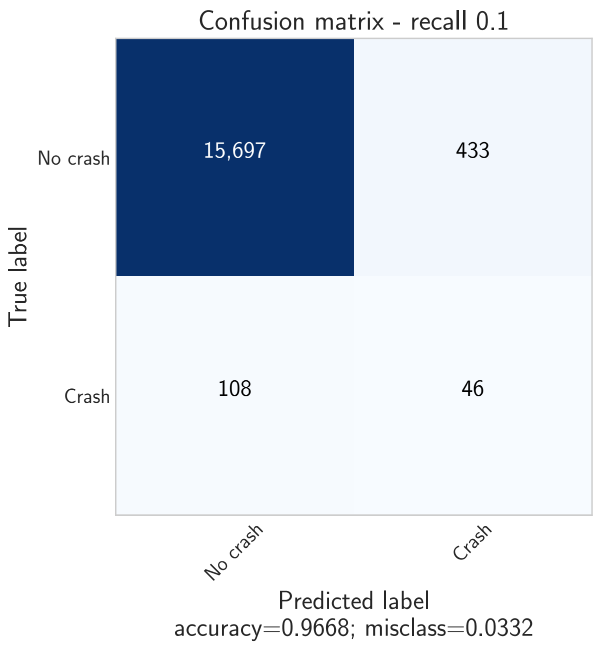


Figure 6.4: Precision-recall on the train set

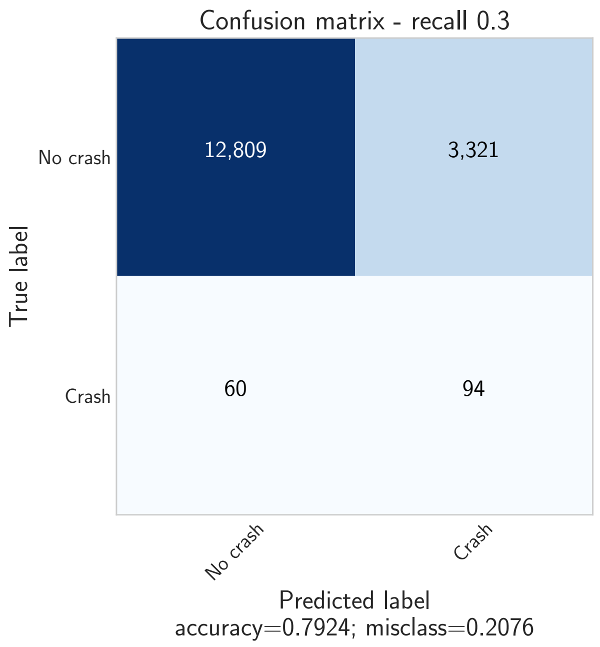
Figure 6.5 shows the confusion matrix for the different threshold values calculated on the train set. It can be clearly noted that the model is extremely powerful for low estimated recall values. Unfortunately, the performance quickly deteriorates, with each additional true positives introducing many false positives. This proves our presumption that the model is extremely sensitive to the threshold selection. For an estimated recall value of 0.05, the model correctly predicts 41 of the 154 crashes, while predicting 166 crashes wrongly. Again, we would like to point out that the moving windows leading to the same crash are correlated and can curve the outcomes. For higher recall values, the model is barely better than a randomly guessing model and its predictive power can be considered insignificant. This leads us to the conclusion that the hypothesis' rejection can be confirmed for lower estimated recall values and accepted for higher estimated recall values. Further research could focus on the subject of improving the threshold selection to achieve higher performance.



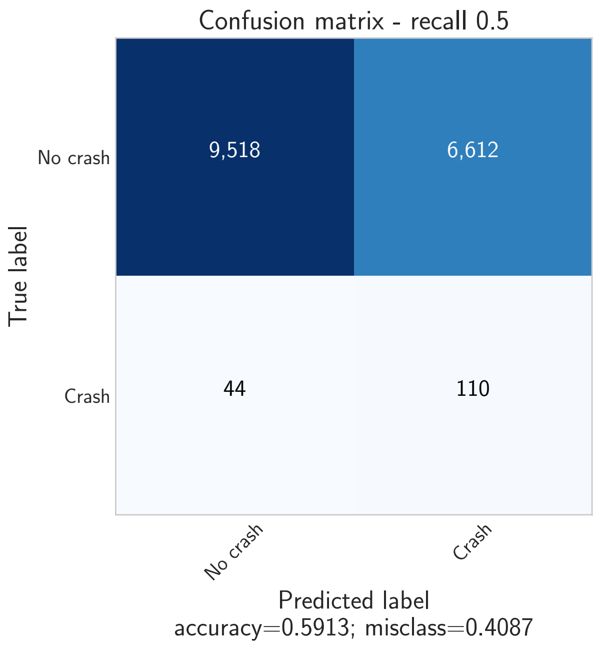
(a) Threshold calculated for recall of 5 %



(b) Threshold calculated for recall of 10 %



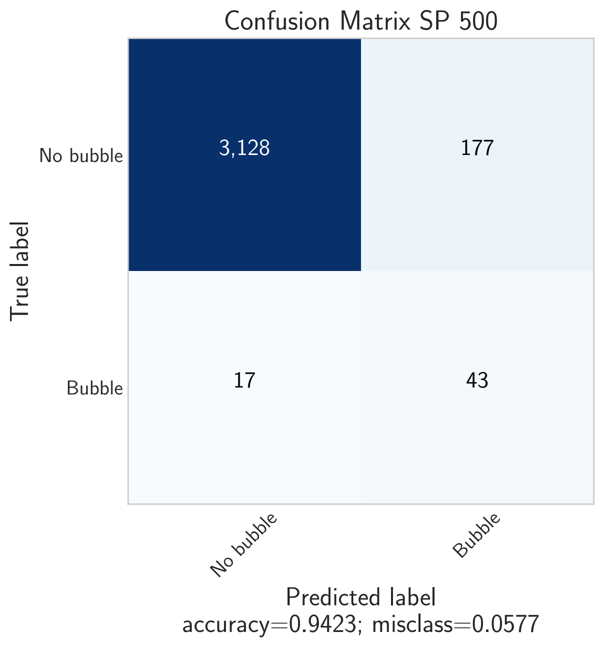
(c) Threshold calculated for recall of 30 %



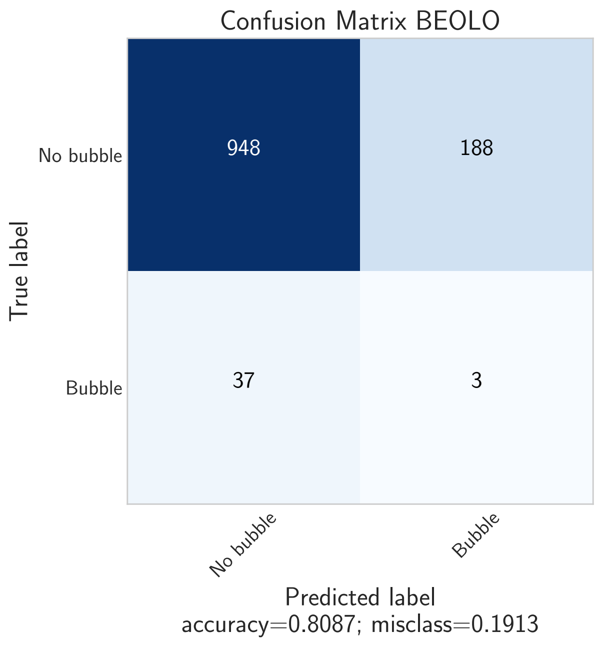
(d) Threshold calculated for recall of 50 %

Figure 6.5: Confusion matrix for different recall values, with thresholds calculated on the training set

So far, the analysis has always been executed on an asset assembly (section 5.2.2.2), with the underlying assumption that all assets behave similarly. We're keen to know how the model would perform on the individual assets. The confusion matrix for a recall of 10 % is decomposed in its individual assets in figure 6.6, it is clear to see that all the predicted bubbles come from two assets S&P 500, BEOLO. This shows that the thresholds can not be calculated in an aggregated manner, but rather should be calculated on each asset individually. Even though, the confusion matrix has helped us to evaluate the performance of the model, it fails to give us two important insights. First, it can not show us how the rolling windows behind predicted labels behave. Secondly, it does not show the economic feasibility. Plotting a simple trading strategy based on the model could help us to answer those questions.



