

UTILIZING MACHINE LEARNING FOR TRADING ALGORITHMS EXPLOITING THE TIME SERIES MOMENTUM ANOMALY

MARTIN ODENBRAND AND SEBASTIAN
SVENSSON BROMERT

Master's thesis
2019:E31



LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics


TYP AV DOKUMENT <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> Delrapport	<input type="checkbox"/> Kompendium <input type="checkbox"/> Rapport	DOKUMENTBETECKNING LUTFMS—3368—2019
--	---	--

INSTITUTION Matematikcentrum. Matematisk statistik, Lunds universitet, Box 118, 221 00 LUND
FÖRFATTARE Martin Odenbrand, Sebastian Svensson Bromert
DOKUMENTTITEL OCH UNDERTITEL Utilizing Machine Learning for Trading Algorithms Exploiting the Time Series Momentum Anomaly
SAMMANFATTNING <p>Momentum or trend following investing refers to trading strategies constructed around the idea that in financial markets, the current trend will, more often than not, prevail. In the context of asset prices, this means that previous returns or the price development of an asset is indicative of similar future returns and price development. While at odds with established theory such as the weaker form of market efficiency, as defined by the efficient market hypotheses, the pricing anomalies have proven robust enough for an industry of funds, using systematic trading, to rely heavily upon them. This thesis aims at building a profitable trading strategy around the momentum anomaly by using machine learning and common momentum indicators.</p> <p>The underlying assets will be futures contracts due to their frequent use in the industry and generally high liquidity. Following the exploration of different machine learning algorithms, Random forest was chosen and subsequently optimised on training data by cross-validation using the model evaluation metric Matthews Correlation coefficient. These fitted models were backtested in three ways and benchmarked against simple trading strategies as well as buy-and-hold strategies using several performance metrics. The final result indicates great performance during strong negative trends, such as the financial crises, and an ability to lessen drawdowns. However, the models ultimately fail to act as robust and profitable strategies over a longer time horizon.</p>
NYCKELORD Machine learning, time series momentum, moving average crossover, MACD, Hodrick-Prescott filter, random forest, pricing anomaly, computational finance
DOKUMENTTITEL OCH UNDERTITEL - SVENSK ÖVERSÄTTNING AV UTLÄNDSK ORIGINALTITEL Maskininlärning för finansiella handelsalgoritmer baserat på anomalin av momentum hos tidsserier

UTGIVNINGSDATUM år 2019 mån 6	ANTAL SID 101	SPRÅK <input type="checkbox"/> svenska <input checked="" type="checkbox"/> engelska <input type="checkbox"/> annat
---	-------------------------	--

ÖVRIGA BIBLIOGRAFISKA UPPGIFTER	ISSN 1404-6342
	ISBN
	2019:E31

I, the undersigned, being the copyright owner of the abstract, hereby grant to all reference source permission to publish and disseminate the abstract.

Signature  _____

Date 2019-06-13

Contents

1	Introduction	7
1.1	Problem	7
1.2	Objective	8
1.3	Scope	8
2	Financial markets and momentum strategies	9
2.1	Financial markets and investing	9
2.1.1	Speculators and short selling	9
2.1.2	Investment methodologies	9
2.2	Futures contracts	10
2.2.1	Futures contracts as a time series	10
2.3	Momentum - Early research and honourable mentions	11
2.4	Momentum - Modern research	11
2.4.1	Cross sectional momentum	11
2.4.2	Time series momentum	12
2.4.3	A swift comparison	12
3	Mathematical theory	14
3.1	Introduction to Machine Learning	14
3.1.1	Classification	14
3.2	Errors from machine learning	15
3.2.1	Balancing the model	15
3.3	Utilising cross-validation	16
3.4	Classifiers	17
3.4.1	K-nearest neighbours	17
3.4.2	Support Vector Machine (SVM)	18
3.4.2.1	Support vector classifier	18
3.4.2.2	Support vector machines	20
3.4.3	Naive Bayes	20
3.4.4	Logistic regression	21
3.5	Bagging	21
3.5.1	Bootstrap aggregation ("Bagging")	21
3.5.1.1	Variance	21
3.6	Random forest	22
3.6.1	The decision tree	22
3.6.2	The forest	23
3.7	Hyper-parameter optimisation	24
3.7.1	Grid Search using Cross-Validation	24
3.7.2	Randomised search using Cross-Validation	24
3.8	Features	24
3.8.1	Filters	24
3.8.1.1	Notation	25

3.8.2	Simple moving average	25
3.8.3	Exponential moving average	26
3.8.4	Moving average crossovers	26
3.8.5	MACD	27
3.8.6	Hodrick-Prescott filter	28
3.9	Feature selection	29
3.9.1	Regularised logistic regression	30
3.10	Stratified sampling	30
3.10.1	CUMSUM filter	30
3.11	Model evaluation scoring	31
3.11.1	Accuracy	31
3.11.2	F1-Score	32
3.11.3	Negative Log Loss	33
3.11.4	Matthews Correlation Coefficient	34
3.12	Backtesting	34
3.12.1	Calculating returns from futures	34
3.12.2	Traditional walk forward	35
3.12.3	Warm-up walk forward	35
3.12.4	Combinatorial backtesting with Cross-Validation	36
3.13	Benchmarking	37
3.13.1	Performance metrics	37
3.13.1.1	Maximum drawdown	37
3.13.1.2	Sharpe ratio	38
3.13.1.3	Sortino ratio	38
3.13.1.4	Gross return	38
4	Method	39
4.1	Data gathering	39
4.2	Features	39
4.2.1	Adapting the trading indicators	40
4.2.2	Choosing parameters	41
4.2.3	Moving average crossover	41
4.2.4	MACD	42
4.2.5	Hodrick Prescott	44
4.3	Label creation	44
4.4	Data analysis and preparation	44
4.4.1	Stratified sampling	44
4.4.2	Overlapping data	45
4.4.3	Training and test split	46
4.5	Feature selection	47
4.6	Model selection	47
4.6.1	Initial comparison	48
4.6.2	Holding period and threshold decision	48
4.6.3	Hyper-parameter optimisation	49
4.6.4	Initial test of final model	49
4.6.5	Feature importance	49
4.7	Model backtesting	49
4.7.1	Individual asset backtesting	49
4.7.1.1	Parameter choice	50
4.7.2	Consolidated asset backtesting	50
4.8	Benchmarking	50

5	Results	51
5.1	Feature selection	51
5.2	Model selection	51
5.2.1	Initial model comparison	51
5.2.2	Holdingperiod and threshold decision	52
5.2.3	Hyper-parameter optimisation	54
5.2.3.1	Support Vector Machine	55
5.2.3.2	Random forest	55
5.2.4	Initial test of final model	55
5.2.5	Feature importance	57
5.3	Backtesting and Benchmarking	58
5.3.1	Traditional Walk-Forward	59
5.3.2	Warm-up Walk-Forward Backtesting	62
5.3.3	Combinatorial	65
5.3.4	Consolidated training data	66
6	Discussion	68
6.1	Feature selection	68
6.2	Model selection	68
6.2.1	Support Vector Machine	68
6.2.2	Random forest	69
6.2.3	Model evaluation metrics	69
6.2.4	Holding period and threshold decision	69
6.3	Backtesting and benchmarking	70
6.3.1	More successful	70
6.3.1.1	Crisis identification	70
6.3.1.2	Lower drawdowns	70
6.3.1.3	Outperforming simple MA benchmarks	70
6.3.2	Less successful	70
6.3.2.1	Computational restraints	70
6.3.2.2	Traditional walk-forward	70
6.3.2.3	Generalise results	71
6.3.2.4	Method approach	71
6.3.3	Transaction costs	71
7	Conclusion	72
	Appendices	76

Abstract

Momentum or trend following investing refers to trading strategies constructed around the idea that in financial markets, the current trend will, more often than not, prevail. In the context of asset prices, this means that previous returns or the price development of an asset is indicative of similar future returns and price development. While at odds with established theory such as the weaker form of market efficiency, as defined by the efficient market hypothesis, the pricing anomalies have proven robust enough for an industry of funds, using systematic trading, to rely heavily upon them. This thesis aims at building a profitable trading strategy around the momentum anomaly by using machine learning and common momentum indicators. The underlying assets will be futures contracts due to their frequent use in the industry and generally high liquidity. Following the exploration of different machine learning algorithms, Random forest was chosen and subsequently optimised on training data by cross-validation using the model evaluation metric Matthews Correlation coefficient. These fitted models were backtested in three ways and benchmarked against simple trading strategies as well as buy-and-hold strategies using several performance metrics. The final result indicates great performance during strong negative trends, such as the financial crises, and an ability to lessen drawdowns. However, the models ultimately fail to act as robust and profitable strategies over a longer time horizon.

Summary - Utilizing machine learning for trading algorithms exploiting the Time series momentum anomaly

Martin Odenbrand, Sebastian Svensson Bromert

Momentum and trading

Momentum refers to trading strategies constructed around the idea that in financial markets, the current trend will, more often than not, prevail. In the context of asset prices, this means that previous returns of an asset is indicative of similar future returns.

In this thesis, 7 different futures contracts based on equity indices were used as underlying assets. Futures were chosen due to their frequent use in the industry. While there are differences in the pricing of futures contracts compared to, for instance equities, such characteristics did not have any impact on the results. As such, the novice reader may think of the underlying contract as a stock.

When attempting to capitalise on momentum, a trader will buy the asset when the price indicates an upward trend and short the asset when a downward trend is identified (when going short, the trader makes money when the asset price falls).

To identify upward and downward trends, it is common to apply filters to the price. A filter is a signal processing method which removes some unwanted component from the signal, in this case noise. The noise can be thought of small fluctuations in the price which obscures the larger trends. The filters incorporate a smoothing parameter which controls the amount of noise removed from the price series.

Figure 1.1 incorporates such filtered signals, by the use of a moving average, where the underlying asset is bought when the yellow line, the fast filter, goes above the red line, the slow filter, and sold when the opposite holds true. What differentiates the yellow line from the red one is the smoothing parameter by which the yellow line has a lower level of smoothing than the red one.

Data and machine learning

The work was conducted in three steps. First, different machine learning algorithms were evaluated, one was chosen and subsequently optimized.

Second, data for the algorithm was created by calculating the percentage difference between fast and slow filters, as exemplified by *Figure 1.1*. By using different parameter combinations for the fast and the slow



Figure 1: A simple trading strategy based on the crossover of two filtered signals represented by the yellow and the red lines.

filters as well as other filtering techniques, a total of 25 trading strategies were used to train the algorithm. Data before 2005 was used to train the models and data after 2004 was used for tests.

Finally, four different types of backtests were used to test the algorithm. The performance was determined by several performance metrics, including risk adjusted returns. These metrics were compared to those of buy-and-hold strategies as well as simple static trading strategies as illustrated in *Figure 1.1*.

Backtests and results

Following the evaluation of different machine learning algorithms, the Random forest was chosen and subsequently trained on the training data. The Random forest was optimized using Matthews correlation coefficient.

In the backtests, the results were somewhat ambiguous. However, the algorithms showed an ability to limit drawdowns, the maximum loss over the period 2005 to 2019, compared to the benchmarks. Furthermore, the algorithms often outperformed the simple trading strategies. Finally, the algorithms, in general, performed well during the financial crisis of 2008.

There was no coherent result across the 7 contracts. The failure of some algorithms to perform well was largely attributed to their susceptibility of being wrong at the wrong time. If the algorithm is wrong when there is a large move in the market, the resulting losses are hard to recuperate.

Acknowledgements

We would like to express our gratitude to our supervisor Johan Lindström at the division of Mathematical Statistics, Lund University, for taking the time to support us. Furthermore, we would like to extend our gratitude to the team at OQAM for their support in the trading related aspects of this endeavor.

Chapter 1

Introduction

Quantitative finance is a field of applied mathematics dedicated to the mathematical modeling of financial markets. Many problems in quantitative finance relates to the problem of determining future asset prices or, at the very least, the direction of future asset prices. The incentive for such research is quite obvious. An algorithm which consistently, over time, achieves a high risk adjusted return is very valuable, because around the algorithm, a profitable trading strategy can be built.

In financial markets, one of the participants often associated with quantitative finance are hedge funds. A quantitative hedge fund is a fund which relies on systematic strategies based on algorithms for its trading decisions. Conversely, a non-quantitative hedge fund base their trading strategy on fundamental analysis and the trading decisions are made by the fund managers. One of the supposed advantages of running a quantitative hedge fund is that, as the investment decision is made by the algorithm, the risk for non-rational human behaviour is limited

A significant part of the quantitative hedge funds are so called commodity trading advisors or CTAs. This relatively broad term refers to hedge funds that use futures contracts for investing. Another name for a CTA is a managed futures fund. CTAs implement a wide range of strategies, but historically, different momentum strategies have been an important component. [2]

The term momentum has held different meanings over time and is still somewhat ambiguous. As such, the background will include a review of how its theoretical foundation has evolved over time and its place in modern economic theory.

1.1 Problem

Papers written on the subject of systematic trading often rely on rules based decision making. Consider for instance the following strategy where one initiates a long position when a short term moving average ("MA") exceeds the long term moving average. And conversely initiates a short position when the opposite holds true.



Figure 1.1: A simple trading strategy based on the crossover of a 72-day and a 12-day MA. The underlying contract, Z_1 , is bought when the 12-day MA exceeds that of the 72-day. The contract is sold when the opposite holds true.

This type of simple strategy relies on the assumption that the parameter choice, here 12 days and 72 days, for the moving averages is a reasonable one. More specifically, the ideal strategy is responsive enough not to yield significant drawdowns in the case of a trend reversal yet it is not too sensitive to misinterpret temporary reversals as a change in the trend and thus incurring unnecessary transactions costs. As such, the parameter choice will be a balancing act between sensitivity and robustness. Furthermore, while a strategy is profitable over a certain time period, this most not hold true for other periods as financial markets are complex and tend to change.

This thesis will rely on filtering techniques as the one exemplified above. However, it differs by relying more heavily on the machine learning algorithm to decide which rules should be enforced and when. The features used by the machine learning algorithm will consist of filtered price signals differing by the filtering method and parameters. Furthermore, the trading rules will be slightly adapted to better suit the machine learning algorithm.

1.2 Objective

The goal is for the model to yield positive sustainable results which exceeds the more naive approach of trading on predefined rules as the one exemplified in Figure 1.1. The financial viability of the models will be evaluated by a range of metrics such as the risk adjusted return. Furthermore, historically, an important property exhibited by momentum strategies is strong performance when the majority of asset classes exhibit lacklustre performance. This behaviour makes the strategy a valuable addition to the portfolio and the prevalence of such behaviour will be included when forming the basis of evaluation.

Finally, the underlying idea is that the machine learning approach will create a dynamic model which still retains the positive quality of objectiveness inherent in systematic trading strategies.

1.3 Scope

Given the price data for a given asset, with the benefit of hindsight, it is easy to locate time periods of price development which can be construed as a trend. These time periods can span hours or decades which means it is important to specify over which time horizons the model is expected to search for trend. In this thesis, the model is constructed to seek trends spanning from a few days up to, but not exceeding, a year.

The work is limited to creating machine learning algorithms around individual futures contracts. The strategies will be evaluated on a stand-alone basis and not as part of a portfolio.

Chapter 2

Financial markets and momentum strategies

2.1 Financial markets and investing

Financial markets are essentially a market much like any other where different financial instruments are bought and sold. They serve to facilitate the movement of capital between different parties. On one side, there are investors who have capital and who seek interest on that capital in order for it to grow. On the other side, there are, for instance, companies who need the money in order to grow their business and whom are prepared to pay interest. A market emerges from this activity because many of the products that help facilitate this exchange of funds are publicly traded. The most common examples of this are equities and bonds which essentially constitutes publicly traded ownership in a company and publicly traded loans respectively. The fact that these publicly traded instruments have a price which fluctuates with supply and demand gives rise to another market participant, speculators.

2.1.1 Speculators and short selling

A speculator is an investor who invests in an asset with the hope of having the value of the asset increase. In this context, value does not have to be equivalent to price as the investor can utilise short selling. When an investor initiates a short position in an asset, the investor makes money when the asset price depreciates. This can be done in equity markets by borrowing the stock from an entity, selling it in the market at time t_1 , buying it back at a later point t_2 and returning it to the entity. The investor will now book a profit, or loss, equal to the price difference between t_1 and t_2 less the cost of borrowing the asset and the transaction cost. The concept of short selling is important when attempting to capture the momentum anomaly as prices can trend both up and down. Unfortunately, short selling equities is often costly and sometimes even prohibited. As such, the thesis will focus on a type of instrument more suitable for short selling called futures contracts.

2.1.2 Investment methodologies

As a prelude to the review of momentum research it is useful to mention the general types of analysis deployed by investors. In an investment decision process, fundamental analysis involves analysing the fundamental data associated with an asset in order to determine if the asset is under or over valued. One example could involve analysing the financial statements of a company in an equity investment. Technical analysis involves analysing the actual price chart of an asset in an attempt to find patterns which indicate the future direction price. Finally, quantitative analysis in finance involves the use of mathematics and statistics to determine the investment strategy. Quantitative analysis has grown with the rise of quantitative hedge funds and technical advancements enabling the processing of big data.

The question whether technical analysis works or not is a controversial subject. Some researchers have

found evidence that the techniques can yield positive risk adjusted returns [38] while others have not.[16] Quantitative analysis enjoys more acceptance compared to technical analysis. This is perhaps unsurprising as quantitative analysis is built on a foundation of statistics and mathematics as opposed to eyeballing a chart looking for patterns. Yet, when taking a scientific approach to technical analysis by studying time series generated by the trading rules, there is strong evidence that technical analysis seizes on similar investment opportunities often eked out in quantitative analysis. This topic will be revisited in the review of momentum research.

2.2 Futures contracts

A futures contract is a derivative which derives its price from an underlying asset. The contract involves the agreement between two parties where the buyer, at a certain point in time, is bound to buy the underlying asset for a predetermined price. The seller is also bound by the contract to sell the asset at the agreed upon time and price.

Futures contracts have been used since the 17th century and were until the middle of the 20th century limited to commodities as the underlying asset. These types of contracts are actually referred to as forwards. The difference between futures and a forward contracts is that the former is standardized and can be sold on an exchange while forward contracts are agreed upon privately. With a forward, a farmer could lock-in a price for their harvest before it was even sown and thus avoid falling prices if supply outweighed demand after the harvest. On the other side, the buyer could avoid taking the risk of increasing prices after harvest. While there is generally a winner in the agreement, futures contracts works as a risk mitigating tool limiting volatility. The middle of the 20th century saw the rise of financial futures contracts with currencies, interest rates and stock indices as underlying assets. Today most futures contracts are not settled by exchanging a harvest for money, but simply settling the difference in value at expiration.

The above example with the farmer is a case of limiting the downside risk, known as hedging. Unsurprisingly, speculation, as described above, is also prevalent in the futures market. However, one might wonder why a speculator would by a derivative instead of simply buying the underlying asset.

Futures contracts has several useful properties for trading. When buying a futures contract, the buyer only has to pay a fraction of the contract value known as margin and functions as a security for the other party. This means that the trader can use significant leverage when investing in futures. The trader has to increase the margin if the position moves against them to keep the margin at what is known as a maintenance level,

Futures markets are highly liquid which is an important property ensuring efficient pricing and enabling sizeable investments without moving the market.

Considering, for instance, equity index futures, it is generally much more convenient to sell a futures contract than short-selling a single stock. The latter is often prohibited or associated with high costs. As such, futures enables the speculator to easily take advantage of falling market prices.

2.2.1 Futures contracts as a time series

As earlier mentioned, futures contracts are associated with a settlement date referred to as the expiration date. Thus, a given asset will, over time, have a continuous sequence of futures contracts with different expiration dates. When analysing the price development of a futures contract it is desirable to combine these separate contracts into a continuous time series which resembles the price development, had an investor bought and held the contract. This can be achieved by assuming that the futures contracts are rolled over. The roll-over involves continuously holding the most liquid futures contract, selling just before expiration and buying the next futures contract. The contracts are assumed to be sold and bought at their respective closing prices.

2.3 Momentum - Early research and honourable mentions

Among the earlier references to momentum strategies can be found in "The great metropolis" by James Grant from 1938. In the book, Grant cites David Ricardo, a highly successful businessman who in the late 1700s and early 1800s amassed a great fortune by trading stocks and bonds on the London stock exchange. Mr Ricardo contributed his great success to three golden rules, of which two were, "*Cut short your losses*" and "*Let your profits run on*".[15] Mr Ricardos strategy was to sell as quickly as possible when the asset prices started to fall and to refrain from selling assets until the price clearly had started to move in the wrong direction. Another example of an early practitioner of trend following strategies was Jesse Livermore, one of the most famous traders of all time. In Livermore's biography, written by Edwin Lefèvre in 1923, the trader said "*the big money was not in the individual fluctuations but in the main movements that is, not in reading the tape but in sizing up the entire market and its trend*".[24]

Besides quotes from successful investors, there were also early papers on the subject. In "Some A Posteriori Probabilities in Stock Market Action" written in 1937 by Alfred Cowles and Herbert Jones, the authors, while analysing data from 1835 to 1935, find that stock prices tend to move in the direction they came from.[11] In the early days of momentum research, two books were released. While unrelated to momentum strategies, they had a significant impact on the direction of research efforts over the coming decades. Benjamin Graham and David Dodd published "Security Analysis" in 1934 [5] and the "Intelligent investor" in 1949.[14] Their seminal works were centered around fundamental analysis and value investing. Going forward, anything deemed not belonging in the realm of fundamental analysis was labeled technical analysis and was subsequently ignored. This was perhaps due to the difficulty of verifying findings in technical analysis or maybe it was simply dwarfed by the more legitimised fundamental analysis.

In 1965 Eugen Fama published his dissertation arguing for the random walk hypothesis. In the paper, which laid the foundation for what is known as the efficient market hypothesis, Fama describes how stock prices follow a random walk and cannot be predicted.[13] In 1970, Fama came to review his finding and expand on the idea of efficient markets by defining three levels of market efficiency: weak form, semi-strong form and strong form efficiency.[27]

In markets with strong form efficiency, the theory states that all information is incorporated in the share price and that no market participant can earn excess returns. Weak market efficiency on the other hand allows for excess returns, but only through the use of fundamental analysis. Technical analysis is said to fall short because of the random walk hypothesis. If prices follow a random walk and do not contain any additional information, investors have to look beyond the price chart for information, hence the support for fundamental analysis.

In the 70s and 80s, some research on the phenomenon of momentum was conducted. However, the results were often contradictory and it was hard to attribute the results to momentum in isolation as the disputed pricing anomalies still lacked widely accepted explanation.[12][10] It would not be until the 90s that research in momentum strategies started to pick up steam

2.4 Momentum - Modern research

In modern research, two types of pricing anomalies referred to as momentum is said to exist in financial markets. The emergence of the two pricing anomalies and their respective models are described individually before their relevance for this thesis is discussed.