(a) Confusion matrix applied on the test set of the SP 500



(b) Confusion matrix applied on the test set of the BELOLO

Figure 6.6: Confusion matrix for a recall value of 10%, with the threshold calculated on the training set

6.2.2 Second hypothesis

The second hypothesis proposed in chapter 3.1 is:

- H_0 The efficient market hypothesis holds, and the returns generated by the trading strategy do not outperform a buy and hold strategy

- H_1 The efficient market hypothesis does not hold, and the returns generated by the trading strategy do outperform a buy and hold strategy

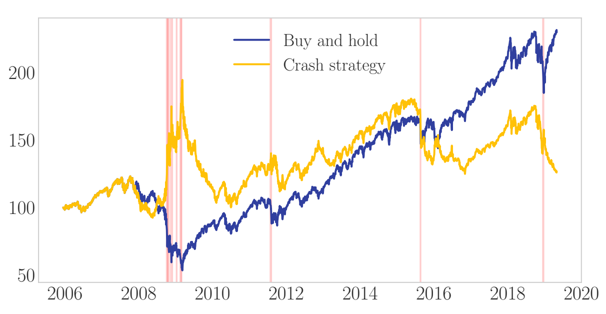
6.2.2.1 Trading strategy

Algorithm 2: Trading algorithm

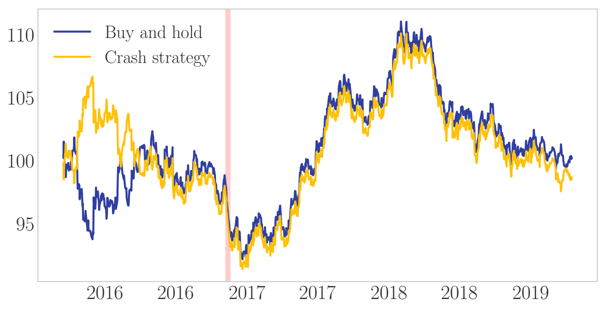
```
Scores = model_predict($movingWindow_{Test}$)
Position = True (True is long and False is short)
t = 0
Return = 0
for i in Scores do
    if Position == True then
        if i > Threshold then
            Position = False
            Return = Return - $log\_returns_i$
            t = 0
        else
            Position = True
            Return = Return + $log\_returns_i$
        end
    else
        if i > Threshold then
            Position = False
            Return = Return - $log\_returns_i$
            t = 0
        else
            if t >= 5 then
                Position = True
                Return = Return + $log\_returns_i$
            else
                Position = False
                Return = Return - $log\_returns_i$
                t = t +1
            end
        end
    end
end
```

This subsection is devoted to assess the economic viability of the model and see whether the null hypothesis can be rejected. Here, the threshold will be calculate on the train set for each asset individually. The threshold is calculated for a 10 % recall value, as it displayed the best precision-recall trade-off on the train set (figure 6.3). The trading strategy is extremely simple and discussed in detail in algorithm 2. In essence, the algorithm comes down to buying the asset when the crash probability is lower than the threshold and shorting when it is higher. The model maintains its short position for five days following the last crash 'signal', this is a parameter which could also be studied in further research .

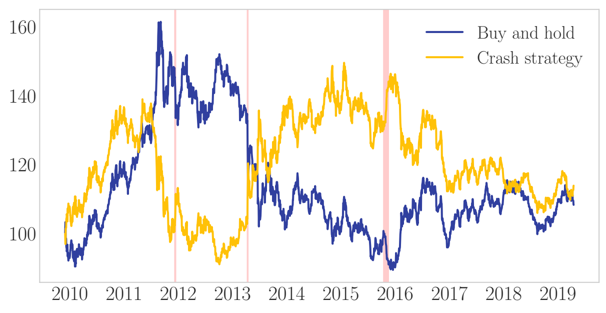
Figure 6.7 [figure A.5 in appendix] shows the performance of the model on different assets. For most assets the model does not result in superior performance, except for coffee (figure A.5h). Some assets even perform extremely bad (figures A.5c, A.5d, A.5h). The only fair conclusion is that the null hypothesis needs to be accepted. However, it is interesting to note that the model achieves at capturing some crashes and is able to profit from them, e.g. the financial crash of 2008 in the S&P 500 (figure 6.7a), the crash in coffee in 2013 (figure 6.7c). It is worth mentioning that the model is indicating on the last day of the test set [2/05/2019] to short the S&P 500, by this date 02/06/2019 the S&P 500 has fallen 5.6 % relative to its 2/05/2019 position.



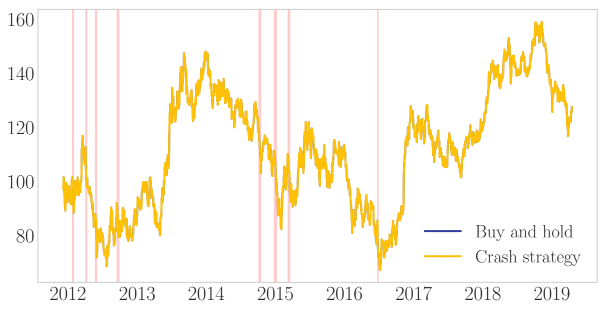
(a) S &P 500



(b) EUR USD



(c) LLoC1



(d) US10Y

Figure 6.7: Performance of a trading strategy based on the composed model

6.2.3 Further work

This introductory work has shown hopeful results that could possibly be used as the basis for further research. Many improvements can be made that will possibly lead to better results. This section offers an overview of improvements that according to the writer could have significant impact on performance. For each phase they are ranked from highest anticipated impact on performance to lowest anticipated impact on performance.

6.2.3.1 Phase 1: Generating synthetic data resembling financial data

This work has tried to generate synthetic data that exhibit powerlaws where the mean powerlaw of the synthetic data matches the range of powerlaws exhibited by real financial series. Further research should try to generate series that match the distribution of power laws rather than just the mean.

Even though, the Potts model is capable of grasping some basic dynamics present in the financial markets, it still lacks the complexity to be considered as a reliable financial agent model. The Kuramoto model is able to capture features as: degree of influence and volume which is not captured by the Potts model. Further, time is an inherent parameter to the Kuramoto model while it is an arbitrarily chosen monte carlo update step for the Potts model. This work believes that this model could achieve better at grasping the nuances presence in real financial data.

When extra computational power is at hand, it is advisable to find the best Potts model configuration through a grid search.