2.4.1 Cross sectional momentum

In 1993 Jegadeesh and Titman wrote "Returns of buying winners and selling losers: Implications for stock market efficiency". The authors found evidence of what is known as **Cross sectional momentum** a **relative measure**

Their trading strategy is based on constructing a stock portfolio and updating parts of it on a monthly basis.

At the beginning of a given month t , the stocks are ranked from best to worst performers considering the past J months. The top decile D_9 is labeled "winners" and bottom decile D_1 is labeled "losers". Next, the stocks in D_9 are bought while the stocks belonging to D_1 are sold. The stocks are held for K months, the so called holding period, before they are sold. At any given month t , the positions initiated at time $t - K$ are closed which means that only $\frac{1}{K}$ of the portfolio is rebalanced each month. The trading strategy was conducted over 16 combinations of J and K where $J = \{3, 6, 9, 12\}$ and $K = \{3, 6, 9, 12\}$ months. This simple strategy yielded abnormal positive returns that could not be explained by common risk factors. As the title implies, the paper had implications for the idea of efficient markets and resulted in additional research efforts being directed towards momentum strategies.[22]

2.4.2 Time series momentum

The second type of momentum, **Time series momentum**, an **absolute measure**, came to prominence in 2012 with the publishing of the paper "Time series momentum".[32]

By running a regression on the lagged values of returns, the authors found significant predictive power in returns lagged up to 12 months. The regression was run on individual futures contracts and aggregated across contracts after adjusting for volatility. The observations were made for all 58 futures contracts included in the study which were selected across asset classes, including equities, currencies, commodities and bonds. Furthermore, the authors constructed a trading strategy where a long position is entered if an asset s has had a positive return over the past 12 months and a short position if the return has been negative.

$$r_{t,t+1}^{TSMOM,s} = \text{sign}(r_{t-12,t})r_{t,t+1}^s \quad (2.1)$$

A diversified portfolio consisting of asset classes mentioned is shown to yield significant abnormal risk adjusted returns over the period 1985 to 2010.

Further research validated the findings and expanded on the idea of Time series momentum, ensuring its robustness. In a "Century of Evidence on Trend-Following Investing"[21] the authors conducted similar tests to those in "Time series momentum".[32] However, the time frame was extended to a century and a larger selection of assets were included. This larger study found similar results to that of the of "Time series momentum".[32]

By now, it had become generally accepted that asset returns carried information and the literature on the topic became an important part of the accepted, and ever growing, field of quantitative finance. However, in practice, Time series momentum strategies often relied on filtering techniques, such as moving averages, based on asset prices and not returns. These investment strategies, based on price, were often implemented as a tool in technical analysis. While returns are derived from price and the two should arguably be linked, the price based approach was labeled technical analysis and did not enjoy the same acceptance as the return based strategies deemed worthy of being labeled quantitative analysis. This despite recognised papers being written on the subject as early as 1992. [8]

In recent years there has been a convergence in the theory based on returns and those based on price. In "Time-Series Momentum versus Moving Average Trading Rules", a paper written in 2012, the authors show that the trading signal from returns based time series momentum is related to a change in direction of a moving average based on price. [29] Similarly, the idea of using moving average crossovers as a trading signal has, in several papers, shown to yield similar results to that of returns based Time series momentum. [9][25][4]

2.4.3 A swift comparison

Both cross sectional momentum and time series momentum has proven to be effective trading strategies in the past, yielding significant abnormal risk adjusted returns. Nowadays, most research efforts are directed towards Time series momentum because it has shown to be more profitable historically. One of the practical issues with cross sectional momentum is that it is a relative measure. In a trading scenario, this means that the strategy have to take long positions and short positions in assets no matter the market conditions. This

proves problematic during the periods when all assets fall or rise in tandem.[6]

In this thesis, when approaching the problem of constructing a trading algorithm, the main tool will be machine learning algorithms. Machine learning is closely related to pattern recognition which is the fundamental skill used by practitioners of technical analysis and a prominent component in quantitative analysis. The aspect of pattern recognition is more pronounced in the theory of Time series momentum compared to cross sectional momentum. Thus, the former appears to be a more suitable choice when utilising machine learning. The thesis will solely focus on Time series momentum which hereafter will be referred to as simply momentum.

Chapter 3

Mathematical theory

The development of a trading algorithm involves several steps, each with its own challenges and theoretical foundation.

Machine learning is a central aspect in the project. There are many different machine learning algorithms to choose from, where some are more suitable than others depending on the problem at hand. In this chapter, after describing the problem in the context of machine learning 3.1, relevant theory is established for the algorithms that were tested 3.43.6. Furthermore, techniques to help address problems and increase the potential performance of the algorithms are covered.

The algorithm needs data, or features, to train on in order to make predictions for the trading strategy. These features are selected to capture the time series momentum prevalent in financial markets. Here, the theory used when extracting the features and the reasoning behind them is established.3.8.1

The model needs to be evaluated in order to assess its effectiveness. This involves both traditional model evaluation metrics for machine learning algorithms 3.11 and backtesting 3.12 to assess the monetary value generated, or lost, by the algorithm. These concepts and the implications of using time series data when backtesting is also covered in this chapter.

3.1 Introduction to Machine Learning

Machine learning is the use of statistical models and algorithms together with computer systems to perform a specific task while not giving explicit instructions to the system on how to solve said task. This thesis focuses on the type of algorithms known as supervised learning algorithms. These algorithms are trained by supplying a set of features together with the correct predictions given those features. The two components are said to make up a training set. To assess the performance of the algorithm, there is also a test set of data which the algorithm has not yet seen. After training the algorithm, it is challenged to make predictions given the features in the test set. Those predictions are subsequently compared to the correct ones.

3.1.1 Classification

Supervised learning problems can be divided into two types of problems, regression and classification problems. A regression problem involves predicting a continuous value while a classification algorithm aims to predict a limited number of classes.

An algorithm which could accurately predict the future asset returns over a given time period would be the holy grail in a trading scenario. However, to the authors knowledge, this has yet to be accomplished by anyone. Here, the issue will be approached as a classification problem aiming to predict the general direction of asset prices. The definition is further described in section 4.3.

3.2 Errors from machine learning

As mentioned, there are many different machine learning algorithms. In order to evaluate an algorithm, it is important to understand the fundamental way in which it operates. In binary classification problems the goal is to predict an outcome $y = G(x)$ where $G(x) = \text{sign}[f(x) + \epsilon]$ and ϵ is white noise, i.e. $E[\epsilon_i] = 0$ and $V[\epsilon_i] = \sigma_\epsilon^2$. This is achieved by estimating a function $\hat{f}(x)$ and minimising the mean squared error, which can be decomposed as

$$E[(y_i - \hat{f}[x_i])^2] = \underbrace{\left(E[\hat{f}[x_i] - f[x_i]]\right)^2}_{\text{bias}} + \underbrace{V[\hat{f}[x_i]]}_{\text{variance}} + \underbrace{\sigma_\epsilon^2}_{\text{noise}} \quad (3.1)$$

Note that the equation 3.1 holds true in the general case of machine learning problems.

When bias is high, model is typically suffering from underfitting. The algorithm has failed to recognise patterns between the features and outcomes, thus generating inaccurate predictions.

Overfitting is a result of a too precise model, where the algorithm accounts for too many parameters in combination with high importance. Such models tend to have high variance since a small change in parameter values yields significantly different results. [26]

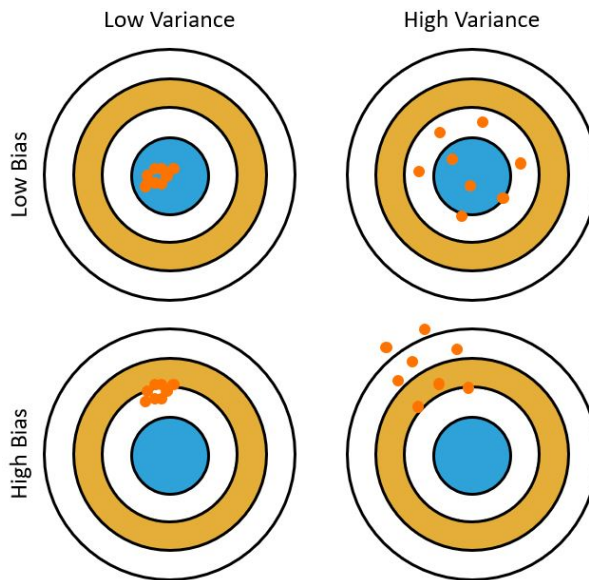


Figure 3.1: Visualising the implications of bias and variance

3.2.1 Balancing the model

Consequently the goal is to keep both bias and variance low, which is not always an easy task. As such, when machine learning is applied to a problem, a significant part of the task involves selecting the correct model and tuning its parameters for optimal results. A higher complexity yields a lower bias but also a higher variance and vice versa.[18]

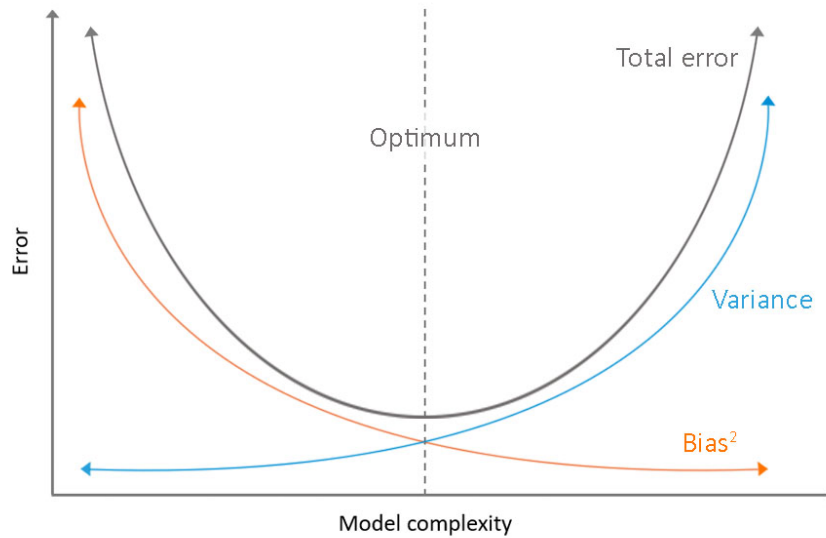


Figure 3.2: The error from bias typically decreases as model complexity increases while the error from variance increases

3.3 Utilising cross-validation

Imagine having daily price data for a given asset spanning 10 years. One way to approach the problem of creating and testing a model is to train the model on data from the first 7 years and to test the model on data from the latter 3 years. While this is a perfectly reasonable method, cross-validation provides an alternative approach.

Generally, the more data a model can utilise for training, the better it will be at predicting future outcomes. This makes intuitive sense as the model becomes accustomed to a richer set of scenarios. This is especially important in the context of the problem at hand. While it varies across contracts, price data for futures contracts is somewhat limited. Furthermore, financial markets and as an extension futures prices display very different behaviour over time reflecting the health of the underlying economy. By using K-fold cross-validation, training and testing the model can be done across the entire data set maximising the number of the scenarios the model is exposed to. The data split is performed such that:

1. The data sets is partitioned into k folds.
2. For all subsets $i = 1, \dots, k$
 - (a) Train the machine learning algorithm on all subsets except subset i .
 - (b) Test the fitted model on subset i .

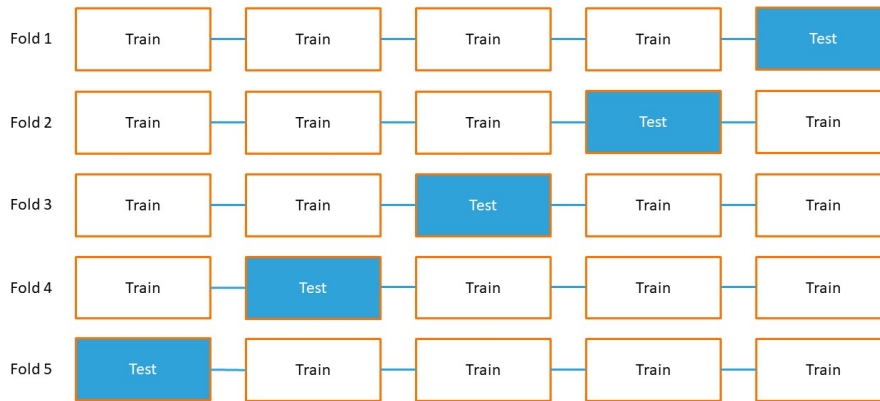


Figure 3.3: K-fold split with $k = 5$

Apart from efficient use of data, cross-validation will be used to choose the parameters supplied to the different models. The optimal parameters for a given model varies depending on which data it is trained. By using cross-validation, the parameter choice can be based on the diverse economic environments prevalent at different times in the time series. This is further described in 3.7.1.

3.4 Classifiers

As mentioned, there are many algorithms for classification problems in machine learning. A subset of these were tested before one was chosen. This section describes the theory underpinning the subset of algorithms tested.

3.4.1 K-nearest neighbours

K-nearest neighbours or KNN is a non-parametric learning algorithm. KNN constructs a database where the observations in the training data have been categorised into classes. This can be visualised in the case of two features.

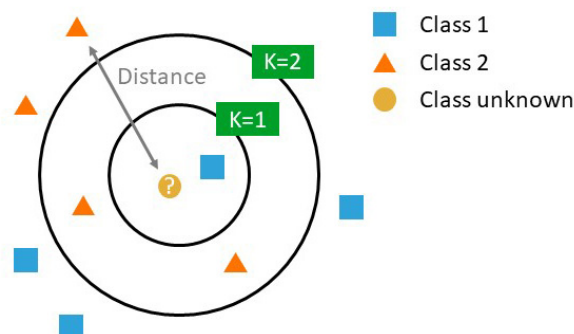


Figure 3.4: Visualising K-nearest neighbours assuming two classes

When classifying an observation, the classifier studies the features and locates the nearest neighbours in the training set. The closeness of neighbours is based on the similarity of the features associated with the

observation. What is "near" is commonly determined by the euclidean distance between the features of the observations.

The K in KNN refers to the number of neighbours that should be taken into account when classifying an unknown observation. Consider figure 3.4, if $K = 1$ the observation would be classified as belonging to class 1. However, if $K = 3$ the number of class 2 neighbours would exceed the number of class 1 neighbours and the observation would be classified as belonging to class 2. This is under the assumption of equally weighted voting rights in the neighbourhood. The voting rights can be weighted by distance to the target observation.

3.4.2 Support Vector Machine (SVM)

The Support Vector Machine ("SVM") is a machine learning algorithm which utilises hyperplanes to separate the features into different classes. This is achieved by creating nonlinear boundaries via the construction of linear boundaries in a large, transformed version of the feature space.

3.4.2.1 Support vector classifier

In the categorical case, the training data consist of N pairs $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$ with $x_i \in \mathbb{R}$ and $y_i \in \{-1, 1\}$. The hyperplane is then defined by

$$x : f(x) = x^T \beta + \beta_0 = 0, \quad (3.2)$$

where β is a unit vector: $\|\beta\| = 1$. To introduce classification a sign function is induced from $f(x)$

$$G(x) = \text{sign}[x^T \beta + \beta_0]. \quad (3.3)$$

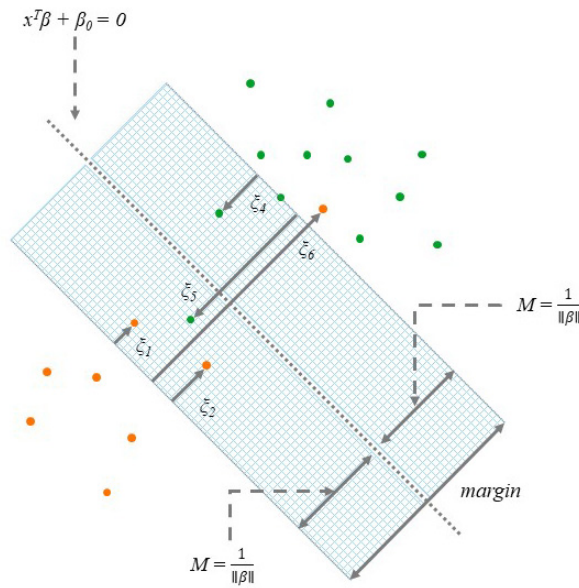


Figure 3.5: Support vector visualisation where points (ϵ_i) are overlapping.

The margin M is maximised subject to the budget constraint $\sum \epsilon_i \leq \text{constant}$. The optimisation problem becomes

$$\begin{aligned} & \max_{\beta, \beta_0, \|\beta\|=1} M \\ & \text{subject to } y_i(x^T \beta + \beta_0) \geq M(1 - \epsilon_i), \end{aligned} \quad (3.4)$$

$\forall i, \epsilon_i \geq 0, \sum_{i=1}^N \epsilon_i \leq \text{constant}$. By dropping the norm constraint on β and defining $M = 1/\|\beta\|$, Equation 3.78 can be written equivalent as

$$\min \|\beta\| \quad \text{subject to} \begin{cases} y_i(x_i^T \beta + \beta_0) \geq 1 - \epsilon_i & \forall i, \\ \epsilon_i \geq 0, \sum_{i=1}^N \epsilon_i \leq \text{constant} \end{cases} \quad (3.5)$$

Since the inequality constraints are linear and the problem is quadratic it can be solved using quadratic programming with Lagrange multipliers. To make computations more efficient, the equation is, once again, rewritten as:

$$\begin{aligned} \min_{\beta, \beta_0, \epsilon_i} \quad & \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N \epsilon_i \\ \text{subject to} \quad & \epsilon_i \geq 0, y_i(x_i^T \beta + \beta_0) \geq 1 - \epsilon_i \quad \forall \end{aligned} \quad (3.6)$$

where C is a cost parameter replacing *constant*. The Lagrange primal function becomes

$$L_P = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N \epsilon_i - \sum_{i=1}^N \alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \epsilon_i)] - \sum_{i=1}^N \mu_i \epsilon_i \quad (3.7)$$

which is minimised w.r.t $\beta, \beta_0, \epsilon_i$. Solving via derivatives equal to zero yields

$$\beta = \sum_{i=1}^N \alpha_i y_i x_i, \quad (3.8)$$

$$0 = \sum_{i=1}^N \alpha_i y_i, \quad (3.9)$$

$$\alpha_i = C - \mu_i, \quad \forall i, \quad (3.10)$$

with constraints $\alpha_i, \mu_i, \epsilon_i \geq 0 \quad \forall i$ The Lagrangian dual objective function becomes

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j x_i^T x_j, \quad (3.11)$$

The Karush-Kuhn-Tucker condition adds the constraints

$$\alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \epsilon_i)] = 0, \quad (3.12)$$

$$\mu_i \epsilon_i = 0. \quad (3.13)$$

$$y_i(x_i^T \beta + \beta_0) - (1 - \epsilon_i) \geq 0, \quad (3.14)$$

Equations 3.8-3.14 uniquely characterises the solution to the primal and dual problem. Using this, a solution can be found by maximising the dual problem function 3.11 Note that from 3.8 one recognise that the solutions take the form:

$$\hat{\beta} = \sum_{i=1}^N \hat{\alpha}_i y_i x_i, \quad (3.15)$$

Due to equation 3.12 α_i is $\neq 0$ only when the boundary 3.14 is an equality. The observations are called the support vectors, which has given the method its name. For points that does not exceed the margin i.e. $\epsilon_i = 0$, β_0 can be estimated. Finally, the binary decision function $G(x)$ can be estimated as

$$\hat{G}(x) = \text{sign}[\hat{f}(x)] = \text{sign}[x^T \hat{\beta} + \hat{\beta}_0] \quad (3.16)$$

3.4.2.2 Support vector machines

The linear support vector methodology explained above is easily transferred to the nonlinear case by using basis functions $h(x_i) = (h_1(x_i), h_2(x_i), \dots, h_M(x_i))$ instead of using points $(x_1), (x_2), \dots, (x_N)$. The hyperplanes are now replaced by the nonlinear function $\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0$ while the classifier function $\hat{G}(x)$ stays the same.

Using inner products $\langle h_i(x), h_j(x) \rangle$ the optimisation problem 3.78 can be solved, without specifying the transformation $h(x)$, by using a kernel function

$$K(x, x') = \langle h(x), h(x') \rangle \quad (3.17)$$

Three examples of kernel functions are:

$$\begin{aligned} \text{dth-Degree polynomial : } K(x, x') &= (1 + \langle x, x' \rangle)^d \\ \text{Radial basis : } K(x, x') &= \exp(-\gamma \|x - x'\|^2) \\ \text{Neural network : } K(x, x') &= \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2). \end{aligned} \quad (3.18)$$

The Lagrangian dual function to be solved has the form

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle h_i(x), h_j(x) \rangle, \quad (3.19)$$

$$(3.20)$$

The solution function $f(x)$ can be expressed as

$$\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0 \quad (3.21)$$

$$= \sum_{i=1}^N \hat{\alpha}_i y_i \langle h(x), h_i(x) \rangle + \hat{\beta}_0 \quad (3.22)$$

$$= \sum_{i=1}^N \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0. \quad (3.23)$$

3.4.3 Naive Bayes

A Naive Bayes classifier is a machine learning model based on Bayes' theorem with an independence assumptions between predictors. This assumption is the reason the technique is referred to as "Naive". Bayes' theorem is

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \quad (3.24)$$

where $X = (x_1, x_2, x_3, \dots, x_n)$ represent a vector of features values for a given observation and y represents a class. As such, the equation 3.24 can be rewritten as:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)P(y)}{P(x_1)P(x_2)\dots P(x_n)} \quad (3.25)$$

where $P(x_i|y)$ is defined as the probability of a feature value x_i for an observation with class label y .

For all observations in the feature set, the denominator in 3.25 remains the same and a proportionality can be introduced.

$$P(y|X_1, x_2, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i|y) \quad (3.26)$$

The class is predicted by finding the most likely candidate y across the set of classes:

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i|y) \quad (3.27)$$

3.4.4 Logistic regression

Similarly to Naive Bayes, the logistic regression is named after a central concept, the logistic function. The logistic function takes the form of an S-shaped curve 3.6 that maps real-valued numbers to a value between 0 and 1.