6.2.3.2 Phase 2: Train a machine learning model on synthetic data

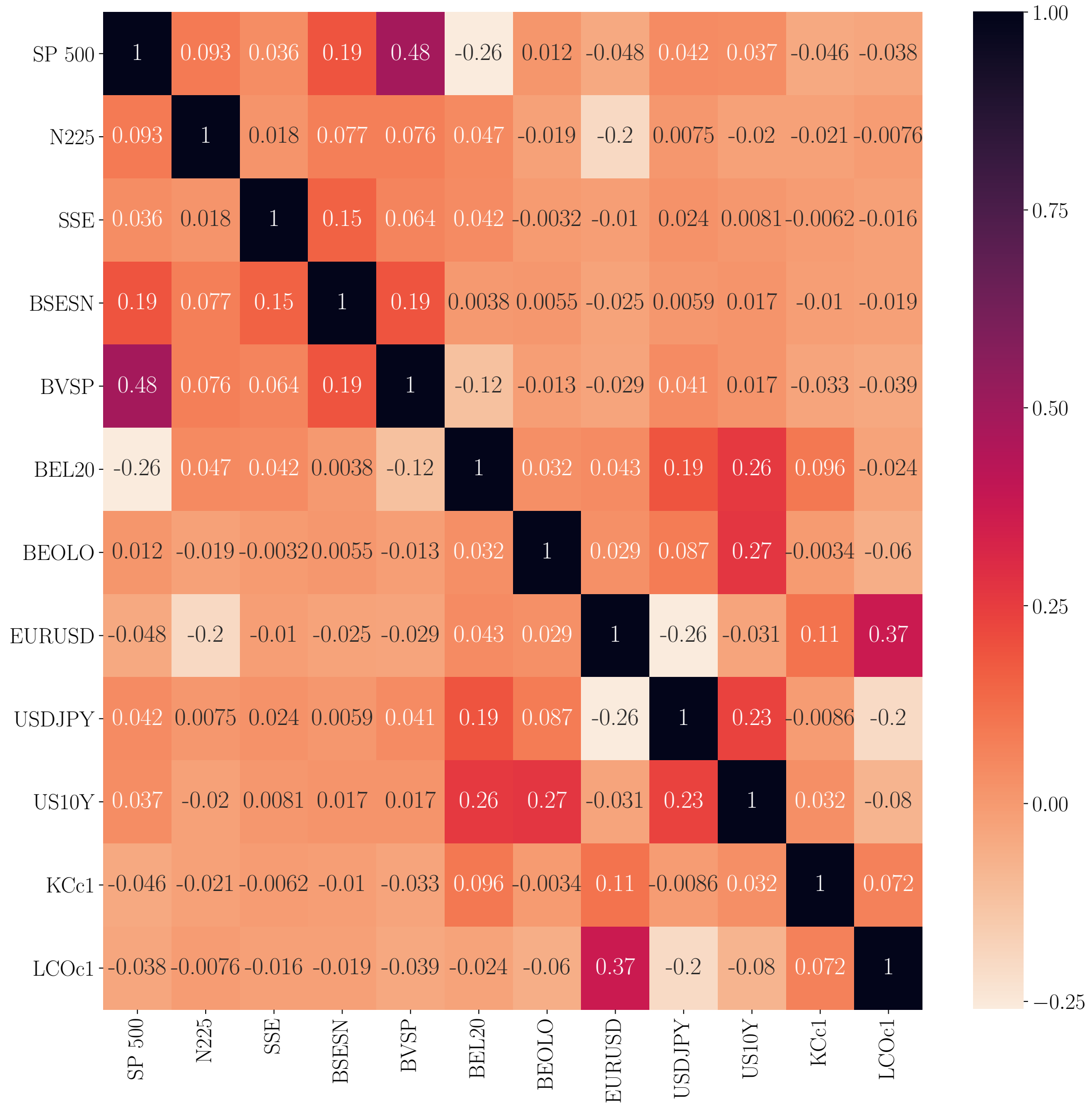
Firstly, the model proposed in appendix A.1, should be further examined as the first results look promising. Secondly, it would be interesting to see how the model would perform on different definitions of crashes. This would answer the question whether it can only predict crashes or rather drawdowns as a whole. Finally, the best performing parameters are always best identified by means of a grid search with interactions.

6.2.3.3 Phase 3: Deploy the machine learning model on financial data

Each set is the cumulation of the different assets. As each asset starts at a different point in time, each split does not necessary contain time series from the exact same time period. We advise further research to match the dates of the sets. We believe it is interesting to compose the confusion matrix for all the assets individually, but with the thresholds estimated for each individual asset rather than on the aggregation of the assets. Further, It'd be useful to look which series are behind the predictions and try to unravel the features behind the black box. For the moment, the model always tries to predict whether a crash will occur within a week. It would be fascinating to explore different time periods.

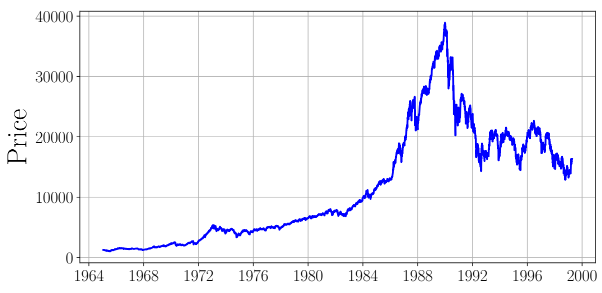
Appendix A

Appendix

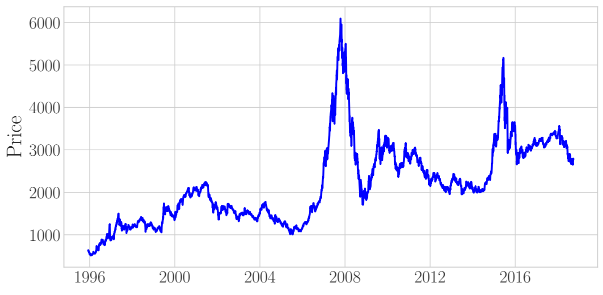


	SP 500	N225	SSE	BSESN	BVSP	BEL20	BEOLO	EURUSD	USDJPY	US10Y	KCc1	LCOc1
SP 500	1	0.093	0.036	0.19	0.48	-0.26	0.012	-0.048	0.042	0.037	-0.046	-0.038
N225	0.093	1	0.018	0.077	0.076	0.047	-0.019	-0.2	0.0075	-0.02	-0.021	-0.0076
SSE	0.036	0.018	1	0.15	0.064	0.042	-0.0032	-0.01	0.024	0.0081	-0.0062	-0.016
BSESN	0.19	0.077	0.15	1	0.19	0.0038	0.0055	-0.025	0.0059	0.017	-0.01	-0.019
BVSP	0.48	0.076	0.064	0.19	1	-0.12	-0.013	-0.029	0.041	0.017	-0.033	-0.039
BEL20	-0.26	0.047	0.042	0.0038	-0.12	1	0.032	0.043	0.19	0.26	0.096	-0.024
BEOLO	0.012	-0.019	-0.0032	0.0055	-0.013	0.032	1	0.029	0.087	0.27	-0.0034	-0.06
EURUSD	-0.048	-0.2	-0.01	-0.025	-0.029	0.043	0.029	1	-0.26	-0.031	0.11	0.37
USDJPY	0.042	0.0075	0.024	0.0059	0.041	0.19	0.087	-0.26	1	0.23	-0.0086	-0.2
US10Y	0.037	-0.02	0.0081	0.017	0.017	0.26	0.27	-0.031	0.23	1	0.032	-0.08
KCc1	-0.046	-0.021	-0.0062	-0.01	-0.033	0.096	-0.0034	0.11	-0.0086	0.032	1	0.072
LCOc1	-0.038	-0.0076	-0.016	-0.019	-0.039	-0.024	-0.06	0.37	-0.2	-0.08	0.072	1

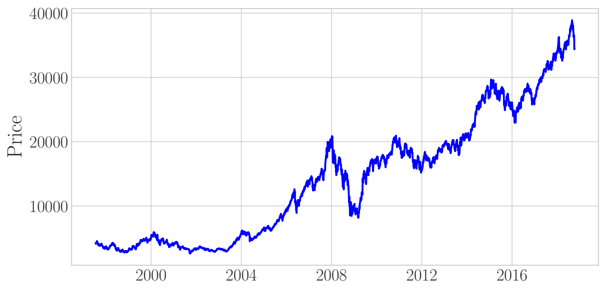
Figure A.1: Correlation between the various assets



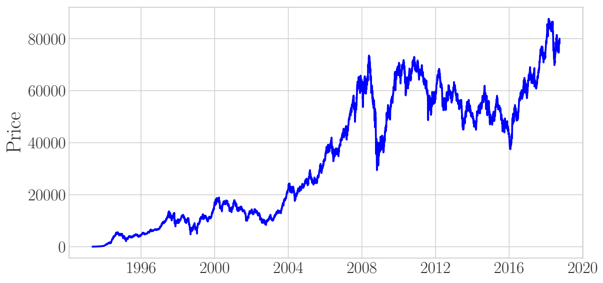
(a) N225



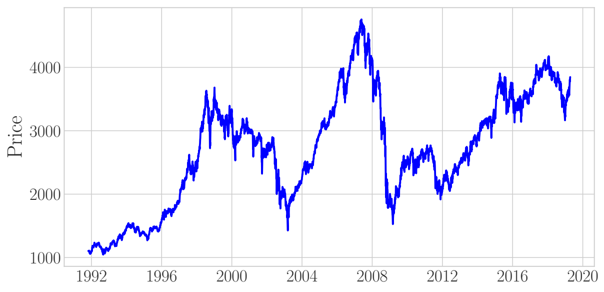
(b) SSE



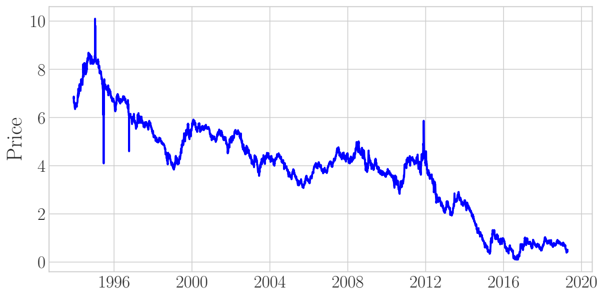
(c) BSESN



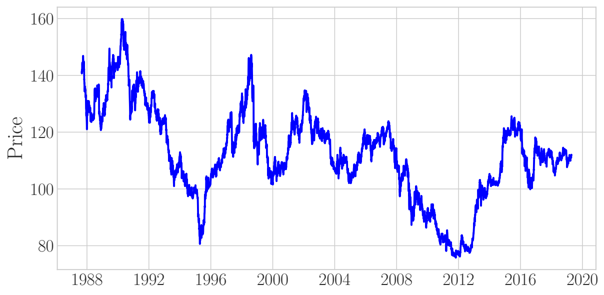
(d) BVSP



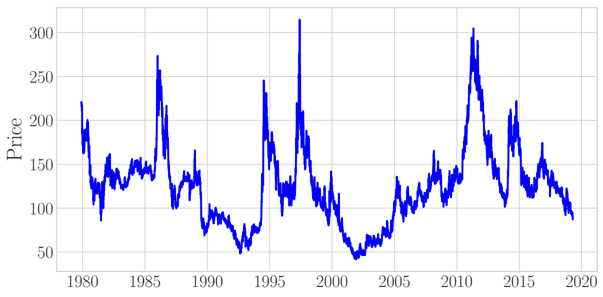
(e) BEL20



(f) BEOLO

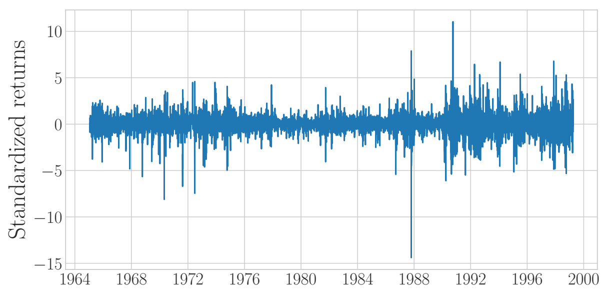


(g) USDJPY

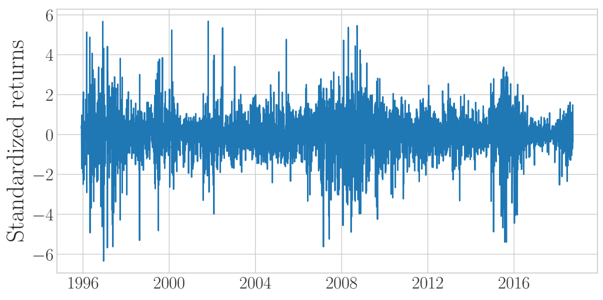


(h) KCc1

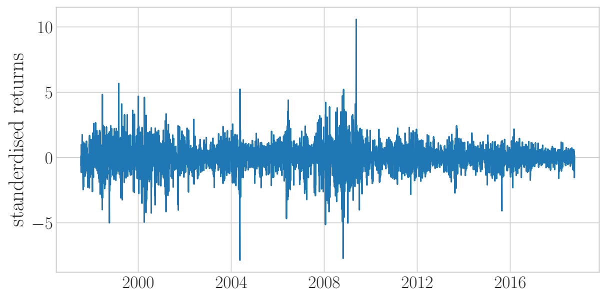
Figure A.2: Price evolution of the various assets