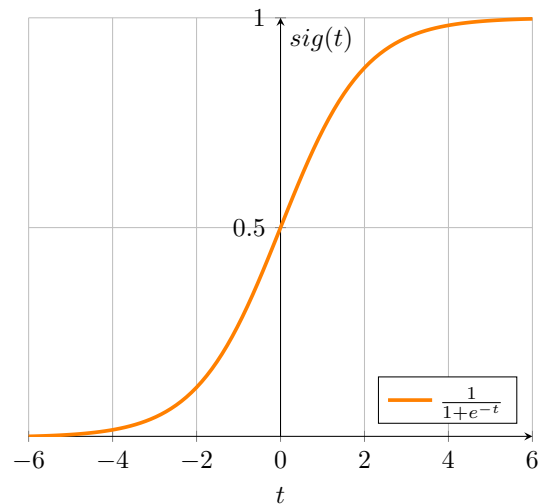


Figure 3.6: The logistic function

Using the notation as introduced in section 3.4.3, the conditional probability of a label given a set of features can be expressed as:

$$P(y|X) = \frac{1}{1 + e^{-f(x)}} \quad (3.28)$$

where $f(x)$ is a linear function of the features and their corresponding weights:

$$f(x) = w_1x_1 + w_2x_2 + \dots + w_nx_n \quad (3.29)$$

3.5 Bagging

3.5.1 Bootstrap aggregation ("Bagging")

As discussed in the previous section, overfitting is an issue strongly related to variance, when dealing with classifiers. A bagging method can help reduce variance by so called majority voting. The proportion of classification between several different estimators form a probability for an observation belonging to a given class. This is done by generating N data sets by random sampling with replacement and fitting one estimator for each data set. The N estimators then classes are predicted by a majority vote. It is important that each estimator gives class-probability rather than the classification itself.

Consider for instance a simple binary classification problem where the true probability is $p_1(x) = 0.75$. If each of the estimators accurately predict a 1 then $p_1(x) = 1$, which is incorrect.

3.5.1.1 Variance

The variance reducing property of bagging can be further reinforced by how the sampling from data sets is conducted. This is because the variance depends on the number of bagged estimators (N), the average variance of a single estimator's prediction ($\bar{\sigma}$) and the average correlation among their respective forecasts ($\bar{\rho}$).

$$V[bagging] = \bar{\sigma}^2 \left(\bar{\rho} + \frac{1 - \bar{\rho}}{N} \right) \quad (3.30)$$

Clearly, bagging becomes more efficient as estimators become less correlated. When $\bar{\rho} \rightarrow 1$, $V[\text{bagging}] \rightarrow \bar{\sigma}^2$ and when $\bar{\rho} \rightarrow 0$, $V[\text{bagging}] \rightarrow \bar{\sigma}^2 \left(\frac{1}{N}\right)$. When dealing with financial data, or time-series in general, non-randomised data sets tend to have correlated data. This is because a feature x_t at a given time t usually carries some information that is present in x_{t+1} and x_{t-1} as well. Thus, tools for reducing correlation in the training sets improves the variance reduction from bagging.

3.6 Random forest

Random forest is an ensemble method, which means that it aggregates the results from a ensemble of estimations, in this case simple decision trees. Random forest relies on Bagging as described in 3.5.1. In order to fully understand the dynamics of a Random forest one has to grasp the concepts of a decision tree.

3.6.1 The decision tree

A decision tree is made up of internal nodes and end nodes, often referred to as leaves. In the case of binary classification, the decision tree splits nodes into two. True/false or $\{0, 1\}$. This is repeated for each node until the algorithm reaches a leaf, the end point, which is assigned a single class, true or false. Building the tree is equivalent to the learning process of the decision tree. Now, the following questions arise, how does the split work and when are nodes split into two and when are they converted into leaves? To clarify this there are a couple of aspects to consider:

- Node impurity
- Minimum sample split
- Minimum sample leaf
- Max depth

Node impurity

First of all, there needs to be some measurement for evaluating the different splits in order for the model to make decisions and build the tree. Consider a node m , representing a region R_m , with N_m observations. Let:

$$\hat{p}_{m,k} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k), \quad (3.31)$$

be the proportion of class k observations in node m . In a classification problem there are three commonly used impurity measurements: Missclassification error, Gini index and Cross-entropy. This thesis will only focus on the Gini index which is:

$$\text{Gini index} : \sum_{k \neq k'} \hat{p}_{m,k} \hat{p}_{m,k'} = \sum_{k=1}^K \hat{p}_{m,k} (1 - \hat{p}_{m,k}) \quad (3.32)$$

In the binary case and if p is the proportion of the negative label the Gini index is:

$$\text{Binary Gini index} : 2p(1 - p) \quad (3.33)$$

However, when calculating the impurity measure there are some restrictions on how the algorithm divides the data.

Minimum sample split

First of all there is a restriction of how many observations there need to be in each of the two splits. As an example consider a data set of 1000 observations. If the minimum sample split is set to 200, the first

split can not be $[195,805]$ nor $[810,190]$, but the split $[200,800]$ is acceptable. This applies regardless of the impurity of $[195,805]$ and $[810,190]$.

Minimum sample leaf

As explained above, a leaf is created when an end node is reached, and a class label is assigned. The minimum sample leaf sets a restriction on how many observations there needs to be in a leaf. Building on the example above, the minimum sample leaf is assumed to be 100. This means that the two nodes, with 200 and 800 observations each, can not be split into smaller nodes than 100. For example, the two nodes could be split into $[100,100]$ and $[150,650]$ respectively.

Max depth

The final aspect to consider is the maximum depth of the tree. This controls the maximum number of levels the tree can build and indirectly controls the maximum amount of end nodes that can be created. Once again consider the example above. If the max depth is 2, no more splits are allowed after the split $[100,100]$ and $[150,650]$.

Financial example

A simple example in a financial setting. Consider a binary classifier $\{0, 1\}$ on 1000 observations with two features: stock price, $X_1(t) = C$ SEK and volatility, $X_2(t)$. The maximum depth is set to 3, minimum sample leaf to 100 and minimum sample split to 200. See figure 3.7.

By partitioning over all the possible combinations of stock prices and observations in the training set, the decision tree creates two groups (or nodes) $X_1(t) < 50$, 900 observations and $X_1(t) > 50$, 100 observations. In this example, it means that if a stock has a price of over 50 SEK the tree suggests it belongs to class 0. If the stock price is below 50 SEK, the investigation continues. When a stock price is below 50 SEK it might be that the volatility $X_2(t)$ is an important feature. The tree finds that for high volatility (>0.025) when stock price is less than 50 SEK the class is likely to be 1. Thus splitting this node into $X_1(t) < 50, X_2(t) > 0.025$, 100 observations and $X_1(t) < 50, X_2(t) < 0.025$, 800 observations.

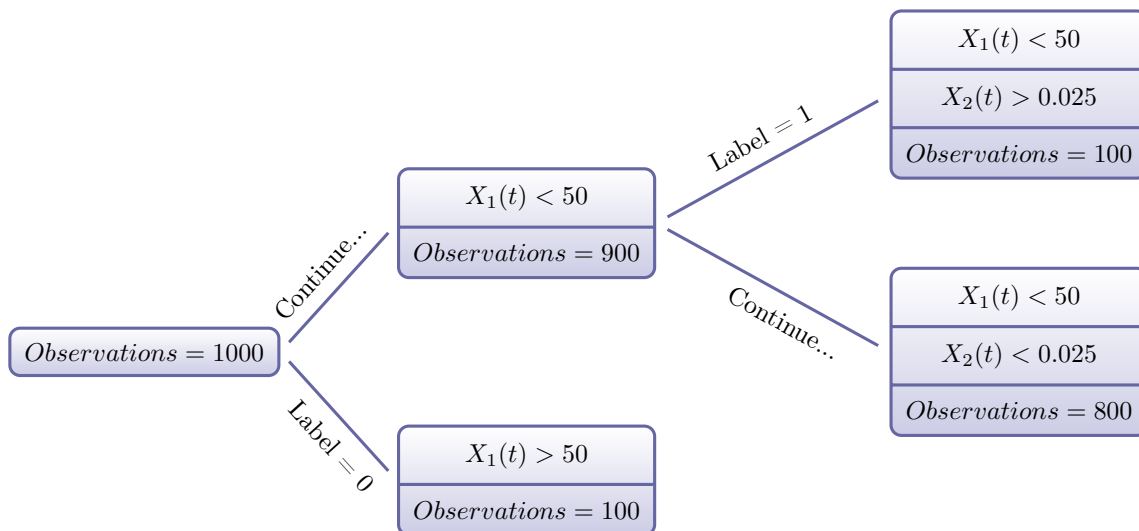


Figure 3.7: An example of a decision tree

3.6.2 The forest

The problem with using a single classification tree is its susceptibility to overfitting. As the name implies, the forest relies on combining many decision trees. A key point is that the individual trees only has access to a subset of the features and data points. This divergence in training data across the trees leads to diversity

in the forest and more robust predictions. The Random forest can be seen to take advantage of the "Wisdom of the crowd". When the desired number of trees are populated, the class prediction is decided by majority voting. As the trees are grown independently multi-core processing can be used in a simultaneous growth process.

3.7 Hyper-parameter optimisation

As both SVM and Random forest depend on parameter values it is essential to choose these with care and to analyse the implications of different choices. Using the earlier explained cross-validation method in combination with grid search one can efficiently compare the errors from different model combinations and find the optimal parameter values.

3.7.1 Grid Search using Cross-Validation

A grid search creates the partitions from ranges of parameter values where each partition is evaluated using cross-validation. Iterating over all the partitions returns a combination of parameter values that yield the largest average evaluation score over the entire training set. This approach does not account for the fact that some combination of parameter values might be more suitable for some financial regimes than others, it simply returns the parameters associated with the highest average evaluation score. Due to its design, grid search can be computational heavy.

3.7.2 Randomised search using Cross-Validation

A less computational heavy alternative is the randomised search using cross-validation. The algorithm performs n times where parameter values are random sampled from a given list. Equivalent to the grid search, it then performs a cross-validation and calculates the average evaluation score. The parameters yielding the highest average score are then returned.

3.8 Features

As previously mentioned, the machine learning algorithm is trained on data referred to as features. A feature is a measurable characteristic of a given phenomenon. The features used in this project are derived from the price data of a given asset. Because the price data is a time series, the features also take the form of time series as illustrated below.

Table 3.1: Example of a hypothetical feature space for a machine learning problem

Date	Feature 1	Feature 2	...	Feature n
2017-03-04	1.02	0.80	...	0.95
2017-03-03	0.98	0.75	...	0.62
⋮	⋮	⋮	⋮	⋮
2012-05-02	0.30	1.11	...	1.33

The features are derived individually and are the result of commonly used trading strategies for exploiting Time series momentum. In this section, these trading strategies are introduced together with the theory underpinning each of them. It is worth noting that the strategy itself will not be taught to the model as this would defy the purpose of machine learning. Instead several time series on which trading strategies can be built are provided to the model as features. It is up to the model to create from the features.

3.8.1 Filters

A common denominator across the trading strategies are filtering techniques. Filters are used in signal processing to remove unwanted component from a signal. Here, price data constitutes the signal and the

unwanted component are the small fluctuations in the time series obscuring a larger trend. An example of smoothing is illustrated in figure 3.8 where a 30-day simple moving average ("SMA") section 3.8.2 is compared to the original price chart.

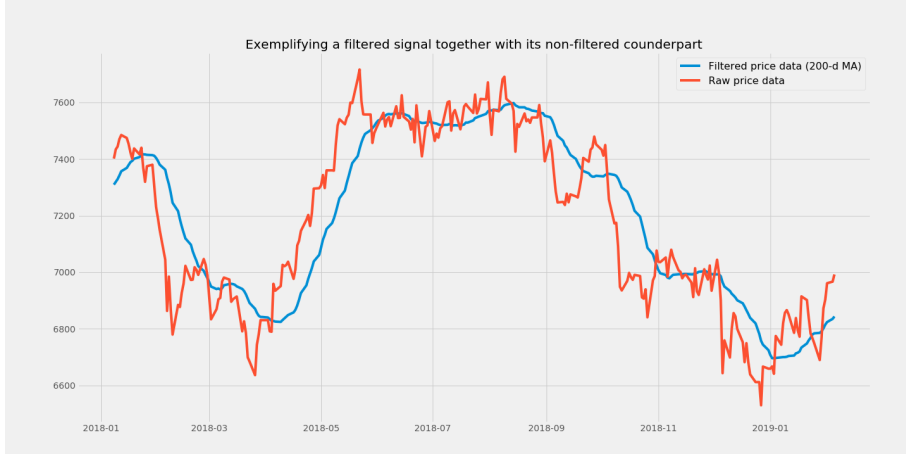


Figure 3.8: Visualising the effect of smoothing with raw price data and a 30 day SMA

What price changes are deemed small fluctuations is highly subjective and it is essentially determined by the trend that the filtered signal is supposed for pick up. The longer the trend, the higher the smoothing parameter. The parameter choice is a balance between noise reduction and information loss.

3.8.1.1 Notation

Here, some notation for the filtering techniques is introduced. Considering time series data for a given asset, the ordered sequence of observations are denoted $y = \{\dots, y_{-2}, y_{-1}, y_0, y_1, y_2, \dots\}$. The underlying trend is an unobservable process x_t . Let \hat{x}_t be the estimator of said process and the result of the filtering process:

$$\hat{x}_t = \mathcal{L}(y) \quad (3.34)$$

with estimates $\hat{x} = \{\dots, \hat{x}_{-2}, \hat{x}_{-1}, \hat{x}_0, \hat{x}_1, \hat{x}_2, \dots\}$

Considering the problem of discerning a trend without the benefit foresight, the requirement of a casual filter follows naturally. Furthermore, by limiting the scope to time invariant filters the estimate can be expressed by:

$$\hat{x}_t = \sum_{i=0}^{n-1} \mathcal{L}_i y_{t-i} \quad (3.35)$$

Using this expression, the liner filter is completely defined by the window function \mathcal{L}_t and its support.

3.8.2 Simple moving average

A simple moving average is defined as

$$\mathcal{L}_i = \frac{1}{n} \mathbb{1}_{\{i < n\}} \quad (3.36)$$

where the only parameter is n , the window length. The estimator can be expressed as:

$$\hat{x}_t = \frac{1}{n} \sum_{i=0}^{n-1} x_{t-i} \quad (3.37)$$

The length of the window defines the smoothness of the filtered signal and the parameter choice corresponds to the important balance between noise reduction and information loss.

3.8.3 Exponential moving average

An exponential moving average filter can be described as as

$$\mathcal{L}_i = (1 - \theta)\theta^i \mathbf{1}_{\{i > 0\}} \quad (3.38)$$

The exponential moving average is similar to the moving average but it puts more emphasis on recent values by a weight decay which increases further away from a given observation. The exponential moving average is:

$$\hat{y}_t = (1 - \theta) \sum_{i=0}^{\infty} \theta^i y_{t-i} \quad (3.39)$$

where $0 < \theta < 1$ controls the speed at which the weights decay. The smaller the θ , the faster the weights will decay. By this definition, the parameter for a given time span S can be calculated as:[1]

$$\theta = \frac{S - 1}{S + 1} \quad (3.40)$$

3.8.4 Moving average crossovers

A common way to define trading rules is using what is known as moving average crossovers. The moving averages are in general based on the price of the asset and not returns. The moving average crossover can be expressed as:

$$\hat{y}_t^{f,s} = \frac{1}{f} \sum_{i=0}^{f-1} y_{t-i} - \frac{1}{s} \sum_{i=0}^{s-1} y_{t-i} = \hat{y}_t^f - \hat{y}_t^s \quad (3.41)$$

The idea is based on utilising two moving averages, one fast, \hat{y}_t^f and one slow, \hat{y}_t^s , where $f < s$. When $\hat{y}_t^f > \hat{y}_t^s$ the price is said to be in an upward trend and conversely, when $\hat{y}_t^f < \hat{y}_t^s$ the price is said to be in a downward trend. As such, the trading rule is generally to buy the asset when the moving averages cross $\hat{y}_t^f > \hat{y}_t^s$ and to short the asset when $\hat{y}_t^f < \hat{y}_t^s$. As one of the most fundamental trading indicators, the strategy as illustrated in the introduction, figure 1.1.

The choice of smoothing parameters f, s will determine which trends the indicator picks up on. This process, which is further described in the method 4.5, is aided by the use of return signature plots. These plots illustrates the relative weights assigned to historical returns by the feature. To construct such plots, a function yielding the weights is derived.

Under the assumption of log prices, the weights assigned to observations can be derived by expanding on 3.41 and expressing the price at a time $t - i$ as the difference between the price at time t and the sum of the returns in the interim $(t - n, t]$. [4]

$$y_{t-n} = y_t - \sum_{i=0}^{n-1} r_{t-i} \quad (3.42)$$

Expanding the summation yields a pattern:

$$y_{t-n} = \begin{cases} y_t & \text{if } n = 0 \\ y_t - r_t & \text{if } n = 1 \\ y_t - r_t - r_{t-1} & \text{if } n = 2 \\ \vdots & \end{cases} \quad (3.43)$$

By taking the sum over the emerging pattern in 3.43 an equivalent expression to that of 3.41 can be rewritten as:

$$\hat{y}_t^n = \frac{1}{n} \sum_{i=0}^{n-1} r_{t-i} = y_t - \sum_{i=1}^{n-1} \frac{n-i}{n} r_{t+1-i} \quad (3.44)$$

Now an expression for the moving average crossover can be derived.

$$\hat{y}_t^{f,s} = \hat{y}_t^f - \hat{y}_t^s = \left[y_t - \sum_{i=1}^{f-1} \frac{f-i}{f} r_{t+1-i} \right] - \left[y_t - \sum_{i=1}^{s-1} \frac{s-i}{s} r_{t+1-i} \right] = \sum_{i=1}^{s-1} \frac{s-i}{s} r_{t+1-i} - \sum_{i=1}^{f-1} \frac{f-i}{f} r_{t+1-i} \quad (3.45)$$

From equation 3.45 one can draw the conclusion that the moving average crossover of logarithmic prices consists of linearly weighted previous returns. The trend expression can be simplified as:

$$\hat{y}_t^{f,s} = \sum_{i=1}^{s-1} w_i r_{t+1-i} \quad (3.46)$$

Where the weights can be expressed as:

$$w_i = \begin{cases} \frac{i}{f} - \frac{i}{s} & \text{for } 1 \leq i \leq f-1 \\ 1 - \frac{i}{s} & \text{for } f \leq i \leq s-1 \end{cases} \quad (3.47)$$

The equations 3.47 are of relevance when analysing how the weighing schemes changes with different combinations of f and s . For easier comparison over different pairs of s and f the weights are normalised by the sum of the weights:

$$\sum_{i=1}^{f-1} \left(\frac{i}{f} - \frac{i}{s} \right) + \sum_{i=f}^{s+f-1} \left(1 - \frac{i}{s} \right) = \sum_{i=1}^{f-1} \frac{i}{f} + \sum_{i=k}^{s+f-1} 1 - \sum_{i=1}^{s-1} = \frac{1}{2}(f-1) + s - \frac{1}{2}(s-1) = \frac{1}{2}(s-f) \quad (3.48)$$

The normalised weights can be expressed as:

$$w_i = \begin{cases} \frac{2}{(s-f)} \frac{i}{f} - \frac{i}{s} & \text{for } 1 \leq i \leq f-1 \\ \frac{2}{(s-f)} - \frac{i}{s} & \text{for } f \leq i \leq s-1 \end{cases} \quad (3.49)$$

3.8.5 MACD

Moving Average Convergence Divergence or MACD was invented by Gerald Appel in the late 70s.[1] The indicator is based on an exponential moving average crossover. Just like in the case of the simple moving average, the crossover is the difference between a fast and a slow moving average where the the speed is decided by the decay the parameter θ . The MACD can be expressed as:

$$\hat{y}_t^{f,s} = (1 - \theta_f) \sum_{i=0}^{t-1} \theta_f^i y_{t-i} - (1 - \theta_s) \sum_{i=0}^{t-1} \theta_s^i y_{t-i} \quad (3.50)$$

where $\theta_f < \theta_s$. When using MACD, Appel suggests including an additional component, an indicator based on the difference between the MACD and a exponential moving average of itself. The indicator, which is referred to as the MACD difference, is defined as:

$$\hat{y}_t^{f,s,h} = \hat{y}_t^{f,s} - (1 - \theta_h) \sum_{i=0}^{t-1} \theta_h^i \hat{y}_{t-i}^{f,s} \quad (3.51)$$

Where h is the number of observations used with Equation 3.40 to yield the smoothing parameter. The main indicator, the MACD, is interpreted in the same way as the moving average crossover generating buy and sell signals in the event of crossovers. The MACD difference is a supporting indicator used to identify trend reversals earlier than by simply using MACD. The signal for trend reversal is said to be particularly strong when a crossover of the MACD difference coincides with relatively high or low values for the MACD

[1]. The two indicators are plotted in figure 3.9 where the reader may attempt to identify a pattern if one exist.

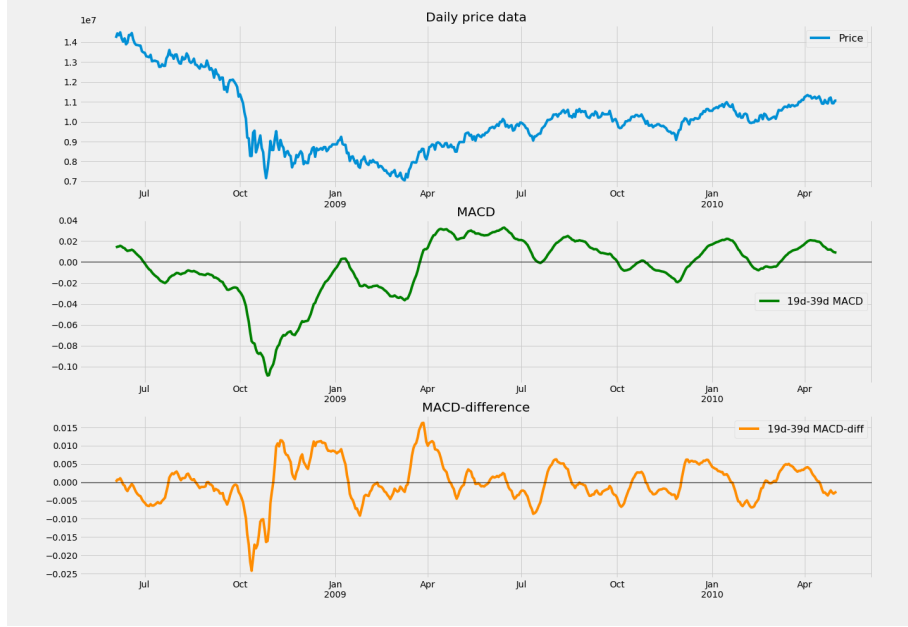


Figure 3.9: Price data, 19-39 MACD and 19-39-9 MACD difference, see 4.2.4

Similarly to the moving average crossover, the weights assigned to returns over time can be derived for both the MACD and the MACD difference. Starting with MACD, under the assumption of log prices, the return at time t can be expressed as $\sum_{i=1}^{t-1} r_{t-i} = y_t - y_1$. Equation 3.39 can be rewritten as:

$$\begin{aligned} \hat{y}_t &= (1 - \theta) \sum_{i=0}^{t-1} \theta^i y_{t-i} = (1 - \theta) \sum_{i=0}^{t-1} \theta^i \sum_{j=1}^{t-i} r_j \stackrel{\text{Inter.ch sums}}{=} (1 - \theta) \sum_{j=1}^t r_j \sum_{i=0}^{t-j} \theta^i = \\ &= (1 - \theta) \sum_{j=1}^t r_j \frac{1 - \theta^{t-j+1}}{1 - \theta} = \sum_{j=1}^t r_j (1 - \theta^{t-j+1}) \end{aligned} \quad (3.52)$$

Which means the MACD based on returns can be expressed as:

$$\hat{y}_t^{f,s} = \sum_{j=1}^t r_j (1 - \theta_f^{t-j+1}) - \sum_{j=1}^t r_j (1 - \theta_s^{t-j+1}) = \sum_{j=1}^t r_j (\theta_s^{t-j+1} - \theta_f^{t-j+1}) = \sum_{j=1}^t r_j w_j \quad (3.53)$$

For the MACD difference, the weights can also be derived. This is a somewhat lengthy process and will thus be omitted here. The curious reader is referred to [4]

$$\hat{y}_t^{f,s,h} = \sum_{k=1}^t r_k \left[(\theta_s^k - \theta_f^k) - (1 - \theta_h) \left(\frac{\theta_s^k - \theta_h^k}{1 - \frac{\theta_h}{\theta_s}} - \frac{\theta_f^k - \theta_h^k}{1 - \frac{\theta_h}{\theta_f}} \right) \right] = \sum_{k=1}^t r_k w_k \quad (3.54)$$

3.8.6 Hodrick-Prescott filter

The Hodrick Prescott (HP) filter is frequently used in econometric studies to extract a low-frequency component from a time-series. [19] The filter is, for example, often used by economists to decompose GDP data into a trend component and cyclical component.[3] Consider, in the context of the problem at hand, that the price development can be decomposed into an estimated trend component \hat{x}_t and noise ϵ_t

3.9.1 Regularised logistic regression

When including a regularisation parameter in the logistic regression, the freedom of the model is restricted. By using Lasso [40] or l_1 regularisation, the regression will strive to assign weights, Equation 3.29, of zero to the features. The idea is to discard the features which correspond to the zero-weights. This method of regularisation relies on the l_1 -norm and involves adding an “absolute value of magnitude” of the weights as penalty term to the loss function.

The weights are calculated by minimizing the loss function

$$L_{lasso}(\hat{w}) = \sum_{i=1}^n (y_i - x'_i \hat{w})^2 + C \sum_{j=1}^m |\hat{B}_j| \quad (3.62)$$

where C is a penalising term which determines the strength of the regularisation.

3.10 Stratified sampling

While machine learning algorithms typically perform better with an increased amount of training data, another important aspect is the relevance of the data.

To determine what is relevant, it is useful to consider the nature of the data. Price data for a given asset can display radically different behaviour over time. Imagine a period of relative calm followed by a strong upward move which is sustained over longer period. In this case, it would be desirable to include data coinciding with the initial significant upward divergence. This is warranted under the assumption that the algorithm becomes better at identifying such significant trend changes. Furthermore, it is also desirable to include observations for the calm period with a reasonable frequency in order to make the model accustom to that type of environment as well.

One intuitive way to identify relevant data would be to define a threshold volatility and to select all data points exceeding the threshold. Volatility is defined as the standard deviation of log returns. One issue with this approach is that the volatility of asset prices tend to change over time. Changes in volatility regimes often applies across securities and even across asset classes as a reaction to some external event in the wider economy.[43]

For the sampling method not to oversample in times of high volatility, it is desirable to have a threshold which changes according to the current volatility regime. This can be achieved by using a CUMSUM filter.

3.10.1 CUMSUM filter

The CUMSUM filter is a sequential analysis method commonly used for quality control. It aims to detect a shift in the average value of a measured value away from its target value. For the problem at hand, the shifts are identified by comparing the returns to the recent volatility. Consider the log returns from a given asset $\{r_t\}_{t=1, \dots, T}$, the cumulative sums are defined as

$$S_t^+ = \max\{0, S_{t-1}^+ + r_t - \mathbb{E}_{t-1}[r_t]\} \quad (3.63)$$

$$S_t^- = \min\{0, S_{t-1}^- + r_t - \mathbb{E}_{t-1}[r_t]\} \quad (3.64)$$

With initial boundary conditions, $S_0^+ = 0$ and $S_0^- = 0$. The equations 3.63 and 3.64 corresponds to cumulative sums for upward and downward moves respectively. The observations deemed divergent from the target value are determined by a threshold h . The threshold is activated when:

$$S_t \geq h \Leftrightarrow \exists \tau \in [1, t] \left| \sum_{i=\tau}^t (r_i - \mathbb{E}_{i-1}[r_i]) \geq h \right. \quad (3.65)$$

where $S_t = \max\{S_t^+, -S_t^-\}$. Note that after each activation of the threshold h the value S_t is reset to zero. The h will be defined as a exponential moving average 3.8.3 of historic volatility. The exponential moving average will yield a smoothed volatility curve with emphasis on recent volatility, thus reflecting the current volatility regime.

3.11 Model evaluation scoring

Assessing the quality of different models requires a measurement that makes sense from a financial perspective as well as mathematically. Measuring the total error discussed in section 3.2 would be ideal but is not easily applicable for a classification problem. For this reason, five other scoring methods were used when developing models:

1. **Accuracy**
2. **F1**
3. **Negative log-loss**
4. **Matthews Correlation Coefficient**
5. **Recall**

All of which are based on the concepts of the confusion matrix. A confusion matrix evaluates a classification model by comparing the predictions to their true values. In the binary case it is a simple 2x2 matrix 3.2 with the elements:

- **True Positive:** Model **accurately** predicts an assets future **upward** price movement
- **True Negative:** Model **accurately** predicts an assets future **downward** price movement
- **False Positive:** Model **inaccurately** predicts an assets future price to go **up** when it in fact goes down
- **False Negative:** Model **inaccurately** predicts an assets future price to go **down** when it in fact goes up

Table 3.2: Table illustrating a 2x2 confusion matrix.

		Predicted	
		0	1
Actual	0	True Negative	False Positive
	1	False Negative	True Positive

3.11.1 Accuracy

Accuracy is then calculated as the total number of accurate predictions divided by the total amount of predictions. Because the developed trading strategy allows for short selling, it is as important to predict positives (upward movement) as negatives (downward movements).

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \quad (3.66)$$

However, one needs to be aware of two flaws associated with the accuracy measurement. First, if the distribution of data is skewed, say for example the data set consist of more positive outcomes than negative, a model can achieve high accuracy by always predicting positive outcomes. Secondly, especially when applied to finance, accuracy may not be a proper measurement of model quality. A model with high accuracy may still lose money if the inaccurate predictions generate larger losses than accurate ones makes profit. A strategy can have 60% accuracy and still lose money. If it predicts most of the small upward movements correctly but fails to predict substantial losses occurring at rare occasion, such as a financial crisis, all of the portfolio's accumulated value may be lost.

To account for these imperfections of the Accuracy measure, other scoring methods were used as well.

3.11.2 F1-Score

When the data set has a class which dominates the majority of outcomes, F1 gives a more accurate picture than Accuracy. The F1-Score is defined as the (equally weighted) harmonic mean of recall and precision:

$$F1 = 2 * \frac{precision * recall}{precision + recall} \quad (3.67)$$

where,

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (3.68)$$

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (3.69)$$

Recall, also referred to as "true positive rate" is the share of true positive predictions amongst all positive outcomes. For this problem, it is the percentage of the total observations that have an upward price movement which the model successfully predicts as upward price movements.

Precision is the amount of predictions that are true positives out of the total amount of predicted positive outcomes. In other words, the percentage of accurately predicted upward movements from the total amount of predicted upward price movements.

The reader may note that F1-Score focuses on the positive predictions and neglects negative ones. This becomes particularly problematic when the data set consist of more positive outcomes than negative ones. F1 is designed for cases when the set consists of more negative classifications than positive. To illustrate this, imagine a very imbalanced set with 80 positive classifications and only 20 negative classifications. The model has, for some reason, started to only predict positive outcomes and the model performance is evaluated. The confusion matrix becomes:

		Predicted	
		0	1
Actual	0	0	20
	1	1	80

This scenario would yield the following accuracy and F1-score:

$$Accuracy = \frac{80 + 0}{80 + 0 + 20 + 0} = 80\% \quad (3.70)$$

$$Recall = \frac{80}{80 + 0} = 100\% \quad (3.71)$$

$$Precision = \frac{80}{80 + 20} = 80\% \quad (3.72)$$

$$F1 = 2 * \frac{0.8 * 1}{0.8 + 1} = 88.89\% \quad (3.73)$$

A high accuracy and a high F1-Score, this looks promising. However, the model has only guessed positive outcomes and has failed to predict any of the negative cases. This imperfection can be addressed by exchanging the definition of positive and negative . The example above would then yield the following scores:

$$Accuracy = \frac{0 + 80}{0 + 80 + 0 + 20} = 80\% \quad (3.74)$$

$$Recall = \frac{0}{0 + 20} = 0\% \quad (3.75)$$

$$Precision = \frac{0}{0 + 0} = 0\% \quad (3.76)$$

$$F1 = 2 * \frac{0 * 0}{0 + 0} = 0\% \quad (3.77)$$

Accuracy still indicates a good classifier, but now F1 recognises the imperfections of the model. As such, it is important to analyse the data set and determine whether the data has a majority of upward or downward price movements and define positive and negative labels accordingly.

3.11.3 Negative Log Loss

To account for the issues with inaccurate predictions resulting in large losses one might attempt a Negative log loss scoring function. This method takes predicted class probabilities into consideration.

The problem with a standard accuracy measurement is the equally weighing of predictions with high respectively low probability. Predictions with a high respective low certainty leading to an inaccurate prediction are treated and weighted equally. A more reasonable measurement would penalise high certainty predictions that turn out to be false. The log loss metric accounts for this by calculating the log-likelihood of the classifier given its true label.

$$L[Y, P] = -\log[Prob[Y|P]] = -N^{-1} \sum_{n=0}^{N-1} \sum_{k=0}^{K-1} y_{n,k} \log[p_{n,k}] \quad (3.78)$$

where

- $p_{n,k}$ is the probability associated with prediction n of label k .
- Y is a $1 - of - K$ binary indicator matrix, such that $y_{n,k} = 1$ when observation n was classified as label k out of K possible labels, and 0 otherwise.

Consider a simple classifier that predicts two positive outcomes, where the true labels are positive and negative. As an example it creates one accurate and one inaccurate prediction, thus yielding an accuracy of 50%. The log-loss for this scenario would be the following if the respective probability of hit and miss were in the range of [0.5-0.9]:

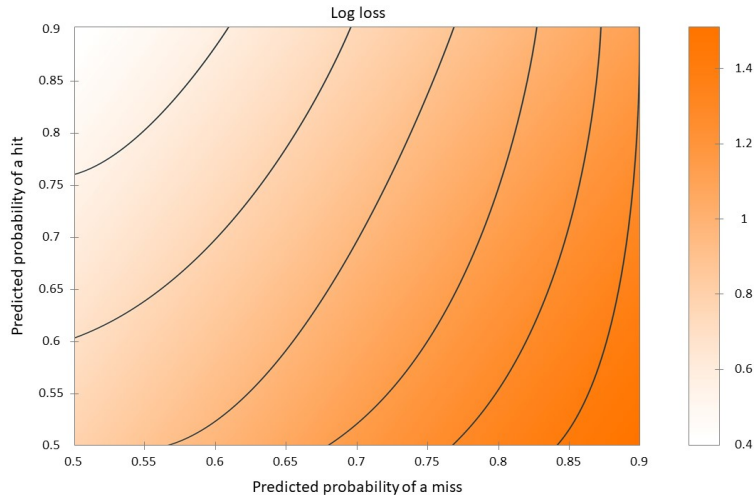


Figure 3.10: Log loss as a function of predicted probabilities of hit and miss

Figure 3.10 depicts a higher log-loss on the right hand side since the model should be punished for producing inaccurate predictions with a high probability. Similarly, there are lower log-loss on the upper part of the picture, as accurate predictions with high certainty is desirable. The top left corner represents the optimal scenario, having produced one successful prediction with 90% certainty and missed one prediction, however

after a certainty of only 50%. The bottom right corner represents the worst scenario, missing a prediction with a certainty of 90% and one successful prediction after 50% certainty.

Intuitively, the models should be ranked by how high the score is. Since the log-loss metric is positive and presents a model as perfect when $\log\text{-loss} = 0$ negative log loss was used instead. In order for the negative log loss scoring to make perfect sense in a financial scenery, higher probabilities would need to be correlated to more extreme returns. Thus, a high probability of predicting a positive classification should also be more likely to generate a high return. Otherwise, the model will be less likely to catch extreme value returns, yielding false predictions that may cause substantial losses.

3.11.4 Matthews Correlation Coefficient

The Matthews Correlation Coefficient ("MCC"), introduced by Brian W. Matthews in 1975 is a measure used to evaluate the quality of classifications. [30] It is a particularly useful measure when the classes are highly imbalanced. [7]. MCC is essentially a correlation coefficient between the observed and the predicted classifications. It yields a value in $\{-1, 1\}$ where $+1$ is a perfect prediction, 0 is random and -1 represents an inverse prediction. In the two-class case, given a confusion matrix, the measure is defined as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (3.79)$$

The measure can be generalised to the multiclass case. With a $K \times K$ confusion matrix C the MCC can be defined as

$$MCC = \frac{\sum_k \sum_l \sum_m C_{kk} C_{lm} - C_{kl} C_{mk}}{\sqrt{\sum_k (\sum_l C_{kl}) (\sum_{k' | k' \neq k} \sum_{l'} C_{k'l'})} \sqrt{\sum_k (\sum_l C_{lk}) (\sum_{k' | k' \neq k} \sum_{l'} C_{l'k'})}} \quad (3.80)$$

3.12 Backtesting

Backtesting is the general term describing methods for assessing how well a trading strategy performs by exploring how it would play out when tested on historical data. Backtesting is used to gain confidence in a model before putting it into practice. Calculating the value of and as an extension, the returns from, futures contract is not arbitrary. This caveat will be addressed before the different backtesting methods are introduced.

3.12.1 Calculating returns from futures

Futures contracts have a quoted price just like other financial instruments. The minimum amount the price of a futures contract can change is referred to as *tick size*. Furthermore, the *tick value* of a contract is the value of each tick. Finally, the futures contract has a *contract value* which is the amount an investor would pay for a single contract. The calculations are most conveniently described by an example. Consider a futures contract denominated in pound sterling with *Tick size* = 0.5, *Tick value* = 5 and *Contract value* = 69955

Table 3.3: Price development of a hypothetical futures contract

Dates	Price
2019-01-28	6730
2019-01-29	6686
2019-01-30	6809
2019-01-31	6878.5

Given the price series in table 3.3, the profit over the period can be calculated in two steps:

$$Number\ of\ ticks = \frac{6878.5 - 6730}{Tick\ size} = \frac{148.5}{0.5} = 297 \quad (3.81)$$

Knowing the value per tick

$$Profit = Number\ of\ ticks \times Tick\ value = 297 \times 5 = \pounds 1485 \quad (3.82)$$

Assuming a portfolio of of $\pounds 500'000$ the maximum amount of contracts which can be bought are

$$Number\ of\ contracts = \left\lfloor \frac{500000}{69955} \right\rfloor = 7 \quad (3.83)$$

If 7 contracts were bought, the trade would result in a portfolio gain of $7 \times \pounds 1485 = \pounds 10395$ yielding

$$Return = \frac{\pounds 10395}{\pounds 500000} = 2.079\% \quad (3.84)$$

3.12.2 Traditional walk forward

In a traditional walk forward, the algorithm is trained on data contained in a moving window. An unknown label at t_0 is predicted by using data from $[t_{-n}, t_{-1}]$. For the label at t_1 , the window is moved forward and data from $[t_{-n+1}, t_0]$ is used.

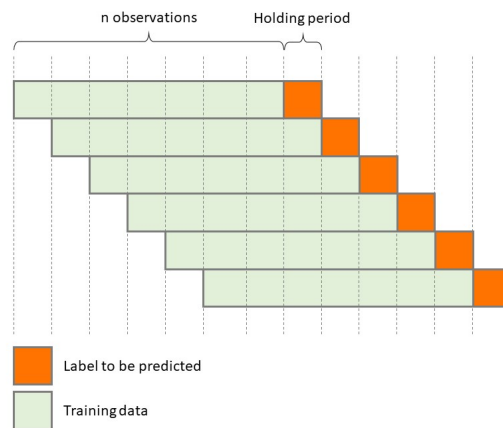


Figure 3.11: Traditional walk forward

3.12.3 Warm-up walk forward

A potential drawback with the traditional walk forward is the possible lack of diversity in the training data. If the training set is limited to a time period of price appreciation, the algorithm might fail to recognise a costly shift in the market. As such, a type of walk forward was included where all data before the prediction date is included. A drawback with the warm-up walk forward is the discrepancies which arise from a continuously increasing training set. It could have an impact on performance over time and makes comparing different contracts harder as the amount of historical data differs across the contracts.

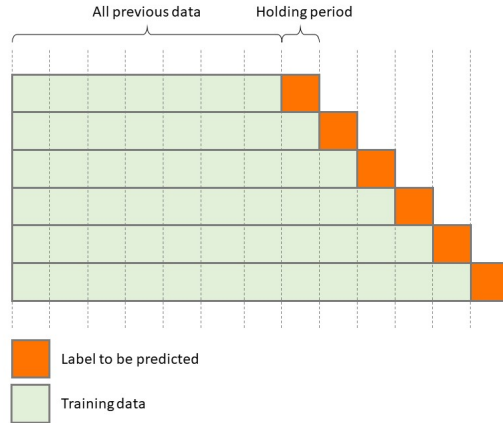


Figure 3.12: Warm-up walk forward

3.12.4 Combinatorial backtesting with Cross-Validation

A problem with the walk forward method above is that it only validates a model performance on the historical prices. Ideally, a model should perform well in the future, for an unknown path, not only for the one observed historically. To check if a model is likely to perform well overall, a plain cross-validation can be used to backtest. However, due to the characteristics of cross-validation, backtesting using this method exclusively would only yield one "new" historical path. Since the goal of this backtest was to observe how the model could have performed it would be more appealing if many different scenarios were generated. This was addressed by using a so called combinatorial backtesting with cross-validation. This method relies on creating different historical paths by combining the results from different sets of cross-validated backtests. A concept which is best illustrated with an example.

Imagine using cross-validation to split the data set into six different groups (G). In addition, two of these groups are used as test groups and the four remaining groups are used as training groups. This type of partitioning represents one unique set which could have, theoretically, been a historical outcome. In total there are $\binom{6}{2}$ different partitions which means there are 15 different sets (S) of data to validate on.

Table 3.4: Illustration of the sets generated by cross-validation partitioning. (x) indicates a testing group. Blank indicates a training group.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
G1	x	x	x	x	x										
G2	x					x	x	x	x						
G3		x				x				x	x	x			
G4			x				x			x			x	x	
G5				x				x			x		x		x
G6					x				x			x		x	x

As these sets were generated, five different historical paths were created by combining different test groups from different sets. For example one path created consisted of G1(S1), G2(S1), G3(S2), G4(S3), G5(S4) and G6(S5). Which means predictions for the first and second time period (G1 and G2) were taken from the first cross-validated set (S1), predictions for the third time period (G3) were taken from the second set (S2) etc. This represents the actual historical path since the data set was initially divided into six groups G1-G6.

See figure 3.13 for an illustration where a data set from 2005-2010 has been divided into six groups: 2005, 2006, 2007, 2008, 2009 and 2010. The first set is thus trained on 2007-2010 and tested on 2005 and 2006.

The second set is trained on 2006,2008,2009,2010 and tested on 2005 and 2007, etc.

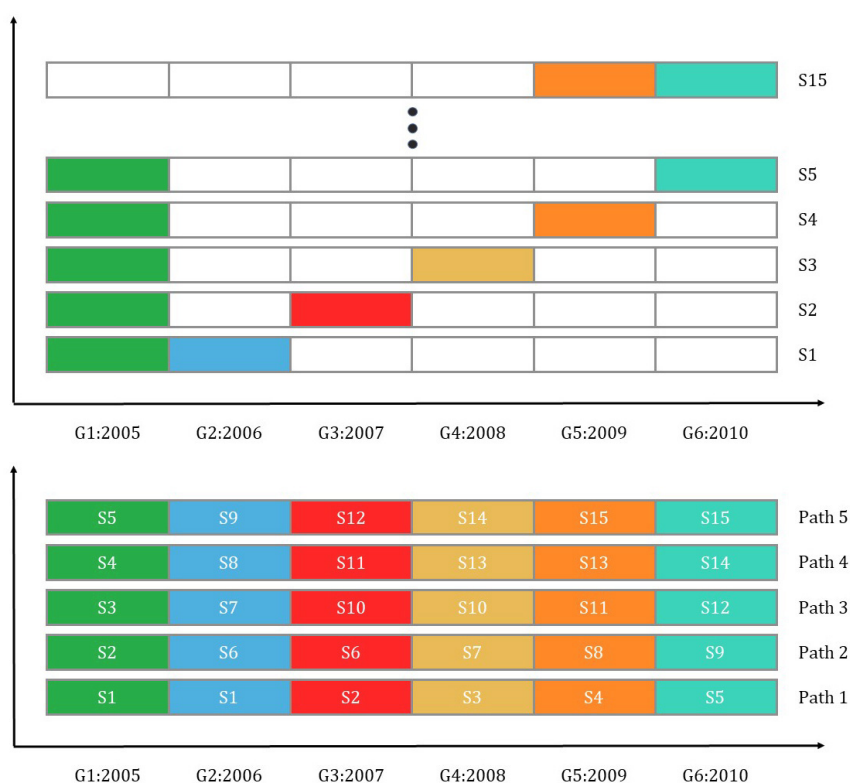


Figure 3.13: Upper part illustrates which fold (colored by year) is a validation set in each of the fifteen cross-validated sets. The lower part illustrates how validated sets are combined to recreate a historical path. Green = 2005, Blue = 2006, Red = 2007, Yellow = 2008, Orange = 2009, Teal = 2010.

3.13 Benchmarking

In benchmarking, the aim is to assess the relative performance of an asset or portfolio of assets compared to relevant peers. The comparison is conducted by comparing metrics derived from the assets or portfolio of assets. In the case of a single asset, while the choice of benchmark is largely proprietary, a natural choice of benchmark would be to use the same asset over the same time period but simply employ a buy-and-hold strategy.