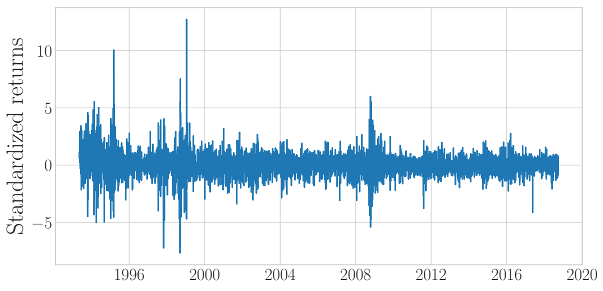
(a) N225



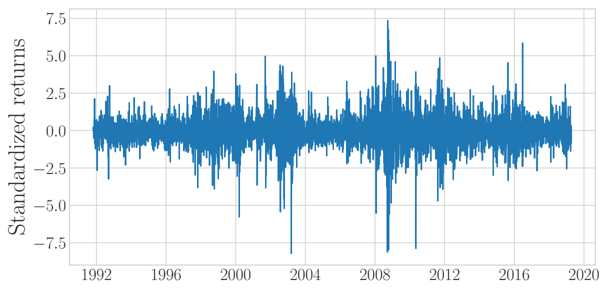
(b) SSE



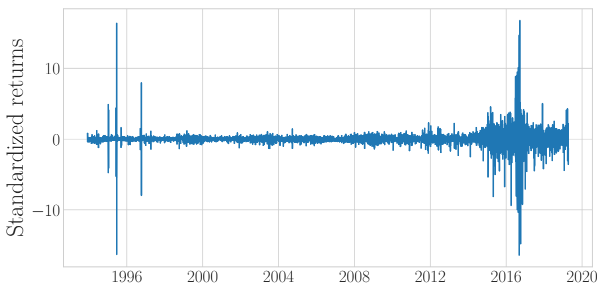
(c) BSESN



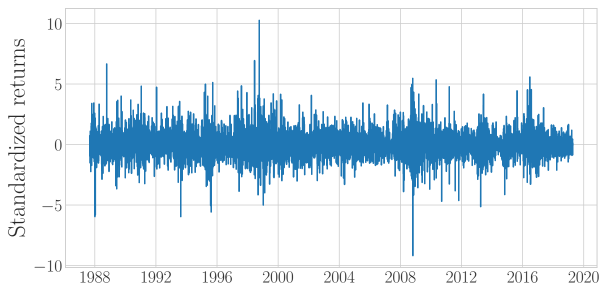
(d) BVSP



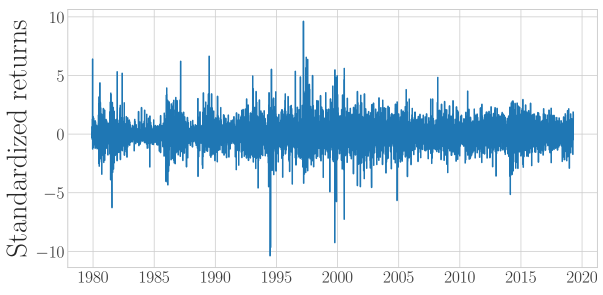
(e) BEL20



(f) BEOLO



(g) USDJPY



(h) KCc1

Figure A.3: Returns of the various assets

Assets	Absence of autocorrelation	Presence of volatility clustering
N225	76 %	91 %
SSE	79 %	68 %
BSESN	71 %	86 %
BVSP	93 %	90 %
BEL20	48 %	89 %
BEOLO	78 %	77 %
USDJPY	92 %	56 %
KCc1	99 %	57 %

Table A.2: Percentage of the samples that exhibit absence of autocorrelation and presence of volatility clustering according to the Ljung-Box test

Asset	Tail index
N225	3.86
SSE	3.55
BSESN	4.38
BVSP	3.53
BEL20	4.45
BEOLO	2.33
USDJPY	4.00
KCc1	4.27

Table A.1: Tail index of various assets

Asset	Average 0.1% drawdown	Average duration	Days inbetween
N225	-13.3	6.6	562
SSE	-11.9	6.7	572
BSESN	-13.5	7.7	581
BVSP	-14.0	6.1	524
BEL20	-14.3	6.1	536
BEOLO	-15.8	2.4	649
USDJPY	-11.4	6.3	515
KCc1	-9.7	6.9	520

Table A.3: Crash features of various assets

A.1 Comparing two models

Two models are generated, starting from the model trained on the synthetic data. The first model is not subjected to post-processing and is directly implemented on the real financial data. The second model undergoes first additional training on the real financial data's training set. On the second validation set, the performance of the two models is compared (figure A.4), with the first model being the blue line and the second model the green line. The red line is the baseline, this is the obtained precision-recall when a crash would be predicted at random. The performance of the second model is superior for low recall values whereas the first model' performance is more consistent overall. This work will continue with the first model as it has been shown to be more widely applicable,e.g. for higher recall values. Nevertheless, the result for the second model looks promising and we encourage further research to look more deeply into its performance.

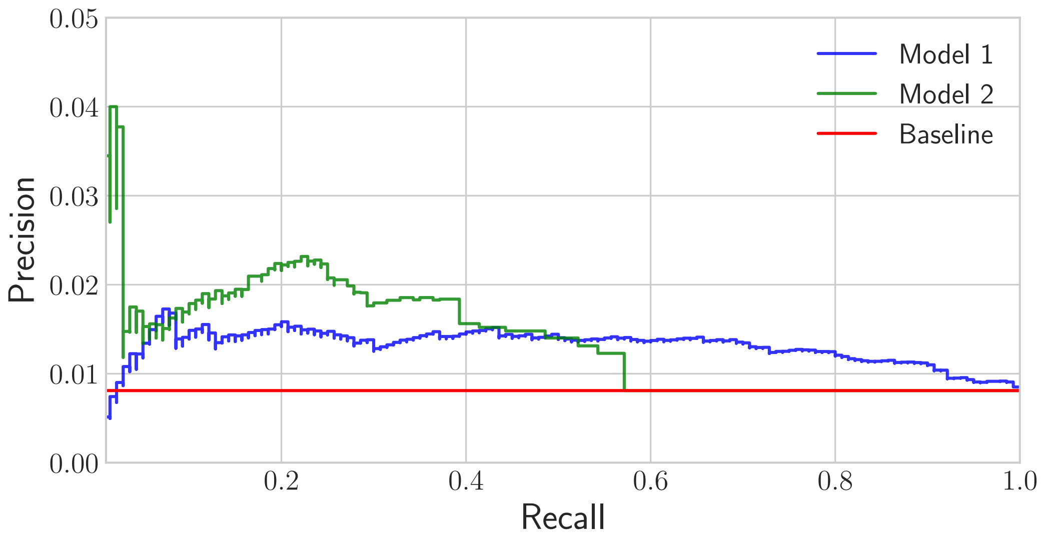
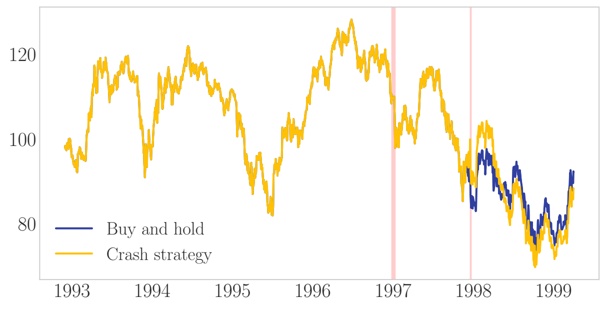
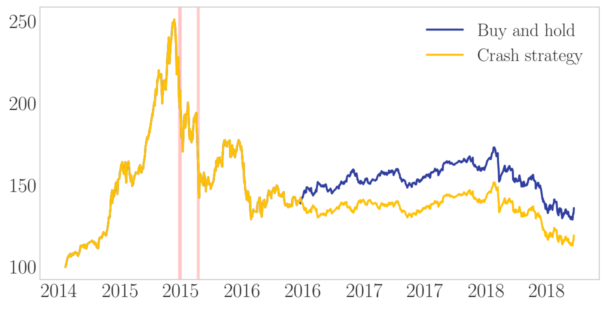


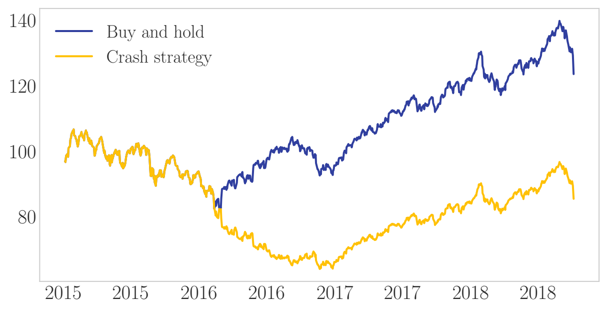
Figure A.4: Precision-recall on the validation set