3.13.1 Performance metrics

Four performance metrics were used to assess the performance of the models and benchmarks: maximum drawdown, Sharpe ratio, Sortino ratio, and gross return.

3.13.1.1 Maximum drawdown

Maximum drawdown is the maximum percentage loss over a given time period measured from peak to a trough before a new peak is attained. The maximum drawdown at time t can be expressed as:

$$MDD(t) = \max_{\tau \in (0,t)} \left\{ \max_{t_0 \in (0,\tau)} \left\{ \frac{y_{t_0} - y_{\tau}}{y_{t_0}} \right\} \right\} \quad (3.85)$$

where y_t is the price at time t . Limiting drawdowns is very important when managing money as substantial losses can result in outflows as investors withdraw their money in dismay. Furthermore, an erosion of capital means that potential increases in asset values measured as a percentage result in lower absolute return which makes it harder to recuperate the losses.

3.13.1.2 Sharpe ratio

The Sharpe ratio was first introduced by William F Sharpe in 1966. [35] The idea is to measure the risk adjusted return by accounting for the volatility of the returns. Strong historical performance is deemed less impressive if the gains were achieved through taking significant risk. The measure is defined as:

$$\text{Sharpe Ratio} = \frac{r_i - r_f}{\sigma_i} \sqrt{n} \quad (3.86)$$

where r_i is the average return of the portfolio or asset i . r_f is the risk-free return, nowadays often assumed to be zero. σ_i is the volatility of returns which is equivalent to the standard deviation of the same. As the Sharpe ratio is quoted on an annual basis, the scaling term \sqrt{n} is needed if the returns are calculated on frequency exceeding one year. If this is the case, n is the number of return periods in a year.

3.13.1.3 Sortino ratio

The Sortino ratio is a measure of risk adjusted return very similar to the Sharpe ratio. It was introduced by Frank A. Sortino and Robert Van Der Meer in 1994 [36] as an improvement to the Sharpe ratio. The difference between the two is that the Sortino ratio only takes into account volatility to the downside or downside risk. The positive returns are excluded from the volatility measure as positive returns is something sought after. The Sorino ratio is defined as:

$$\text{Sortino Ratio} = \frac{r_i - r_f}{\delta_i} \sqrt{n} \quad (3.87)$$

Where δ is equal to the square root of the second-order lower partial moment of returns $\sqrt{LPM_2(\mathbf{r})}$.

$$LPM_2(\tau) = \frac{1}{T} \sum_{t=1}^T \max[\tau - R_t, 0]^2 \quad (3.88)$$

3.13.1.4 Gross return

Finally, for the sake of completeness, the gross return is included. The gross return is defined as:

$$\text{Gross Return}(t) = \frac{y_t - y_{t_0}}{y_{t_0}} \quad (3.89)$$

Chapter 4

Method

All analyses and data visualisation were conducted using Python 3.7. The most important packages used were:

- NumPy - Support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays [37] [41]
- pandas - Data structures and operations for manipulating numerical tables and time series [31]
- matplotlib - Plotting library extension of NumPy [20]
- seaborn - Data visualisation based on matplotlib [42]
- scikit-learn - Machine learning library used for its classifiers and cross-validation tools [34]

4.1 Data gathering

Data on 7 futures contracts based on equity indices were gathered from the Bloomberg Terminal. Only the daily closing price of each contract was used to create the features. As can be seen in table 4.1, the amount of historical data varies across the futures contracts. The lack of data has implications for some of the tests and in some instances, leads to the exclusion of contracts. This will be commented when relevant.

Table 4.1: The futures contract included in the tests. Note that *NK1* historical prices extend beyond 1980-01-01. The time-series was truncated for simplicity. The contract prices are as of 2019-04-25

Ticker	Underlying asset	Country	Start date	Tick size	Tick value	Contract value
<i>Z 1</i>	FTSE 100 Index	UK	1988-02-26	0.5	5	GBP 69955
<i>ES1</i>	S&P 500 Index	US	1997-09-09	0.25	12.5	USD 135887.5
<i>NQ1</i>	Nasdaq 100 Index	US	1999-06-21	0.25	5	USD 138635
<i>GX1</i>	DAX Performance Index	DE	1990-11-23	0.5	12.5	EUR 279462.5
<i>NK1</i>	Nikkei 225 Index	JP	1980-01-01	10	1000	JPY 20820000
<i>CF1</i>	CAC 40 Index	FR	1988-12-07	0.5	5	EUR 50100
<i>DM1</i>	Dow Jones Industrial Average	US	2002-04-05	1	5	USD 125830

4.2 Features

The features utilised in the algorithm are the result of filters being applied to the price data, see section 3.8. Table 4.2 depicts the trading indicators included in the model and the number of variants of each individual indicator.

Table 4.2: Trading indicators included as features in the model

Name	Variants
MA crossover	9
MACD	6
MACD difference	9
HP crossover	4

4.2.1 Adapting the trading indicators

For the machine learning algorithm to perform well, the quality of the training data is imperative. The selection of relevant training data is partly addressed through the the stratified sampling in section 4.4.1. However, many of the traditional trading strategies within the realm of technical analysis are based on chart interpretation.

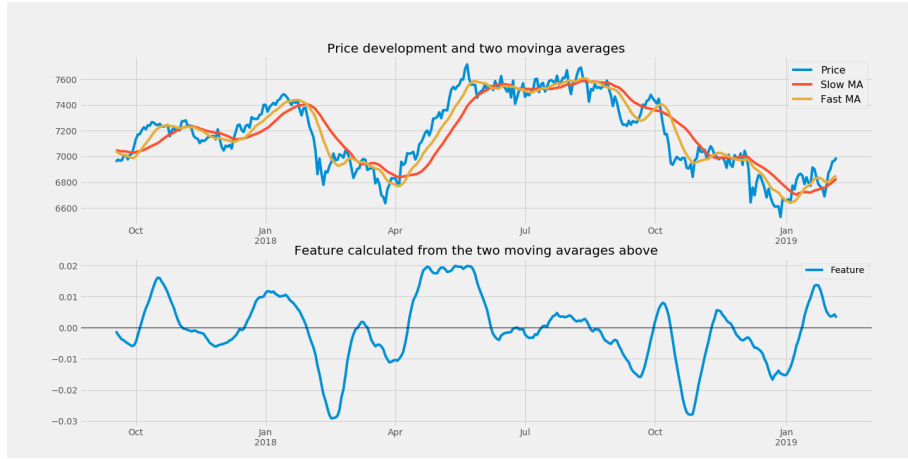


Figure 4.1: Top: $Z 1$ futures price and its 12-d and 72-d SMA. Bottom: The feature over time generated from the two SMA above.

In figure 4.1, the moving averages are easily brought into the chart as they are on the same scale as the price. As the theory proposes, the difference between the two SMAs can be used as a trading indicator which takes the value 0 in case of a crossover. Furthermore, the size of the difference between the two could have predictive qualities as the magnitude can be indicative of the strength of the trend.

Calculating the difference between the two filtered signals seems perfectly reasonable in the context of chart interpretation. However it is not a suitable feature for a machine learning algorithm. Considering that the underlying data is price, the difference between two filtered signals is essentially a dollar value and thus an absolute measure. The machine learning algorithms are often trained on substantial amounts of data which can span several decades. The price level of an asset varies over time and the general price level of an asset can differ substantially over such a long time period. As such, for the strength of the trend to be interpreted in a consistent way over time, the features need to be constructed as relative measures. All features except for the MACD difference will be defined as:

$$\hat{y}_t = \frac{\hat{y}_t^f}{\hat{y}_t^s} - 1 \quad (4.1)$$

Note that the requirement

$$\hat{y}_t^s \neq 0 \quad (4.2)$$

is satisfied for 4.1 as the asset price from which \hat{y}_t^s is derived is strictly greater than zero. The feature for the MACD difference will remain a difference as defined in Equation 3.51. This is because the calculation

of the MACD difference does not satisfy equation 4.2. While the magnitude of the feature will differ when dividing the filtered signals as opposed to calculating the difference, it will not negatively impact the result as the features are scaled. To the authors knowledge, this adaption of trading indicators to a relative measure has not been previously suggested.

4.2.2 Choosing parameters

Regardless of which filter is used, the choice of parameters will have a significant impact on which and to what extent observations in the time series are included in the filtered signal. The trade-off between information loss and noise has implications for the profitability of the trading strategy. If there is a significant reversal in the price trend of an asset the ideal strategy would immediately recognise the trend break and initiate a short position in order to profit from the new trend. A lag in the filtered signal will delay the response from the algorithm and the investor will delay before switching position. Of course, the filtering process does contribute to the algorithm by partly eliminating noise. What first appeared to be a shift in the trend was perhaps simply noise and not a sustainable trend reversal.

While it is tempting to, for a given point in time, optimise the filters in such a way that would yield the highest return over the test period, such optimisation is treacherous. The reason is that financial markets does not exhibit static behaviour. Quite the opposite, financial markets can exhibit radically different behaviour over time which requires the trading indicators not to be overly optimised to certain market conditions.

With this in mind, for each type of filter, several filters based on a range of parameters are included. It will be left to the algorithm to decide on the relevant features. Limitation is put on the parameter range by the choice of limiting the trends analysed to not exceed one year.

The parameters were chosen on the basis of established literature [25] [33] [1] [17] [19] and, when applicable, returns signature plots. This section includes the reasoning behind the parameter choice for each feature and begins with a summary of the parameters used. Furthermore, comments on the adaption of the trading indicator to a machine learning feature in accordance with 4.2.1 is included when warranted.

Table 4.3: The combinations of fast and slow filtered signals (f, s) used in the crossovers. Note that the unit is days.

	MA crossover			HP crossover	
<i>Short</i>	(4;24)	(8;24)	(12;24)	(0;8)	(0;24)
<i>Medium</i>	(12;72)	(24;72)	(36;72)	(0;72)	-
<i>Long</i>	(36;216)	(72;216)	(108;216)	(0;216)	-

Table 4.4: The combinations of fast and slow filtered signals (f, s) in the MACD and the combinations of ($f; s; h$) for the MACD difference as defined in equation 3.51. Note that the unit is days.

	MACD			MACD difference		
<i>Daily</i>	(6;19)	(12;26)	(19;39)	(6;19;9)	(12;26;9)	(19;39;9)
<i>Weekly</i>	(30;95)	(60;130)	(95;195)	(30;95;45)	(60;130;45)	(95;195;45)

4.2.3 Moving average crossover

The moving average crossovers were divided into three groups, short, medium and long aiming to capture different trends in the asset price. By a literature review for similar applications of the filtering method, the conclusion could be drawn that the choice of parameters varies significant across implementations. The choice as stated in table 4.3 was partly inspired by the papers.[25] [33] Furthermore, a return signature plot was utilised to come up with the parameters.

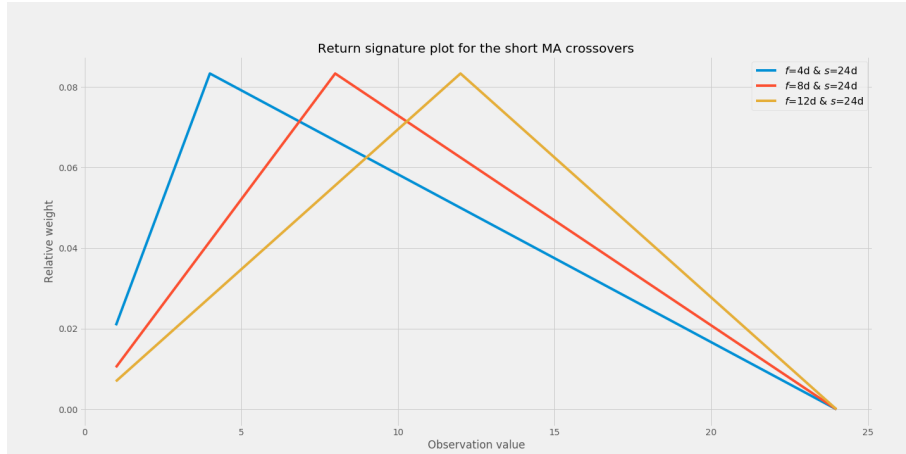


Figure 4.2: Return signature plot for the short term moving average crossovers

Notable here was that the indicator assigns the most weight to intermediate price changes and lower weights to the most recent price changes and older price changes. Of course, this attribute varies across the three crossovers. One might consider the low weights assigned to the most recent returns problematic and propose to have no smoothing on the fast moving average and thus, simply the price. The drawback of such an approach is that the resulting signal would be very noisy.

The parameters were chosen for the weights of the moving averages to be evenly distributed across the recent half of the y^s window. Furthermore, while the parameters across the short, medium and long set of moving average rules varies, the emphasis put on returns relative to each time horizon was consistent. The ratio of observations for the fast to the slow moving average was consistently $\{(1/6), (1/3), (1/2)\}$.

4.2.4 MACD

The MACD or exponential moving average crossover has a very similar definition to that of the simple moving average crossover. The only difference is that the former will put more emphasis on the most recent observations due to the exponential decay of weights.

The choice of parameter for the MACD was heavily influenced by convention. The most widely used combination of long and short time spans, $(f; s)$ for the MACD is $(12; 26)$. This parameter choice was made popular by Gerald Appel. In his book, Appel suggest two more combinations, $(6; 19)$ and $(19; 39)$. Furthermore, the author claims that the indicator is not limited to daily data. The $(12; 26)$ combination could for instance correspond to weeks and not days. [1]

With this in mind, the suggested daily $(f; s)$ -pairs were included as well as a weekly version derived by simply multiplying each of the time horizons by 5, the approximate number of trading days in a week.

Similar to the simple moving average a return signature plot showed that the different parameters yielded significant variation in the weight put on returns.

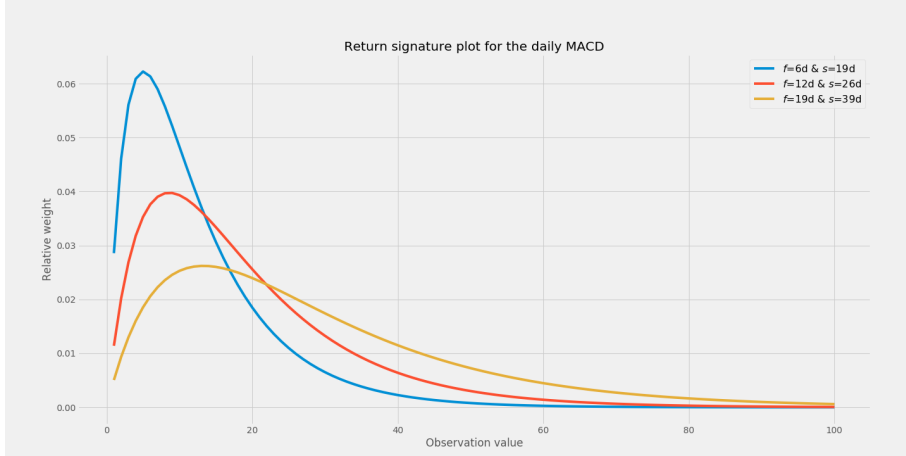


Figure 4.3: Return signature plot for the the daily version of the MACD filter

For the MACD difference, the, by Appel suggested, smoothing parameter for the MACD is 9 days. Analog to the reasoning above, the MACD difference is calculated on both a daily and weekly basis by using the smoothing parameter of 9 and 45 days respectively.

Here it is worth noting that the MACD difference is calculated as stated in equation 3.51

$$\hat{y}_t^{f,s,h} = \hat{y}_t^{f,s} - (1 - \theta_h) \sum_{i=0}^{t-1} \theta_h^i \hat{y}_{t-i}^{f,s} \quad (4.3)$$

where $\hat{y}_t^{f,s}$ is the MACD feature calculated in accordance with 4.1. The reason for not calculating the percentage difference between the two is that the second term does not satisfy the condition $(1 - \theta_h) \sum_{i=0}^{t-1} \theta_h^i \hat{y}_{t-i}^{f,s} \neq 0$.

The return signature plot for the MACD difference exhibits a different behaviour compared to previous plots. While the filter puts significant emphasis on the most recent returns it assigns negative weights to intermediate ones. The negative brings a component of mean-reversion to the filter as the magnitude of the filtered signal will be maximised by recent positive returns following intermediate negative returns.

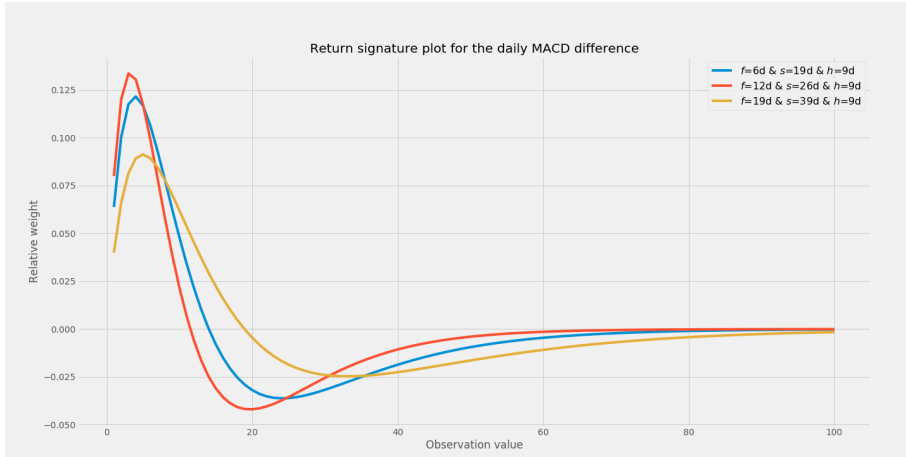


Figure 4.4: Return signature plot for the the daily version of the MACD difference filter

4.2.5 Hodrick Prescott

The one-sided Hodrick Prescott filter is not, in itself, an indicator of trends or trend reversals. As such, the feature based on the HP-filter was constructed by creating a crossover with the HP-filter and a simple moving average of itself. Thus, the feature is defined in accordance with equation 4.1 with

$$\hat{y}_t^f = \hat{y}_t \text{ and } \hat{y}_t^s = \frac{1}{s} \sum_{i=0}^{s-1} \hat{y}_{t-i} \quad (4.4)$$

Trading rules based on this type of indicator has previously been tested for trading in the FX-market. [17]

In the case of the HP- filter feature, there are two types of parameters which has to be decided upon. First, the HP-filter itself takes a smoothing parameter λ which determines the strength of the penalising term in 3.58. Second, the window length s for \hat{y}_t^s has to be decided upon.

The smoothing parameter λ was chosen as $\text{frequency}^2 \times 100$ based on the theory laid out in the paper which popularised the filter. [19] The average number of annual trading days is approximately 255 yielding $\lambda = 6'502'500$.

For the window parameter s , the somewhat arbitrary choice was based on values similar to those used in the MA crossover. The f variable being consistently zero reflects the choice of refraining from adding what is referred to as front end smoothing in the crossover.

4.3 Label creation

Throughout the testing, two different classification methods were implemented: Binary, Equation 4.5, and Ternary, Equation 4.6, classification. The classification was based on the return r_t over a given period holding period, Equation 4.8. The binary strategy always holds a position, long or short. The ternary strategy has the added freedom of holding cash if the predicted return does not exceed a threshold, Equation 4.7. As can be seen in Equations 4.8 and 4.7, a range of values for Threshold and Holding period were used in the model selection.

$$\text{Binary classification} = \begin{cases} 1 & \text{if } R_t \geq 0\% \\ -1 & \text{if } R_t < 0\% \end{cases} \quad (4.5)$$

$$\text{Ternary classification} = \begin{cases} 1 & \text{if } R_t > \text{Threshold} \\ 0 & \text{if } -\text{Threshold} \leq R_t \leq \text{Threshold} \\ -1 & \text{if } R_t < -\text{Threshold} \end{cases} \quad (4.6)$$

$$\text{Threshold} \in \{1\%, 2\%, 3\%, 4\%, 5\%\} \quad (4.7)$$

$$\text{Holding period} \in \{5, 10, 20\} \quad (4.8)$$

4.4 Data analysis and preparation

4.4.1 Stratified sampling

To ensure that the model is trained on relevant examples of price development, an event-based feature sampling method was used. The method is based on applying a CUMSUM filter to the price data to identify significant price moves dependent on the current volatility regime.

The threshold value h used in Equation 3.65 was chosen to be an exponential moving average of the standard deviations of returns. The span θ in the exponential moving average 3.40 is set to 100 observations. The choice of an exponential moving average is somewhat arbitrary yet, motivated by the common phenomenon of volatility clustering. [28]

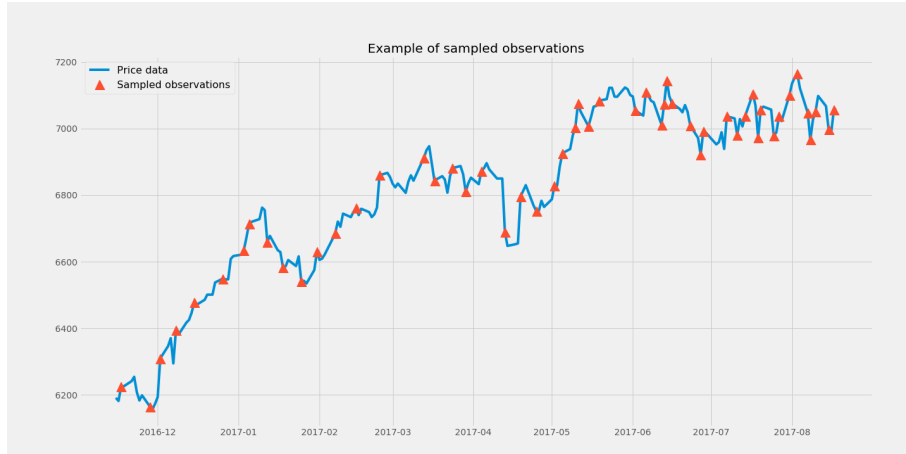


Figure 4.5: Blue line depicts the original time series for Z_1 and red triangles indicates observations selected by the CUMSUM filter

Figure 4.5 illustrates a subset of data where the CUMSUM-filter selects significant data points. Since cross-validation methods was later to be applied to the filtered data, it was important that the sampling frequency remains approximately constant over time. This is relevant because when using cross-validation, the data set is split up and a significant divergence in sample frequency can yield unbalanced data sets. As figure 4.6 depicts, the filter varies over time but does not reach extremely low levels of data used. As expected, the filter carries more data points in times of crises, which are associated with high volatility.