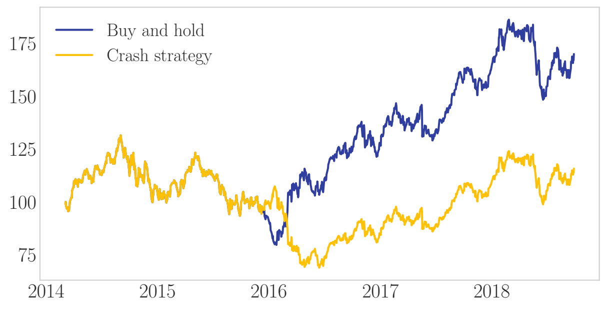
(a) N225



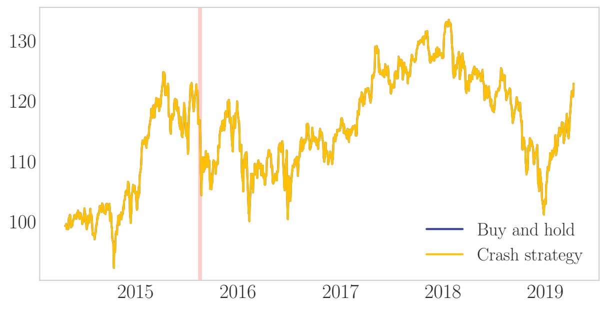
(b) SSE



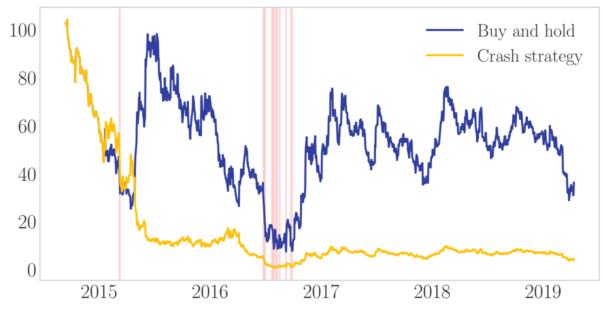
(c) BSESN



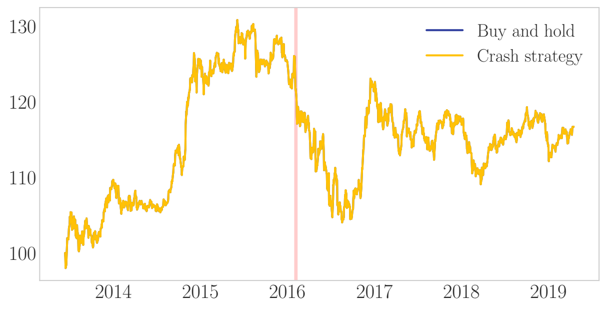
(d) BVSP



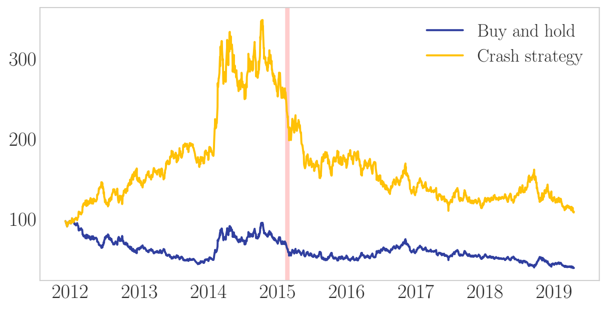
(e) BEL20



(f) BEOLO



(g) USDJPY



(h) KCc1

Figure A.5: Trading algorithm based on the model

Bibliography

Alstott, J., Bullmore, E., and Plenz, D. (2014). “powerlaw: A Python Package for Analysis of Heavy-Tailed Distributions”. In: *PLoS ONE* 9.1, e85777. DOI: 10.1371/journal.pone.0085777. arXiv: 1305.0215 [physics.data-an].

Arisue, H. and Tabata, K. (2001). “First Order Phase Transition of the Q-State Potts Model in Two Dimensions”. In: *Non-Perturbative Methods and Lattice QCD*, pp. 233–241. DOI: 10.1142/9789812811370_0024. arXiv: hep-lat/0007025 [hep-lat].

Aste, T. (2013). *Stylized facts of financial data.* URL: https://wiki.ucl.ac.uk/display/FinSci/Stylized%20facts%20of%20financial%20data.

Balanda, K. and Macgillivray, H. (1988). “Kurtosis: A Critical Review”. In: *The American Statistician* 42, pp. 111–119. DOI: 10.1080/00031305.1988.10475539.

Bar-Yam, Y. and Kantor, D. (2018). “A Mathematical Theory of Interpersonal Interactions and Group Behavior”. In: *arXiv e-prints*, arXiv:1812.11953, arXiv:1812.11953. arXiv: 1812.11953 [physics.soc-ph].

Bengio, Y., Simard, P., Frasconi, P., et al. (1994). “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2, pp. 157–166.

Blanchard, O. and Watson, M. (1982). *Bubbles, Rational Expectations and Financial Markets.* NBER Working Papers 0945. National Bureau of Economic Research, Inc. URL: https://EconPapers.repec.org/RePEc:nbr:nberwo:0945.

Bornholdt, S. (2001). “Expectation Bubbles in a Spin Model of Markets”. In: *International Journal of Modern Physics C* 12, pp. 667–674. DOI: 10.1142/S0129183101001845. arXiv: cond-mat/0105224 [cond-mat.stat-mech].

Bouchaud, J.-P., Matacz, A., and Potters, M. (2001). “Leverage effect in financial markets: The retarded volatility model”. In: *Physical review letters* 87.22, p. 228701.

Bouri, E., Gupta, R., and Roubaud, D. (2018). “Herding behaviour in cryptocurrencies”. In: *Finance Research Letters*. ISSN: 1544-6123. DOI: `https://doi.org/10.1016/j.frl.2018.07.008`. URL: `http://www.sciencedirect.com/science/article/pii/S1544612318303647`.

Bradley, A. P. (1997). “The use of the area under the ROC curve in the evaluation of machine learning algorithms”. In: *Pattern recognition* 30.7, pp. 1145–1159.

Brée, D. S., Challet, D., and Peirano, P. P. (2013). “Prediction accuracy and sloppiness of log-periodic functions”. In: *Quantitative Finance* 13.2, pp. 275–280.

Bree, D. S. and Lael Joseph, N. (2010). “Fitting the Log Periodic Power Law to financial crashes: a critical analysis”. In: *arXiv e-prints*, arXiv:1002.1010, arXiv:1002.1010. arXiv: `1002.1010 [q-fin.ST]`.

Buckland, M. and Gey, F. (1994). “The relationship between recall and precision”. In: *Journal of the American society for information science* 45.1, pp. 12–19.

Buda, M., Maki, A., and Mazurowski, M. A. (2017). “A systematic study of the class imbalance problem in convolutional neural networks”. In: *CoRR* abs/1710.05381. arXiv: `1710.05381`. URL: `http://arxiv.org/abs/1710.05381`.

Campbell, J. et al. (1997). *The Econometrics of Financial Markets*. Princeton University Press. ISBN: 9780691043012. URL: `https://books.google.be/books?id=lkeKhnqUHx8C`.

Chancellor, E. (1999). *Devil take the hindmost : a history of financial speculation / Edward Chancellor*. English. 1st ed. Farrar, Straus, Giroux New York, xiv, 386 p. cm. ISBN: 0374138583.

Chang, G. and Feigenbaum, J. (2006). “A Bayesian analysis of log-periodic precursors to financial crashes”. In: *Quantitative Finance* 6.1, pp. 15–36.

Charles P. Kindleberger, R. Z. A. (1978). *Manias, panics and Crashes*. Vol. 7. Palgrave Macmillan, pp. 38–53.

Chen, J., Hong, H., and Stein, J. C. (2002). “Breadth of Ownership and Stock Returns”. In: *Journal of Financial Economics* 66.November-December, pp. 171–205.

Cho, J. et al. (2015). "How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?" In: *arXiv preprint arXiv:1511.06348*.

Chollet, F. et al. (2015). *Keras*. `https://keras.io`.

Chordia, T., Roll, R., and Subrahmanyam, A. (2008). "Liquidity and market efficiency". In: *Journal of Financial Economics* 87.2, pp. 249–268.

Clauset, A., Shalizi, C. R., and Newman, M. E. (2009). "Power-law distributions in empirical data". In: *SIAM review* 51.4, pp. 661–703.