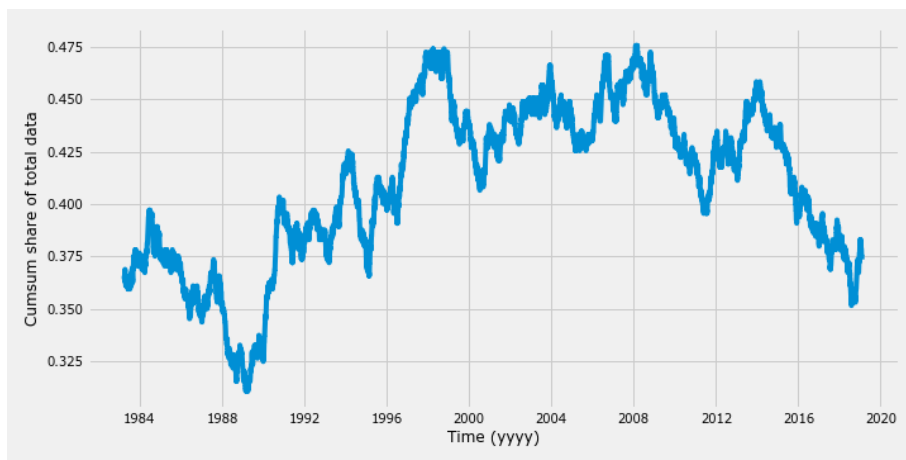


Figure 4.6: Illustrates the total share of data points included in the filtered data set compared to original data set. Generated by using a moving window of 640 datapoints, to represent roughly two and a half years of data.

4.4.2 Overlapping data

In machine learning, it is very important that there is no information leakage between the training set and test set. The complexity of accounting for information leakage increases when working with time series data as opposed to non-sequential data. When using cross-validation, and thus dividing the data set in different subsets, overlapping segments are unfortunately easily created. This can be demonstrated through an example. Consider a time series where the label y_t indicates the future return over a holding period of l days. Naturally, the label y_t is associated with a set of features X_t . To the point, at time t , the label y_t carries information from time $t, t + l$. More specifically, y_t carries partial information about the asset price

from $[t + 1, t + 2, \dots, t + l]$. This is problematic if the the training data includes the pair X_t and y_t and the training data is, for instance, cut at time t . Now observations associated with times $[t + 1, t + 2, \dots, t + l]$ are included in the test set and information leakage has occurred. To remedy this, the test set is truncated by removing l days in the beginning and the end of data set, a technique known as purging.



Figure 4.7: Visualisation of overlapping datapoints in a time-series

4.4.3 Training and test split

To enable model evaluation, the data was split into a train and a test set. The data was not split 50/50 or by any other arbitrary fraction. Instead, the data was split by plotting the data and employing a qualitative approach.

The underlying idea was to include time periods of both upward and downward trends. This notion follows naturally from the interpretation of a machine learning algorithm as a tool to recognise patterns. The inclusion of different market regimes in the training set should enable the model to perform better across different market environments when tested out-of-sample.

The date of the split was chosen to be the the 3rd of January 2005, the first business day of the year. This particular date was chosen for the financial crisis of 2008 to be included in the test set in order to challenge the model with a market in a strong downward trend. Furthermore, the date is after the start date of all futures contracts as depicted in table 4.1. With varying start dates, the amount of data and the diversity with respect to market behaviour will vary across the assets. Such discrepancies and its implications on model performance is further addressed in the discussion. Figure 4.8 and 4.9 exemplify plots used in the decision process where the presence of upward and downward trends in both the training and test set is established.

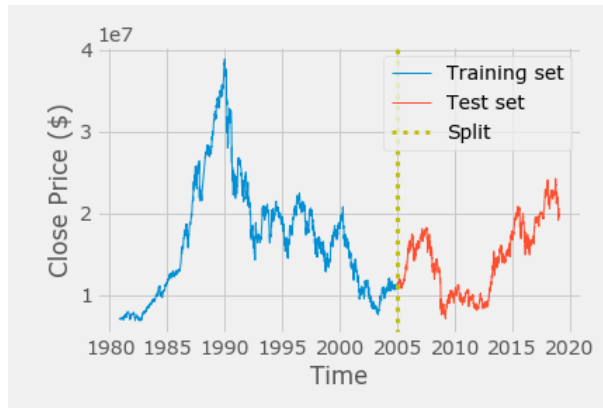


Figure 4.8: Closing price for *NK1*. Illustrating how the data is split between training set (blue) and test set (red).

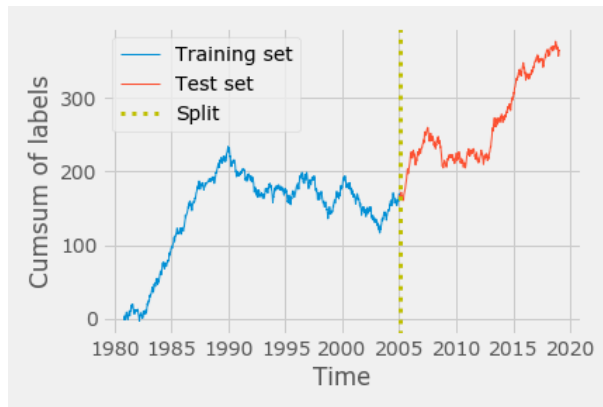


Figure 4.9: Cumulative sum of binary classification labels for *NK1*. Illustrating how the data is split between training set (blue) and test set (red).

4.5 Feature selection

Feature selection was based on using l_1 regularised logistic regression. Because the feature space was made up of relatively similar features, the hope was that the regularisation would be able to disqualify some of trading indicators, preferably a few of each type.

An appropriate value for the constant C in Equation 3.62 was determined by using ten-fold cross-validation as illustrated in figure 4.10 with Accuracy as the scoring method. The attempts at feature selection gave ambiguous results and was later dropped. The results and the reasoning behind this decision is further described in the results section.

4.6 Model selection

As described in the theory, there are many different machine learning algorithms. When deciding on which one to use, it is important to avoid selection bias. In this case, selection bias could involve testing the different algorithms on the test data and picking the one which yielded the highest return in a backtest. Assuming that an investor started using the algorithm in January of 2005, picking the best performing model would undermine the integrity of the test as the selection would have been made on data not yet available

in January of 2005. As such, a model selection process was conducted on data pre 2005, see section 4.4.3. The actual selection phase was broken down into three parts:

1. Initial comparison
2. Holding period and threshold decision
3. Hyper-parameter optimisation

This model selection was exclusively conducted for one futures contract, the *NK1*. This particular futures contract was chosen because it was the longest time series among the contracts. The choice of only including one contract in the model selection was based on the notion that the contracts are fairly homogeneous which means that the set-up should work similarly independent of the contract

Decisions between models and parameters were taken by qualitatively weighing the results of confusion matrices in combination with model evaluation metrics as defined in section 3.11. The findings from this model selection was subsequently carried over and applied to the rest of the assets.

4.6.1 Initial comparison

The goal of the initial comparison was to choose a single machine learning algorithm from the selection described in the theory. Algorithms were taken straight from scikit-learn's machine learning package and used with its default parameters. The initial comparison was conducted over a ten-fold cross-validation on the training set using a scheme where there was one alternating test fold. Model evaluation metrics and confusion matrices were calculated for each of the test folds. Initial conclusions could be supported by plotting the metric and the dispersion of metrics across all machine learning models and evaluation metrics. Furthermore, confusion matrices were used to analyse the in-sample accuracy when predicting the different labels. Each fold did not yield a separate plot, instead the results were aggregated across the 10 folds. As the purpose of this comparison was to create a short-list of promising algorithms, the test was limited to the binary case and a ten day holding period.

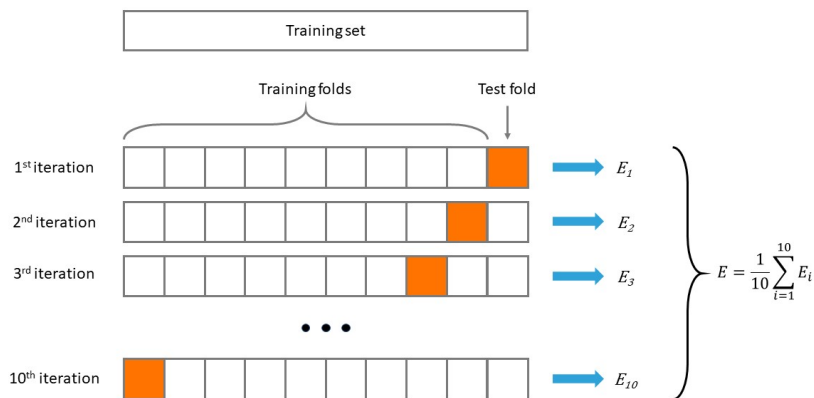


Figure 4.10: The process of cross-validation and method by which a mean of metrics coinciding with the different folds E_n can yield descriptive statistic E

4.6.2 Holding period and threshold decision

Having picked a machine learning algorithm from the first step, the arbitrarily chosen holding period and threshold from step one needed to be replaced with a more substantiated choice. All combination as limited by the values defined in section 4.3 were tested. Similarly to the previous step, a ten-fold cross-validation was

used. However, this time only confusion matrices were analysed. Optimal holding periods and thresholds were identified in the ternary case. In the binary case, suitable holding periods were chosen.

4.6.3 Hyper-parameter optimisation

Hyper-parameter optimisations were conducted in attempt to develop a better understanding of the model's sensitivity towards different parameters, model evaluation metrics and varying time periods. Another objective was to ensure an even distribution of accuracy across the confusion matrix. In other words find parameters such that the model does not rely on guessing the most frequent label. The hyper-parameter optimisation was conducted using two different optimisation techniques from scikit-learn's package: RandomizedSearchCV and GridSearchCV. Five independent searches were conducted with each optimisation using one of the five evaluation metrics established in the theory. Also, since GridSearchCV is computational heavy, each optimisation begun by applying RandomizedSearchCV on a wide range of values, in attempt to narrow the span of optimal parameter values.

As an example, for a given parameter 'n_estimators', the algorithm was fed random values from the array:

$$[20, 40, 60, 80, 100, 120, 140, 160, 180, 200]$$

Assuming an optimal value of 160 was returned this was then translated into an array:

$$[150, 160, 170]$$

which was subsequently fed to the GridSearchCV that finally returned the optimised values, say for example 150.

4.6.4 Initial test of final model

Having decided which models were most suitable as well as which parameter values these should use, two in sample test were conducted to asses in sample performance. Both performance metrics as well as model evaluation metrics were calculated using predictions generated from a ten fold cross-validation on the entire training data set. Not only were these produced for the optimised model but also for its non-optimised equivalent. The non-optimized model refers to the applicable algorithm with the standard parameters as used in Scikit-learn. [34]

As a final step, the model evaluation metrics were calculated by fitting the models on the entire training set and predicting once on the test set.

4.6.5 Feature importance

Similarly as in section 4.6.4 the feature importance was conducted using a ten fold cross-validation on the entire training data set for the final optimised model. Importance is represented as a percentage from 0-100 where 0% corresponds to no importance and 100% corresponds to that feature being the only important feature. Important/non-important features were selected by using a threshold of 0.04. A threshold motivated by the fact that there were 25 different features in total, implying an importance of 0.04 if they were to be split evenly across all features.

4.7 Model backtesting

4.7.1 Individual asset backtesting

Having selected a suitable model, backtests were conducted to measure the monetary performance over time. The three different kinds of backtests, as described in the theory, were used.

The thesis does not focus on portfolio optimisation, only the relative performance compared to other strategies and the underlying asset. As such, for simplicity, only one contract was assumed to have been bought

of each type of futures contract. Because only the returns are of interest, the variation in contract values and the different currencies as seen in table 4.1 will not cause issues. Because the contracts are considered in isolation, a discussion on the implication on margin, as described in section 2.2, is deemed outside the scope of this thesis.

When conducting the back test, for more realistic results, each trade was associated with a transaction cost of 0.25% of the notional value invested. This choice was inspired by established literature. [21]

Note that all tests were not conducted with all assets as some tests are more sensitive to lack of data. This will be further commented in the results.

4.7.1.1 Parameter choice

Two of the backtesting methods are associated with a choice of parameter. The traditional walk forward takes a parameter n which indicates the window length for the observations to be included in the training set, see figure 3.11. The parameter n was chosen to be 500 observations, inspired by established literature and scaled to account for cumsum filtering. [39]

The combinatorial backtesting was conducted by dividing the test data set into $N = 6$ smaller sets and partitioning over $k = 2$ different validation sets, thus yielding five different backtesting paths, as explained in the theory.

4.7.2 Consolidated asset backtesting

One final way of conducting backtesting was designed where all assets were consolidated into one single data set. The idea was to compensate for the lack of data, in assets such as *ES1* and *DM1*, by gathering data from all assets and consolidating them into one large data set. This large training set collected 10 years of data from all assets in the date range of 1995-01-01:2005-01-01. Subsequent to fitting a Random forest using the predetermined parameters on this training set, the model was tested on each asset's test data individually. Furthermore was the backtest iterated 100 times to form confidence intervals for each of the performance metrics.

4.8 Benchmarking

The benchmarks were evaluated using the performance metrics defined in the theory 3.13.1.

One of benchmarks used to evaluate the strategies was the underlying asset corresponding to the each strategy. If for instance a model is built around the futures contract *NK1*, the model was compared to the results if one would simply buy-and-hold *NK1*.

In order to evaluate the usefulness of the machine learning algorithm in the context of technical analysis, the models were also compared to a traditional moving average strategy as described in the theory 3.8.4. To ensure a fair comparison, the parameters were equivalent to those used to create the features as defined in table 4.3. Going forward, this benchmark will be referred to as the "Simple trading strategy".

For all backtesting methods, the value of the strategy was recorded daily and not only when rebalanced. This was motivated by the varying frequency of rebalancing days across the backtesting methods. Not keeping track of daily returns could cause problems because several of the performance metrics are affected by the frequency at which returns are recorded. Working with daily returns consistently ensured fair comparison across strategies and assets.

Chapter 5

Results

5.1 Feature selection

As previously mentioned, the feature selection process was not very successful. The reason for the result was hypothesised to be twofold. First, the information in the data was deemed to be low, leading to a somewhat arbitrary choice of features. This phenomena was exacerbated by the similarity of the features. Secondly, as the market changes, so does the relative importance of the features. For instance, if the market shifts, features based on short-term trading indicators should be favoured as the lag is less prevalent in such signals.

Having been unable to make sense of the incoherent result for any of the assets in neither the binary nor the ternary case, all features were kept.

5.2 Model selection

The reader is reminded that only the asset, *NKI*, is considered in the model selection since the objective was to find the most suitable machine learning algorithm and its parameters.

5.2.1 Initial model comparison

The initial comparison resulted in diverse scores across the algorithms and model evaluation metrics. When examining the evaluation metrics, a high mean and a low variance is preferred as it indicates high performance as well as consistency across the cross-validated folds. Considering the different metrics, Random forest and SVM were deemed most promising. Figure 5.1 depicts how the SVM consistently achieved a lower variance compared to the Random forest as well as higher average scores on several of the evaluation metrics. Consequently, from a pure model evaluation metric perspective, SVM seemed to be the best performing model.

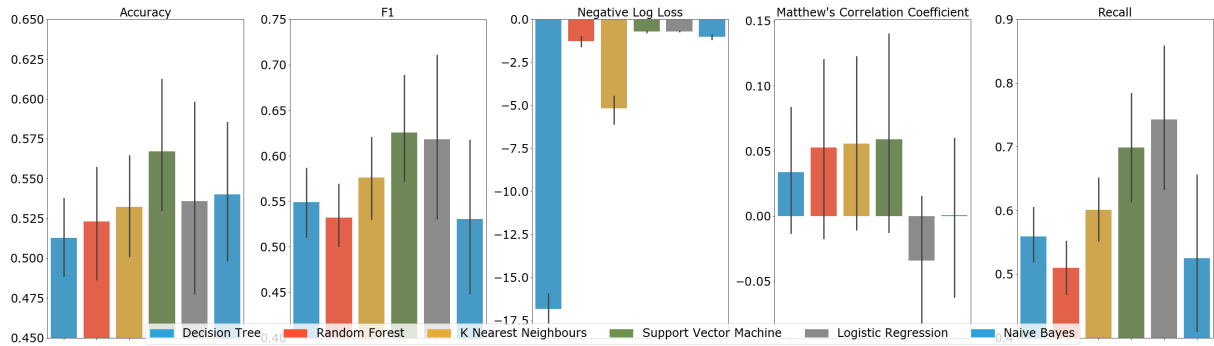


Figure 5.1: Bar charts displaying the mean of each model evaluation metric and the dispersion of values across the 10 folds in the cross-validation. The values were calculated based on a binary representation of the algorithm on *NK1* with a holding period of 10 days.

In addition to the bar chart, confusion matrices were used to gain insight into how each model predicts the different class labels. K-Nearest-Neighbours showed promising results in figure 5.1. However, as depicted in figure 5.2, the algorithm was worse than random at predicting downward moves (-1). The same holds true for SVM and logistic regression where the imbalance is even more significant. The imbalance implicates that the high Accuracy and Recall achieved by the two algorithms stems from an ability to successfully predict upward movements. An ideal model would have the ability to predict both upward and downward movements, rather than guessing one class label. Random forest on the other hand appears relatively balanced over class predictions. From this point, Random forest and SVM were deemed of most interest; with the hope of balancing the SVM while retaining some of its ability successfully predicting upward movements.

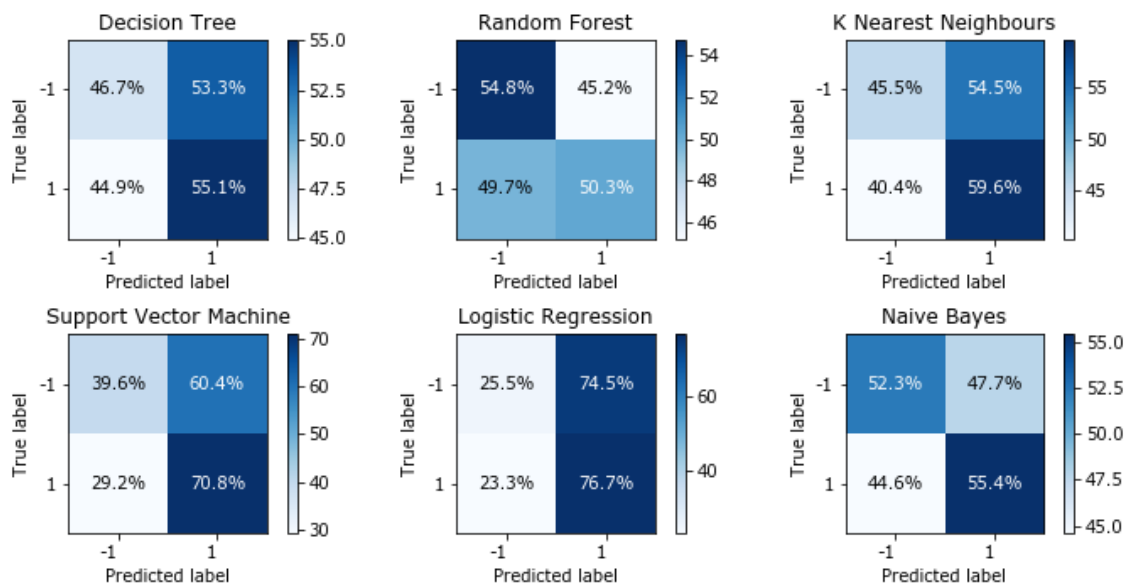


Figure 5.2: Confusion matrices depicting the predictions across labels and the actual values for a range of machine learning algorithms. The values were calculated based on a binary representation of the algorithm on *NK1* with a holding period of 10 days.

5.2.2 Holdingperiod and threshold decision

Since the initial comparison did not clearly establish which algorithm to use, the remainder of the model selection process was iterative in nature. In the end, the final decision fell on the Random forest. For the

sake of simplicity, this section has been written with the benefit of foresight and all analysis is based on using Random forest. This decision was solely motivated by avoiding redundant expositions and the reasoning for disqualifying the SVM will be established later.

First, the distributions between class labels was analysed. In the binary case, the class distribution remained fairly constant across the different holding periods, with a slightly higher share of upward movements, approximately 55%.

In the ternary case, as depicted in figure 5.3, the distribution between class labels varied significantly. As expected, the total share of labels classified as flat (0) increased with the threshold. This applied to all holding periods [5,10,20]. Furthermore, the share of upward movements (1) was consistently higher than the share of downward movements (-1), a phenomenon to be expected due to the upward drift in the market as well as the effect of drawdowns (comparatively higher positive returns are needed to recoup the losses associated with negative returns). A threshold of 5% split the data to such an extent that no upward nor downward movements could be predicted in the cross-validation. The 5% threshold was consequently excluded from the results.

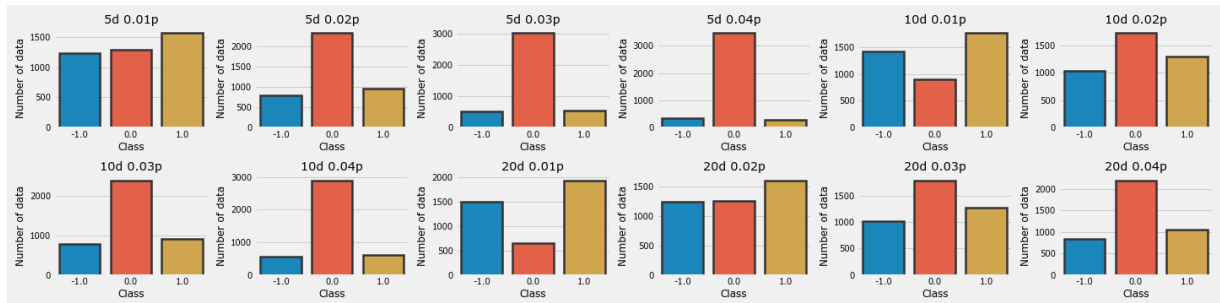


Figure 5.3: The distribution of downward movements (-1) (blue), no movement (0) (red) and upward movements (1) (yellow) for *NK1* under the training data.