Cont, R. (2001a). "Empirical properties of asset returns: stylized facts and statistical issues". In: *Quantitative Finance* 1.2, pp. 223–236. DOI: `10.1080/713665670`. eprint: `https://doi.org/10.1080/713665670`. URL: `https://doi.org/10.1080/713665670`.

Cont, R. (2001b). "Empirical properties of asset returns: stylized facts and statistical issues". In:

Cruz, R. et al. (2016). "Tackling class imbalance with ranking". In: *2016 International joint conference on neural networks (IJCNN)*. IEEE, pp. 2182–2187.

Davis, J. and Goadrich, M. (2006). "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 233–240.

Delong, J. B., Stuart, J., and Marshall, A. (2009). "The Simplest Possible Behavioral Finance Bubble Model". In: *The Economist*, pp. 1–12. URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.186.834&rep=rep1&type=pdf`.

Depken, C. A., Hollans, H., and Swidler, S. (2011). "Flips, flops and foreclosures: anatomy of a real estate bubble". In: *Journal of Financial Economic Policy* 3.1, pp. 49–65. DOI: `10.1108/17576381111116759`. eprint: `https://doi.org/10.1108/17576381111116759`. URL: `https://doi.org/10.1108/17576381111116759`.

Drummond, C. and Holte, R. C. (2000). "Exploiting the Cost (In)Sensitivity of Decision Tree Splitting Criteria". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 239–246. ISBN: 1-55860-707-2. URL: `http://dl.acm.org/citation.cfm?id=645529.658143`.

Egan, J. P. (1975). "Signal detection theory and {ROC} analysis". In:

Enric, C. (2017). “Computational Physics”. In: pp. 34–35. URL: http://itf.fys.kuleuven.be/~enrico/Teaching/monte_carlo_2018.pdf.

Fama, E. F. (1965). “The Behavior of Stock-Market Prices”. In: *The Journal of Business* 38.1, pp. 34–105. ISSN: 00219398, 15375374. URL: http://www.jstor.org/stable/2350752.

Fan, J. and Yao, Q. (2017). *The Elements of Financial Econometrics*. The Elements of Financial Econometrics. Cambridge University Press, pp. 20–23. ISBN: 9781107191174. URL: https://books.google.be/books?id=7NOcDgAAQBAJ.

Fawcett, T. (2006). “An Introduction to ROC Analysis”. In: *Pattern Recogn. Lett.* 27.8, pp. 861–874. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2005.10.010. URL: http://dx.doi.org/10.1016/j.patrec.2005.10.010.

Filimonov, V. and Sornette, D. (2013). “A stable and robust calibration scheme of the log-periodic power law model”. In: *Physica A Statistical Mechanics and its Applications* 392, pp. 3698–3707. DOI: 10.1016/j.physa.2013.04.012. arXiv: 1108.0099 [q-fin.GN].

Forró, Z. (2015). “Detecting Bubbles in Financial Markets: Fundamental and Dynamical Approaches”. In: p. 95. DOI: 10.3929/ethz-a-010510872. URL: http://www.er.ethz.ch/content/dam/ethz/special-interest/mtec/chair-of-entrepreneurial-risks-dam/documents/dissertation/thesis%7B%5C_%7Dzalan%7B%5C_%7Dforro%7B%5C_%7Dfinal.pdf.

Geng, Y. and Luo, X. (2019). “Cost-sensitive convolutional neural networks for imbalanced time series classification”. In: *Intelligent Data Analysis* 23.2, pp. 357–370.

Geraskin, F. (2013). “Everything you always wanted to know about log-periodic power laws for bubble modeling but were afraid to ask”. In: *The European Journal Of Finance* 19.5, pp. 366–391.

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

Gürkaynak, R. S. (2008). “ECONOMETRIC TESTS OF ASSET PRICE BUBBLES: TAKING STOCK*”. In: *Journal of Economic Surveys* 22.1, pp. 166–186. DOI: 10.1111/j.1467-6419.2007.00530.x. eprint: https://onlinelibrary.wiley.com/

doi/pdf/10.1111/j.1467-6419.2007.00530.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-6419.2007.00530.x.

Harrison, J. M. and Kreps, D. M. (1978). "Speculative Investor Behavior in a Stock Market with Heterogeneous Expectations*". In: *The Quarterly Journal of Economics* 92.2, pp. 323–336. DOI: 10.2307/1884166. eprint: /oup/backfile/content_public/journal/qje/92/2/10.2307/1884166/2/92-2-323.pdf. URL: http://dx.doi.org/10.2307/1884166.

He, H. and Garcia, E. A. (2009). "Learning from Imbalanced Data". In: *IEEE Trans. on Knowl. and Data Eng.* 21.9, pp. 1263–1284. ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.239. URL: https://doi.org/10.1109/TKDE.2008.239.

He, K. et al. (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

Hochreiter, S. (1991). "Untersuchungen zu dynamischen neuronalen Netzen". In: *Diploma, Technische Universität München* 91.1.

Hochreiter, S. and Schmidhuber, J. (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Horák, J. and Smid, M. (2009). "On tails of stock returns: Estimation and comparison between stocks and markets". In: *Available at SSRN 1365229.*

Huang, W. et al. (2010). "Return Reversals, Idiosyncratic Risk, and Expected Returns". In: *The Review of Financial Studies* 23.1, pp. 147–168. DOI: 10.1093/rfs/hhp015. eprint: /oup/backfile/content_public/journal/rfs/23/1/10.1093_rfs_hhp015/1/hhp015.pdf. URL: http://dx.doi.org/10.1093/rfs/hhp015.

Jacobsson, E. (2009). "How to predict crashes in financial markets with the Log-Periodic Power Law". In: *Master diss., Department of Mathematical Statistics, Stockholm University.*

Japkowicz, N. and Stephen, S. (2002). "The Class Imbalance Problem: A Systematic Study". In: *Intell. Data Anal.* 6.5, pp. 429–449. ISSN: 1088-467X. URL: http://dl.acm.org/citation.cfm?id=1293951.1293954.

Johansen, A. and Sornette, D. (2002). "Endogenous versus Exogenous Crashes in Financial Markets". In: *arXiv e-prints*, cond-mat/0210509, cond–mat/0210509. arXiv: `cond-mat/0210509 [cond-mat.stat-mech]`.

Johansen, A. and Sornette, D. (2004). *Endogenous versus exogenous crashes in financial markets, in press in "Contemporary Issues in International Finance".*

Johansen, A. (2004). "Origin of crashes in three US stock markets: shocks and bubbles". In: *Physica A: Statistical Mechanics and its Applications* 338.1-2, pp. 135–142.

Johansen, A., Ledoit, O., and Sornette, D. (1998). "Crashes as Critical Points". In: *arXiv e-prints*, cond-mat/9810071, cond–mat/9810071. arXiv: `cond-mat/9810071 [cond-mat]`.

Johansen, A. and Sornette, D. (1999). "Critical Crashes". In: *arXiv e-prints*, cond-mat/9901035, cond–mat/9901035. arXiv: `cond-mat/9901035 [cond-mat.stat-mech]`.

– (2010). "Shocks, Crashes and Bubbles in Financial Markets". In: *Brussels Economic Review* 53.2, pp. 201–253. URL: `https://EconPapers.repec.org/RePEc:bxr:bxrceb:2013/80942`.

Johansen, A., Sornette, D., and Ledoit, O. (1999). "Predicting Financial Crashes Using Discrete Scale Invariance". In: *arXiv e-prints*, cond-mat/9903321, cond–mat/9903321. arXiv: `cond-mat/9903321 [cond-mat]`.

Karim, F. et al. (2018). "LSTM fully convolutional networks for time series classification". In: *IEEE Access* 6, pp. 1662–1669.

Keynes, J. M. (1936). *The General Theory of Employment, Interest and Money.* 14th edition, 1973. Macmillan. URL: `https://cas2.umkc.edu/economics/people/facultypages/kregel/courses/econ645/winter2011/generaltheory.pdf`.

Kingma, D. P. and Ba, J. (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980.*

Kinouchi, O. and Copelli, M. (2006). "Optimal dynamical range of excitable networks at criticality". In: *Nature Physics* 2, 348 EP -. URL: `https://doi.org/10.1038/nphys289`.

Kuramoto, Y. (1975). "International symposium on mathematical problems in theoretical physics". In: *Lecture notes in Physics* 30, p. 420.

Ljung, G. M. and Box, G. E. P. (1978). "On a Measure of Lack of Fit in Time Series Models". In: *Biometrika* 65.2, pp. 297–303. ISSN: 00063444. URL: http://www.jstor.org/stable/2335207.

Maimon, O. and Rokach, L. (2005). "Data mining and knowledge discovery handbook". In:

Mandelbrot, B. (1963). "The Variation of Certain Speculative Prices". In: *The Journal of Business* 36. URL: https://EconPapers.repec.org/RePEc:ucp:jnlbus:v:36:y:1963:p:394.

Martın Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: http://tensorflow.org/.

McClelland, J. L., Rumelhart, D. E., Group, P. R., et al. (1986). "Parallel distributed processing". In: *Explorations in the Microstructure of Cognition* 2, pp. 216–271.

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

McCulloch, W. S. and Pitts, W. (1943b). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

Miller, E. M. (1977). "Risk, Uncertainty, and Divergence of Opinion". In: *The Journal of Finance* 32.4, pp. 1151–1168. ISSN: 15406261. DOI: 10.1111/j.1540-6261.1977.tb03317.x. arXiv: 00368075.

Miller, M. and Modigliani, F. (1961). "Dividend Policy, Growth, and the Valuation of Shares". In: *The Journal of Business* 34. URL: https://EconPapers.repec.org/RePEc:ucp:jnlbus:v:34:y:1961:p:411.

Minsky, H. P. (1982). *Can "it" happen again? : essays on instability and finance / Hyman P. Minsky*. English. M.E. Sharpe Armonk, N.Y, xxiv, 301 p. : ISBN: 0873322134.

Nielsen, M. (2017). *Neural Networks and Deep Learning*. http://neuralnetworksanddeeplearning.com/.

Olah, C. (2015). *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Olver, F. W. et al. (2010). *NIST handbook of mathematical functions hardback and CD-ROM*. Cambridge university press.

Parens, R. and Bar-Yam, Y. (2016). "Step by Step to Peace in Syria". In: *arXiv e-prints*, arXiv:1602.06835, arXiv:1602.06835. arXiv: `1602.06835 [physics.soc-ph]`.

Potts, R. B. (1952). "Some generalized order-disorder transformations". In: *Mathematical proceedings of the cambridge philosophical society*. Vol. 48. 1. Cambridge University Press, pp. 106–109.

Rosenblatt, F. (1962). *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*. Spartan Books.

Shiller, R. (1981). "Do Stock Prices Move Too Much to be Justified by Subsequent Changes in Dividends?" In: *American Economic Review* 71.3, pp. 421–36. URL: `https://EconPapers.repec.org/RePEc:aea:aecrev:v:71:y:1981:i:3:p:421-36`.

– (2015). *Irrational Exuberance*. 3rd ed. Princeton University Press. URL: `https://EconPapers.repec.org/RePEc:pup:pbooks:10421`.

Shleifer, A. and Vishny, R. W. (1997). "The Limits of Arbitrage". In: *Journal of Finance* 52.1. Reprinted in Harold M. Shefrin, ed., Behavioral Finance, Edward Elgar Publishing Company, 2001. Reprinted in Richard Thaler, ed., Advances in Behavioral Finance Vol. II, Princeton University Press and Russell Sage Foundation, 2005., pp. 35–55.

Smith, L. N. (2018). "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay". In: *arXiv preprint arXiv:1803.09820*.

Sornette, D. (2003). *Why Stock Markets Crash - Critical Events in Complex Financial Systems*. Princeton University Press, p. 81.

Sornette, D. and Cauwels, P. (2014). "Financial Bubbles: Mechanisms and Diagnostics". In: 2. URL: `https://www.researchgate.net/publication/261475298_Financial_Bubbles_Mechanisms_and_Diagnostics`.

Sornette, D., Cauwels, P., and Smilyanov, G. (2017). "Can We Use Volatility to Diagnose Financial Bubbles? Lessons from 40 historical bubbles". en. In: Quantitative Finance and Economics. ISSN: 2573-0134.

Sornette, D. and Johansen, A. (2002). “Large stock market price drawdowns are outliers”. In: *Journal of Risk* 4. DOI: `10.21314/JOR.2002.058`. URL: `https://www.risk.net/journal-risk/2161095/large-stock-market-price-drawdowns-are-outliers`.

Sornette, D., Johansen, A., and Bouchaud, J.-P. (1996). “Stock Market Crashes, Precursors and Replicas”. In: *Journal de Physique I* 6.1, pp. 167–175. ISSN: 1155-4304. DOI: `10.1051/jp1:1996135`. URL: `http://dx.doi.org/10.1051/jp1:1996135`.

Sornette, D., Woodard, R., et al. (2013a). “Clarifications to questions and criticisms on the Johansen–Ledoit–Sornette financial bubble model”. In: *Physica A: Statistical Mechanics and its Applications* 392.19, pp. 4417–4428. DOI: `10.1016/j.physa.2013.05.0`. URL: `https://ideas.repec.org/a/eee/phsmap/v392y2013i19p4417-4428.html`.

– (2013b). “Clarifications to questions and criticisms on the Johansen–Ledoit–Sornette financial bubble model”. In: *Physica A: Statistical Mechanics and its Applications* 392.19, pp. 4417–4428.

Swets, J. A. (1988). “Measuring the accuracy of diagnostic systems”. In: *Science* 240.4857, pp. 1285–1293.

Taylor, S. J. (2011). *Asset price dynamics, volatility, and prediction.* Princeton university press.

Thurner, S., Farmer, J. D., and Geanakoplos, J. (2012). “Leverage causes fat tails and clustered volatility”. In: *Quantitative Finance* 12.5, pp. 695–707.

Tirole, J. (1985). “Asset Bubbles and Overlapping Generations”. In: *Econometrica* 53.6, p. 1499. ISSN: 00129682. DOI: `10.2307/1913232`. URL: `http://www.jstor.org/stable/1913232?origin=crossref`.

Warusawitharana, M. (2018). “Time-varying volatility and the power law distribution of stock returns”. In: *Journal of Empirical Finance* 49, pp. 123–141.

Weatherall, J. O. (2013). *The physics of wall street: a brief history of predicting the unpredictable.* Houghton Mifflin Harcourt.

Yang, Q. (2006). “Stock Bubbles, Theory and Estimation”. In: December. URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.426.9149&rep=rep1&type=pdf`.

Yichuan, J. (2014). *jd8001/LPPL*. URL: https://github.com/jd8001/LPPL.

Zhou, W.-X. and Sornette, D. (2007a). "Self-organizing Ising model of financial markets". In: *The European Physical Journal B* 55.2, pp. 175–181. ISSN: 1434-6036. DOI: 10.1140/epjb/e2006-00391-6. URL: https://doi.org/10.1140/epjb/e2006-00391-6.

Zhou, W.-X. and Sornette, D. (2007b). "Self-organizing Ising model of financial markets". In: *The European Physical Journal B* 55.2, pp. 175–181.