For each combination of holding period and threshold, confusion matrices were generated in the same manner as in the Initial Comparison. In the binary case, both 10 and 20 day holding periods yielded positive results. In the end, 20 days was considered the optimal split because of slightly better confusion matrix, see figure 5.4

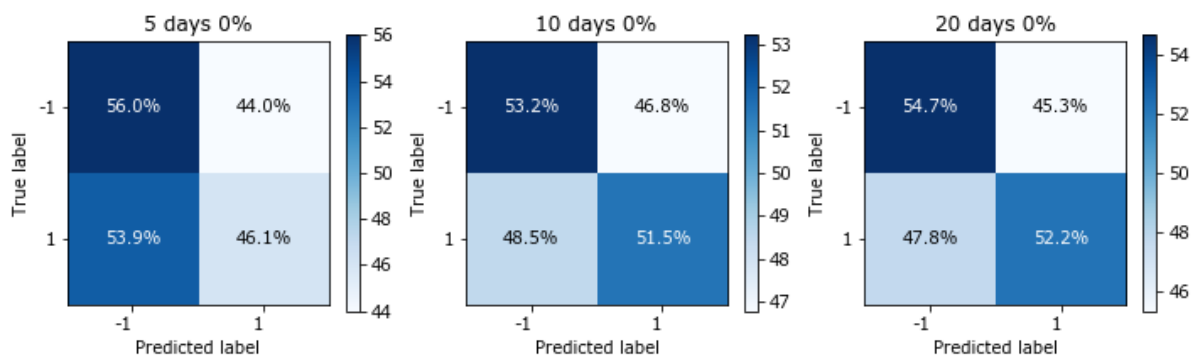


Figure 5.4: Confusion matrices generated from cross-validation on the training set for the three different holding periods 5, 10 and 20 days in the binary case.

An analog analysis for the ternary case demonstrated that an increasing threshold coincided with an increasing accuracy for flat predictions across all holding periods, see figure 5.5. The optimal holdingperiod and threshold in the ternary case was chosen to be 10 days and 2 %. The decision was made on the basis of

keeping an even distribution between class labels while ensuring that the accuracy of downward movements outweighed the rate of false upward movement predictions and vice versa.

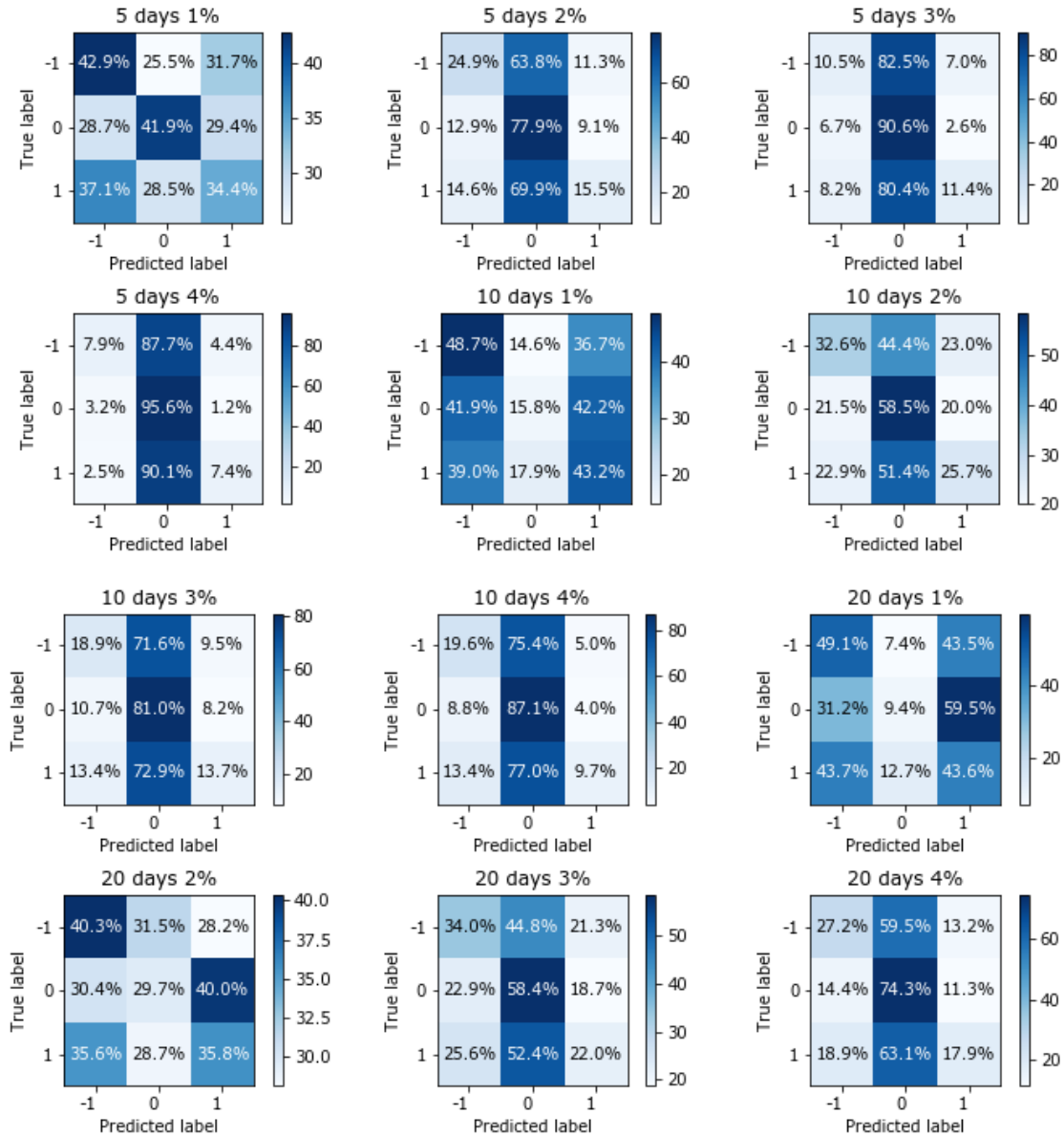


Figure 5.5: Confusion matrices generated from cross-validation on the training set all combinations of hold-ingroups and thresholds in the ternary case.

5.2.3 Hyper-parameter optimisation

The goal of the hyper-parameter optimisation was to further enhance model performance. Due to the inability to select a single machine learning algorithm in the initial comparison, this section consists of two different optimisations, one for SVM one for Random forest. A list of parameters for each algorithm and their implications can be found in the Scikit-learn documentation for `sklearn.svm.SVC` and `sklearn.ensemble.RandomForestClassifier`. [34]

5.2.3.1 Support Vector Machine

The initial comparison suggested that the SVM was the best performing model, though with a higher accuracy for upward movements than downward movements. This raised the question; to what extent was the model simply guessing the most common class. Furthermore, could this be addressed through hyper-parameter optimisation.

The tendency of the SVM to emphasis on the most common class was found to be a complicated issue to address. In the end, optimisation in neither the binary nor the ternary case helped to address the imbalance issue.

5.2.3.2 Random forest

For the binary case the hyper-parameter optimisation was successful in keeping accuracy levels for both classes above 50% regardless of optimisation method chosen. Figure 5.6 also presents Matthews Correlation Coefficient as the best optimised model, although all models produced desirable results.

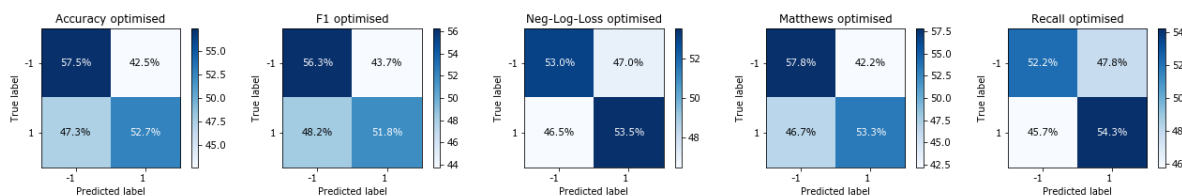


Figure 5.6: Binary confusion matrices generated from an optimised Random forest using each respective evaluation metric.

Similar results were found in the ternary case, all models produced desirable results as the total accuracy was above guessing rate (33%), see figure 5.7. The final decision fell on the Matthews optimised model due to a superior total accuracy of (42.5%).

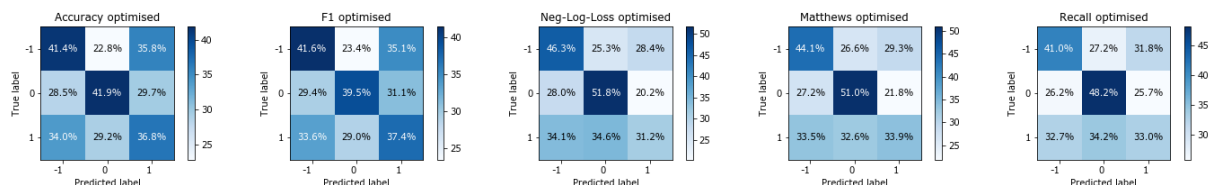


Figure 5.7: Ternary confusion matrices generated from an optimised Random forest using each respective evaluation metric.

The aim of the hyper-parameter optimisation section was to derive what optimisation method was most suitable and which parameter values to use in the backtesting. After having discarded the SVM it was established that optimising a Random forest using Matthews correlation coefficient was the optimal decision in both the binary and ternary case.

5.2.4 Initial test of final model

Having decided on a machine learning algorithm and optimising its parameters, the next step was to see how the optimised model performed compared the non-optimised model. To make a comprehensive comparison, a backtest using the predictions generated from the cross-validation on the training set was conducted and illustrated in figure 5.8. It showed that the the optimised models outperformed its non-optimised equivalents, both in the binary and ternary case. However, only the optimised ternary model outperformed NK1 itself. See table 5.1. Interestingly, the optimised binary model had a significantly larger gross return but was more volatile which resulted in a lower Sharpe ratio and Sortino ratio. Furthermore, the binary model had larger

maximum drawdown compared to the asset.

Table 5.1: Performance metrics generated from a ten fold cross-validation on the training data set. Bolded figures indicates outperforming the asset benchmark.

	Non-Optimised		Optimised		Benchmark
	<i>Binary</i>	<i>Ternary</i>	<i>Binary</i>	<i>Ternary</i>	<i>Asset</i>
<i>Sharpe ratio</i>	-0.23	-0.06	0.15	0.30	0.24
<i>Sortino ratio</i>	-0.33	-0.09	0.22	0.42	0.34
<i>Maximum drawdown</i>	0.95	0.83	0.83	0.61	0.82
<i>Gross return</i>	-0.76	0.37	5.18	1.59	0.62

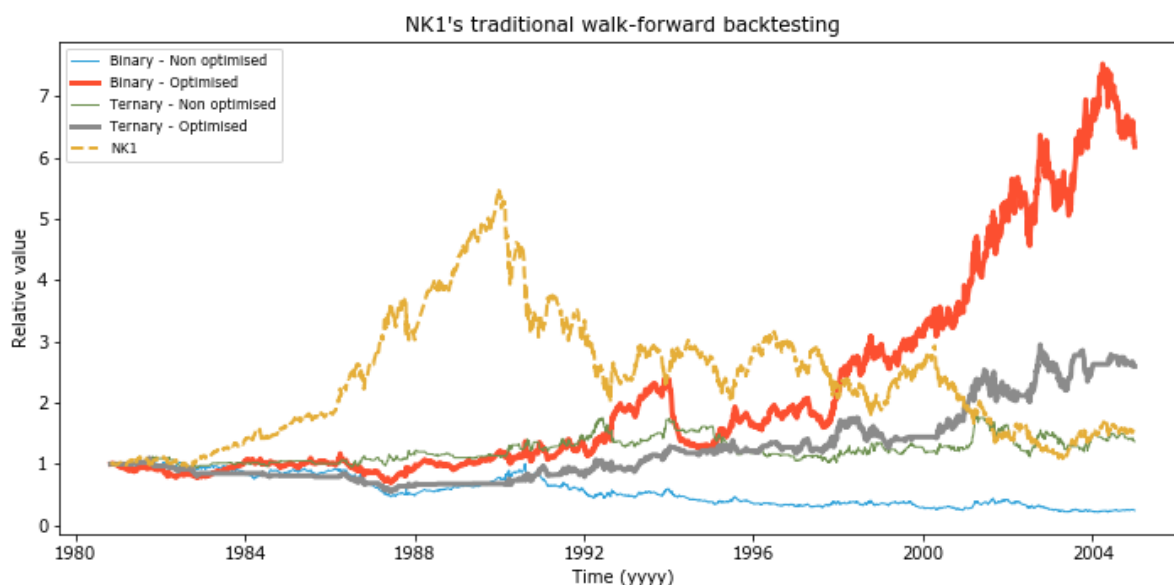


Figure 5.8: Backtesting of *NK1* for the binary and ternary cases on the cross-validated training set.

The result indicated, at least in the ternary case, that the strategy could be profitable over time. Before moving on to actual backtests on the test data, a comparison of the two models, optimised and non-optimised, was conducted using the evaluation metrics. The models were evaluated in two ways. First, by use of cross-validation on the training set. Second, the models were fitted on the entire training data and evaluated on the test data.

As expected, the Matthews optimised model outperformed the non-optimised model in every evaluation metric on the training data in both the ternary and the binary case, see figures 5.9:5.10. However when evaluated on the test data, the two models performed similarly except for the Negative log loss suggesting that the optimised model is superior in weighing probabilities for its predictions

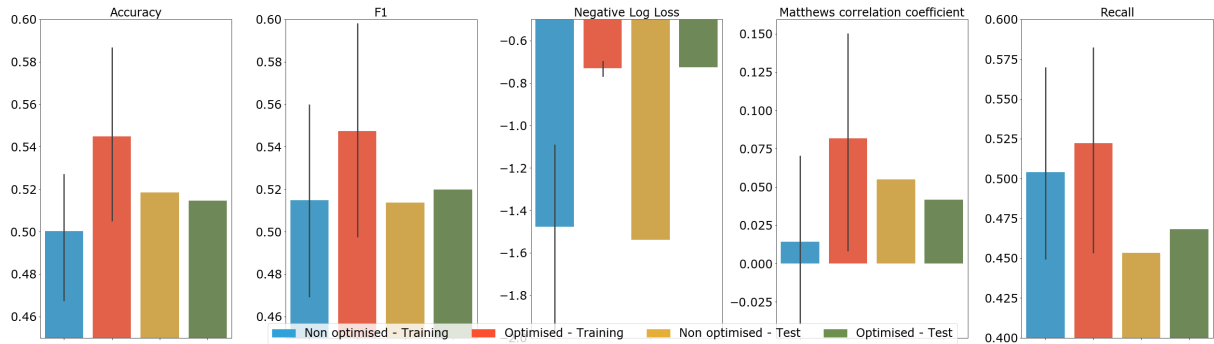


Figure 5.9: Model evaluation metrics for the final optimised Random forest versus a non optimised model in the binary case.

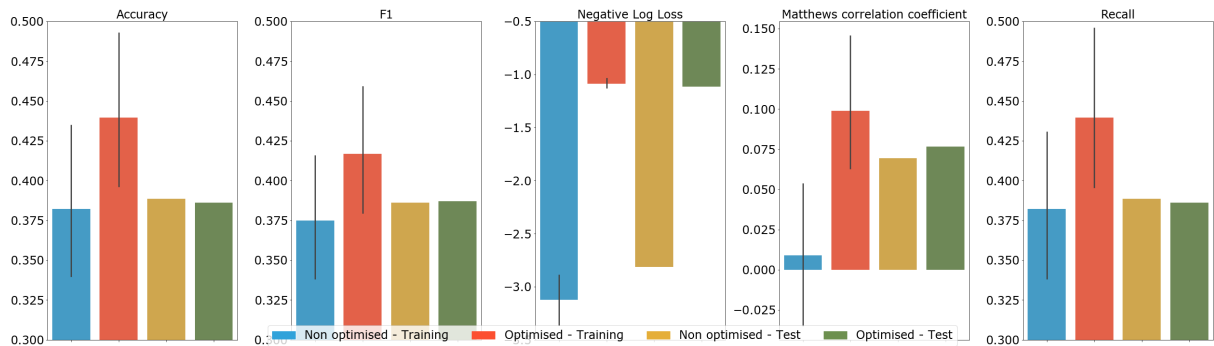


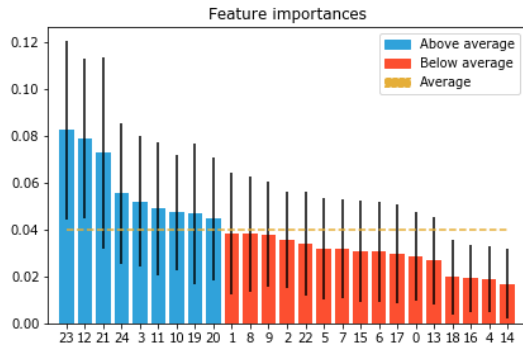
Figure 5.10: Model evaluation metrics for the final optimised Random forest versus a non optimised model in the ternary case.

5.2.5 Feature importance

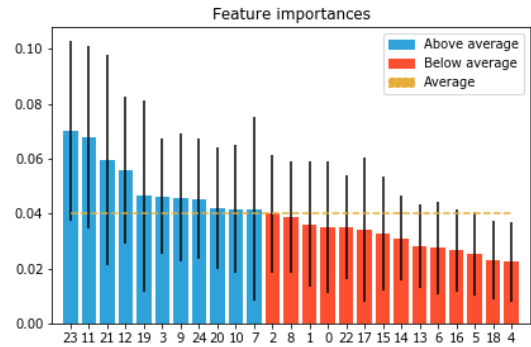
The feature importance shows which features were used most frequently by the random forest. As it was difficult to perform feature selection, the results were expected to be hard to interpret.

In both the binary and the ternary case there were features that were more important than others. Interestingly, # 23, corresponding to 95;195d MACD was chosen as most important in both scenarios, see table 5.2. Furthermore, the algorithm seems to favour features based on strategies capturing longer trends as evident from the high values in the *Parameter* column of table 5.2.

The feature importance had a higher variance in the ternary case compared to the binary case, which resulted in lower importance values and more features above the 0.04 threshold, see figure 5.11. A high variance indicates that different features are important for different folds in the cross-validation. It is worth noting that almost all features could land either above or below the average threshold in a single fold.



(a) Depicts the results from the binary case



(b) Depicts the results from the ternary case

Figure 5.11: Feature importances from the ten-folded cross-validation for *NK1*. Depicts mean values as well as the variance over the ten folds. The best performing feature's names corresponding to each number can be found in table 5.2.

Table 5.2: The features exceeding the average importance of 0.04 and their corresponding id as used in figure 5.11.

Binary				Ternary			
#	Parameter	Type	Importance	#	Parameter	Type	Importance
23	95;195d	MACD	0.082	23	95;195d	MACD	0.070
12	108;216d	MA	0.079	11	72;216d	MA	0.068
21	60;130d	MACD	0.073	21	60;130d	MACD	0.059
24	95;195d	MACD diff	0.056	12	108;216d	MA	0.056
3	216d	HP cross	0.052	9	36;72d	MA	0.046
11	72;216d	MA	0.049	3	216d	HP cross	0.046
19	30;95d	MACD	0.047	19	30;95d	MACD	0.046
10	36;216d	MA	0.047	24	95;195d	MACD diff	0.045
20	30;95d	MACD diff	0.045	7	12;72d	MA	0.042
				10	36;216d	MA	0.042
				20	30;95d	MACD diff	0.042

5.3 Backtesting and Benchmarking

Adding to the underlying asset, a Simple trading strategy was used as a benchmark. The results of the benchmark with respect to the strategy evaluation metrics are included in tables 5.3 and 5.4. For convenience, the performance of the assets were also included in the tables.

In general, the Simple trading strategies performed significantly worse than the underlying asset. However, all assets had at least one Simple trading strategy with lower maximum drawdown. In fact, strategies based on only one asset, (*Z 1*), had, on average, significantly worse maximum drawdown compared to the underlying asset. Furthermore, there were individual cases, *CF1*, where the Simple trading strategy outperformed the underlying asset on all performance metrics. However the average scores were always lower.

The results indicates that the Simple trading strategies did not generalise well over all assets and that they are not reliable strategies on their own. However, they show signs of being able to reduce the maximum drawdown.

Table 5.3: Minimum, average and maximum Sharpe ratio and Sortino ratio for the Simple trading strategies for each asset. Figures in bold indicate outperforming the underlying asset.

	Sharpe ratio				Sortino ratio			
	<i>Min</i>	<i>Average</i>	<i>Max</i>	<i>Asset</i>	<i>Min</i>	<i>Average</i>	<i>Max</i>	<i>Asset</i>
<i>Z 1</i>	-0.42	-0.17	0.19	0.25	-0.58	-0.24	0.27	0.35
<i>ES1</i>	-0.09	0.16	0.38	0.41	-0.13	0.22	0.53	0.57
<i>NQ1</i>	-0.27	0.02	0.21	0.62	-0.37	0.03	0.28	0.87
<i>GX1</i>	-0.15	0.10	0.31	0.44	-0.2	0.14	0.43	0.62
<i>NK1</i>	-0.23	0.08	0.24	0.30	-0.32	0.11	0.34	0.42
<i>CF1</i>	-0.26	-0.01	0.26	0.22	-0.36	-0.01	0.37	0.31
<i>DM1</i>	-0.17	0.11	0.40	0.44	-0.23	0.15	0.56	0.63

Table 5.4: Minimum, average and maximum Gross return as well as Maximum drawdown for the Simple trading strategies for each asset. Figures in bold indicate outperforming the underlying asset.

	Maximum drawdown				Gross return			
	<i>Min</i>	<i>Average</i>	<i>Max</i>	<i>Asset</i>	<i>Min</i>	<i>Average</i>	<i>Max</i>	<i>Asset</i>
<i>Z 1</i>	0.38	0.64	0.82	0.48	-0.71	-0.36	0.38	0.44
<i>ES1</i>	0.29	0.40	0.53	0.57	-0.41	0.33	1.35	1.22
<i>NQ1</i>	0.39	0.53	0.79	0.54	-0.63	-0.10	0.48	3.35
<i>GX1</i>	0.44	0.55	0.74	0.55	-0.53	0.15	1.00	1.60
<i>NK1</i>	0.44	0.62	0.81	0.61	-0.66	0.08	0.65	0.81
<i>CF1</i>	0.43	0.62	0.81	0.59	-0.69	-0.10	0.81	0.29
<i>DM1</i>	0.30	0.44	0.57	0.54	-0.47	0.19	1.35	1.35

When benchmarking, two Simple trading strategy benchmarks were included for comparison. The best and worst performing strategies with respect to the Sharpe ratio. For visualisation of the performance over time for each Simple trading strategy, see Appendix B.

Furthermore, as established in the initial comparison, the settings in table 5.5 were used for the Random forest.

Table 5.5: Holdingperiod and respective threshold for the binary as well as the ternary case. Also lists the parameter values used for scikit-learn’s Random forest function.

	<i>Binary</i>	<i>Ternary</i>
<i> Holding period</i>	20	10
<i> Threshold</i>	0%	2%
<i> n_ estimators</i>	160	160
<i> min_ samples_ split</i>	4	4
<i> min_ samples_ leaf</i>	4	4
<i> max_ features</i>	auto	sqrt
<i> max_ depth</i>	8	8
<i> bootstrap</i>	False	True

5.3.1 Traditional Walk-Forward

The performance metrics from the traditional walk-forward backtest were worse than the underlying asset in most cases except for the binary case with *GX1*, see table 5.6.

Table 5.6: Performance metrics generated from the traditional walk-forward backtest. As a benchmark, the performance of the underlying asset is included. B = Binary, T = Ternary, A = Asset. Figures in bold indicates outperforming the underlying asset.

	Sharpe ratio			Sortino ratio			Maximum drawdown			Gross return		
	<i>B</i>	<i>T</i>	<i>A</i>	<i>B</i>	<i>T</i>	<i>A</i>	<i>B</i>	<i>T</i>	<i>A</i>	<i>B</i>	<i>T</i>	<i>A</i>
<i>Z1</i>	-0.13	-0.01	0.25	-0.18	-0.01	0.35	0.64	0.58	0.48	-0.40	-0.13	0.44
<i>ES1</i>	-0.25	-0.31	0.41	-0.35	-0.40	0.57	0.75	0.71	0.57	-0.62	-0.59	1.22
<i>NQ1</i>	-0.16	-0.08	0.62	-0.23	-0.11	0.87	0.80	0.64	0.54	-0.56	-0.31	3.35
<i>GX1</i>	0.46	0.08	0.44	0.67	0.12	0.62	0.44	0.54	0.55	1.91	-0.05	1.60
<i>NK1</i>	-0.13	-0.21	0.30	-0.19	-0.29	0.42	0.64	0.68	0.61	-0.53	-0.63	0.81
<i>CF1</i>	-0.38	0.10	0.22	-0.51	0.14	0.31	0.83	0.42	0.59	-0.75	0.03	0.29

Given that the the models were trained on a moving window of 500 data points, one might claim that this method reflects the most current market conditions. However, when tested, all models failed to recognise large market selloffs such as the financial crisis. Figure 5.12 and 5.13 illustrates how *NK1*'s and *GX1*'s backtests performed over time. *NK1* struggled consistently and the binary model of *GX1* performed very well in many parts of the backtest. However, the latter failed to predict several sell-offs post 2012. Analog illustrations for the remaining assets can be found in Appendix C.

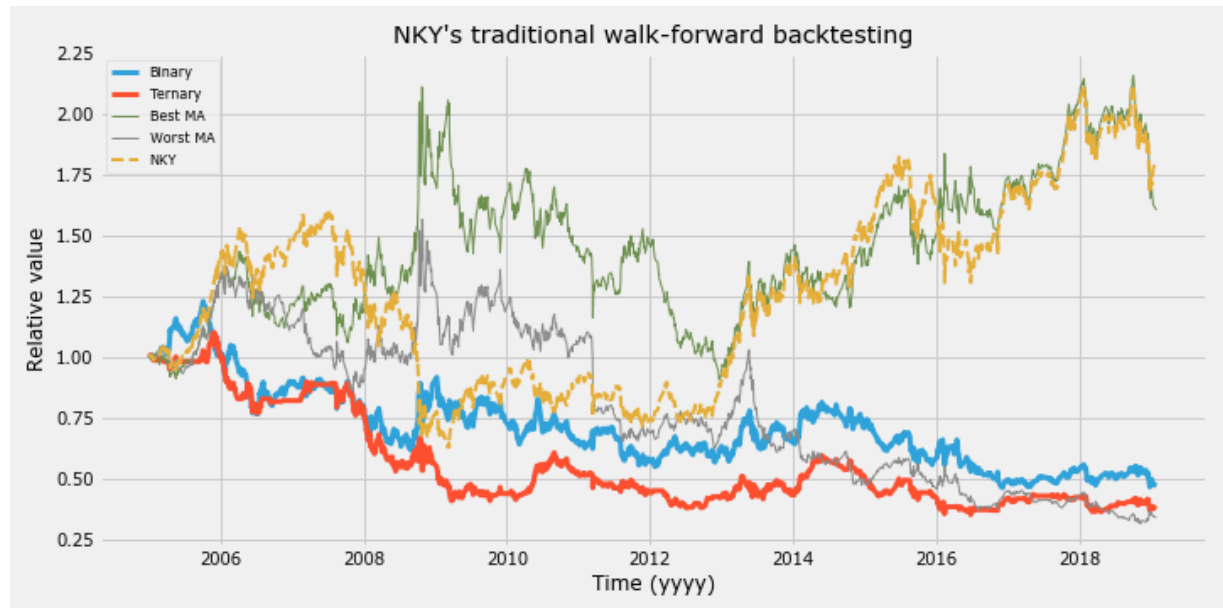


Figure 5.12: Traditional walk-forward backtesting of *NK1* for the binary and ternary cases. The relative value of *NK1*'s contract value as well as the corresponding best and worst performing benchmarks are also depicted.

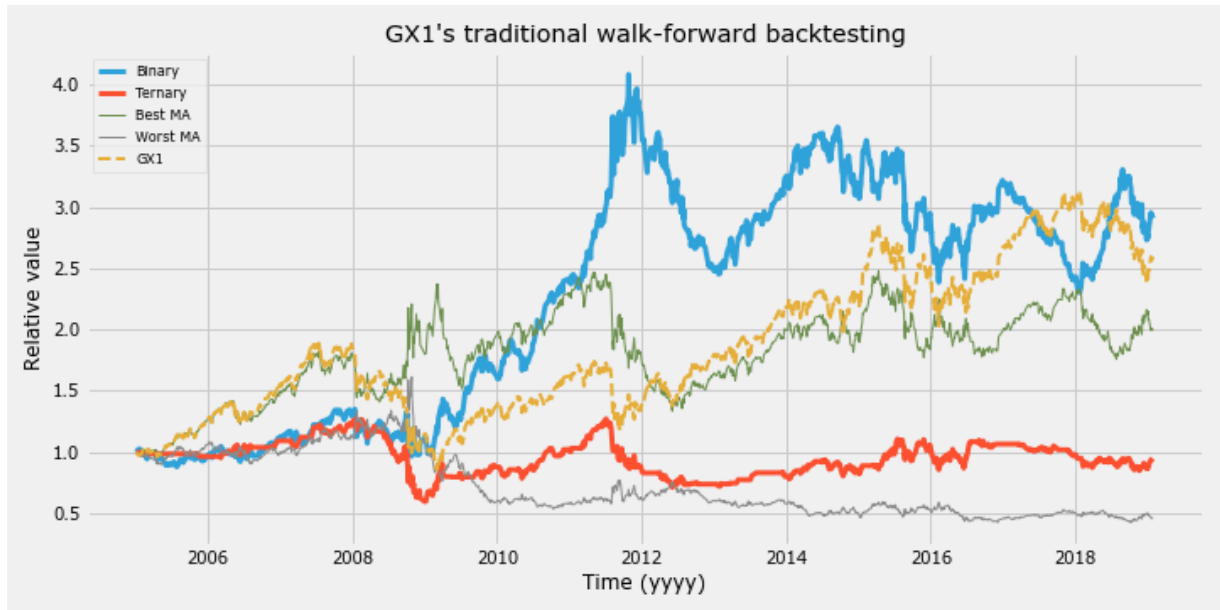


Figure 5.13: Traditional walk-forward backtesting of *GX1* for the binary and ternary cases. The relative value of *GX1*'s contract value as well as the corresponding best and worst performing benchmarks are also depicted.

In an attempt to qualitatively explain why almost all strategies failed, the 500 data points prior to 2005 and 2009 were plotted, see figure 5.14 and 5.15. Prior to 2005, the assets experienced a period of declining prices followed by a recovery. This could explain the passiveness of the models during 2006 to 2008 as they had not been trained on data which included strong upward momentum. When the crisis of 2008 starts, the window had been moved and excluded the sell-off in 2001 to 2003. As such, it is not surprising that none of the models successfully predicted the financial crisis of 2008. From figure 5.15 it can be observed that the relative asset value of *GX1* had a different magnitude compared to the other assets. This could have had an impact on the training data making the momentum more evident and, as a result, yielding a model able to outperform the underlying asset.

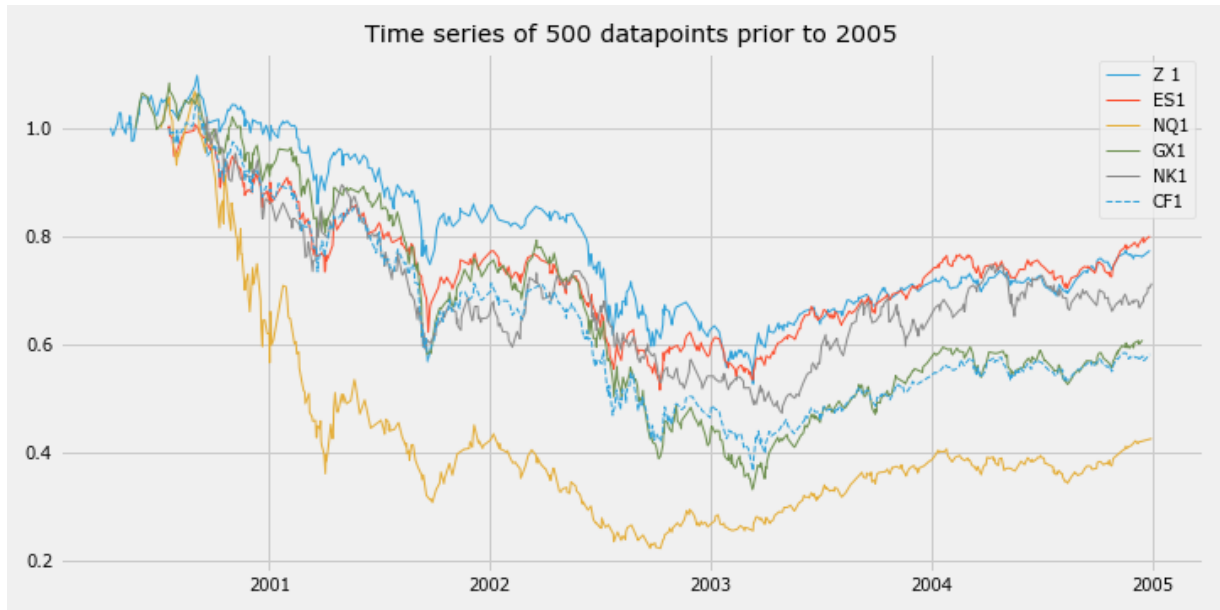


Figure 5.14: Relative contract values for each respective asset generated on 500 data points prior to 2005

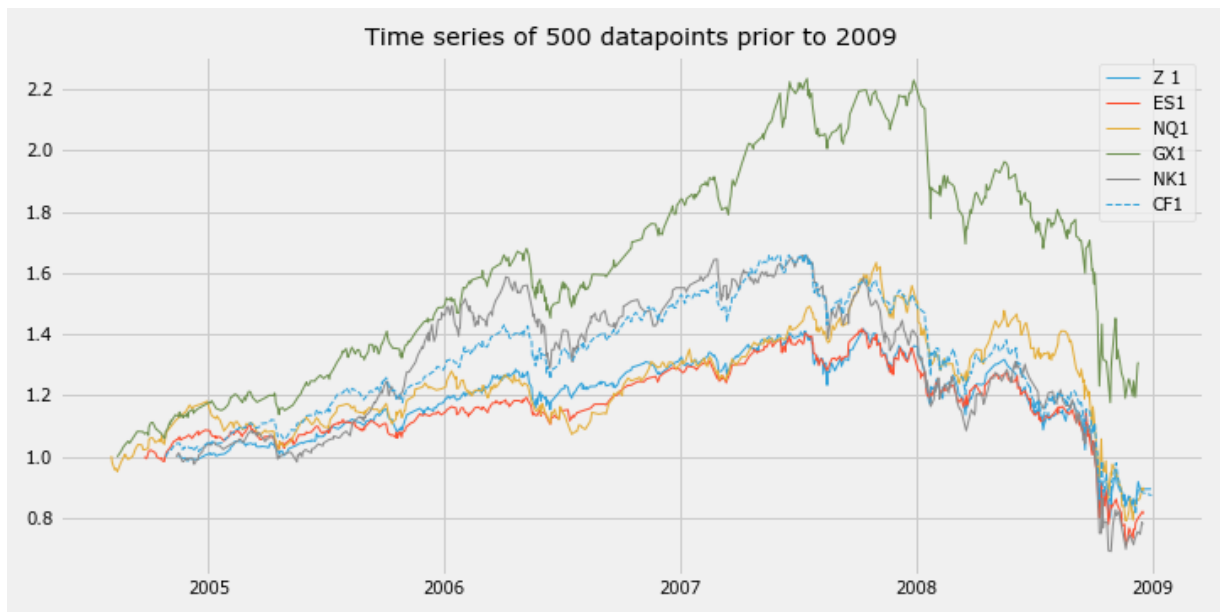


Figure 5.15: Relative contract values for each respective asset generated on 500 data points prior to 2009

The results illustrates the importance for the machine learning model to have seen market regimes in order to successfully predict the regime. As such, it was decided to perform warm-up walk-forward and combinatorial backtesting on assets with enough historical data. The assets deemed appropriate for these types of backtests were: *NK1*, *Z 1*, *GX1* and *CF1*.

5.3.2 Warm-up Walk-Forward Backtesting

The warm-up walk-forward performed significantly better than its traditional counterpart across all assets. In this test, both *NK1* and *Z 1* outperformed their respective benchmark assets.

Table 5.7: Performance scores for the assets *NK1*, *Z 1*, *GX1* and *CF1* generated from their respective warmup walk-forward backtest. As a benchmark, the performance of the underlying asset is included. B = Binary, T = Ternary, A = Asset. Figures in bold indicates outperforming the underlying asset.

	NK1			Z1			GX1			CF1		
	B	T	A	B	T	A	B	T	A	B	T	A
<i>Sharpe ratio</i>	0.37	0.34	0.30	0.05	0.41	0.25	0.71	0.02	0.44	0.04	0.12	0.21
<i>Sortino ratio</i>	0.54	0.50	0.42	0.07	0.62	0.35	1.05	0.00	0.62	0.05	0.17	0.30
<i>Maximum drawdown</i>	0.40	0.39	0.61	0.50	0.28	0.48	0.46	0.62	0.55	0.51	0.44	0.59
<i>Gross return</i>	1.60	1.08	0.81	-0.05	1.13	0.44	5.79	-0.20	1.60	-0.10	0.19	0.30

In the case of *NK1*, all performance metrics favoured both the binary and the ternary models over the underlying asset. In figure 5.16, both models appeared able to either capitalise from or by avoiding drawdowns. This included the financial crisis and the market sell-off in 2015 to 2016. However, both strategies failed to predict the sharp market reversal during the second half of 2018.

For *Z 1*, only the ternary strategy outperformed the asset. The converse holds true for *GX1* where the binary strategy outperformed the asset. The binary representation of *GX1* performed better in the warm-up walk forward compared to the traditional backtest.

CF1 still underperformed. However, even though neither the ternary nor the binary model managed to outperform the asset, the results were far less disappointing compared to the traditional walk-forward method. It is also worth mentioning that the best Simple trading strategy performed very well and beat the underlying asset.

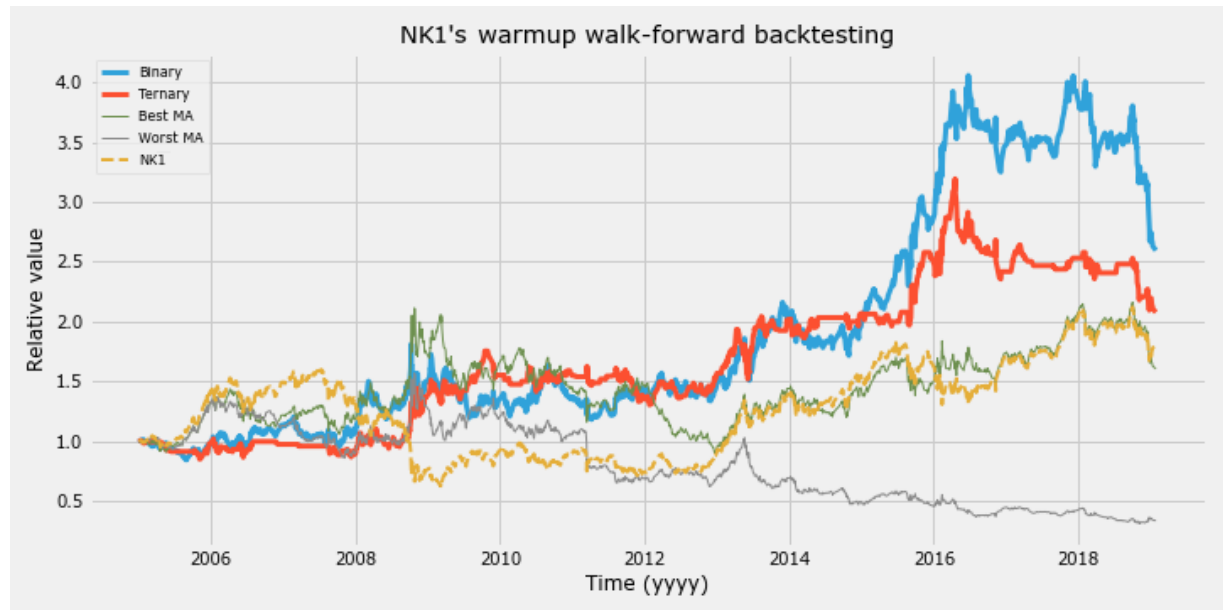


Figure 5.16: Warmup walk-forward backtesting of *NK1* for the binary and ternary cases. The relative value of *NK1*'s contract value and the corresponding best and worst performing Simple trading strategies are also depicted.

As expected, the results indicated that the models performed better when trained on a larger amount of data. However, the optimal amount of data has not been established. The inconsistent results also raised the question what the distribution of class labels looked like for successful and unsuccessful backtests. To shed some light on the issue, confusion matrices were used.

Figure 5.17 illustrates that poor performance, in the ternary case, was correlated with a high degree of

predictions from the opposite position, i.e 1 instead of -1. The opposite held true as *NK1* and *Z 1* had well balanced confusion matrices combined with successful backtests.

As seen in figure 5.18, the same conclusion could not be drawn in the binary case. Neither asset yielded a confusion matrix where the accuracy of predictions of downward movements exceeded 50%. As no reliable pattern was discerned for the different assets, their confusion matrices and the performance, further investigation was conducted in attempt to explain this behaviour.

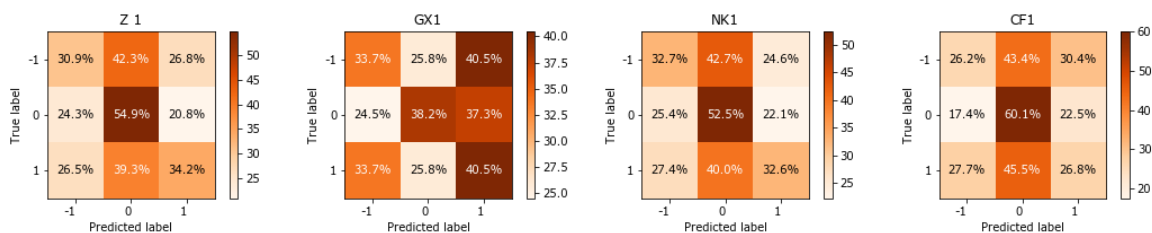


Figure 5.17: Ternary confusion matrices generated from the warm-up walk-forward backtests for *Z 1*, *GX1*, *NK1* and *CF1*

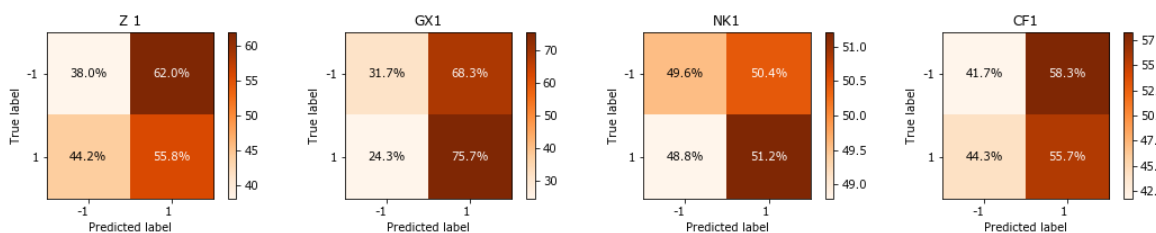


Figure 5.18: Binary confusion matrices generated from the warm-up walk-forward backtests for *Z 1*, *GX1*, *NK1* and *CF1*

Interestingly, *GX1* accurately predicted 75.7% of upward movements but only managed to predict 31.6% of downward movements. Regardless, it outperformed the underlying asset by far with a gross return of almost 580% compared to 160% for the asset. A reason for this phenomenon could be that the model was successful at predicting the largest movements and that a substantial part of false predictions were made up of smaller movements, thus resulting in small losses.

This hypothesis was tested by plotting histograms of the return series for each category: true negatives, false negatives, true positives and false positives. As Figures 5.19 and 5.20 depicts, both *NK1* and *GX1* had larger negative tails for their true negatives compared to their false positives. This, in combination with having a lower mean (i.e more negative and much larger losses) supported the hypothesis that their share of downward predictions were the ones resulting in the largest losses, and thus being most critical to predict accurately. Furthermore, *Z 1* and *CF1* appear to show equal distribution between the different measurements and similar average returns for the true negative/false positive and false negative/true positive. While this phenomenon did not explain why the model failed in the first place, it gave an indication of what characteristics are necessary for a model to succeed.

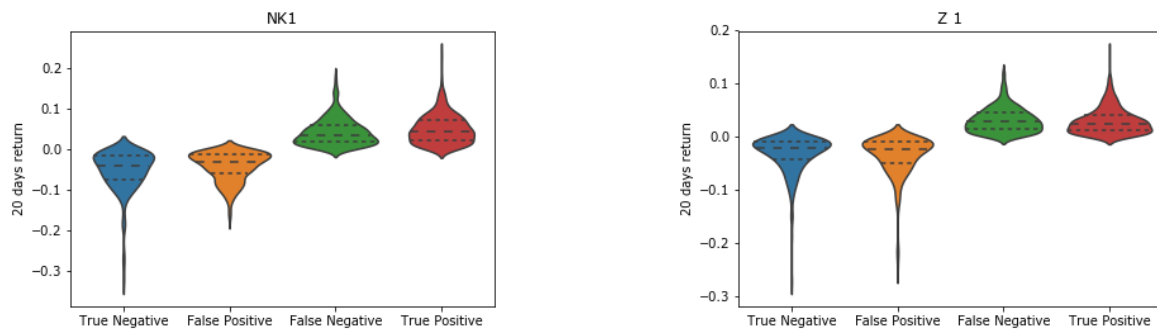


Figure 5.19: Two graphs showing the violin plots of the returns from each of the four groups true negative (blue), false positive (yellow), false negative (green) and true positive (red). Predictions are generated from the backtest corresponding to each respective asset: *NK1* and *Z 1*. The dashed lines denotes the quantiles.

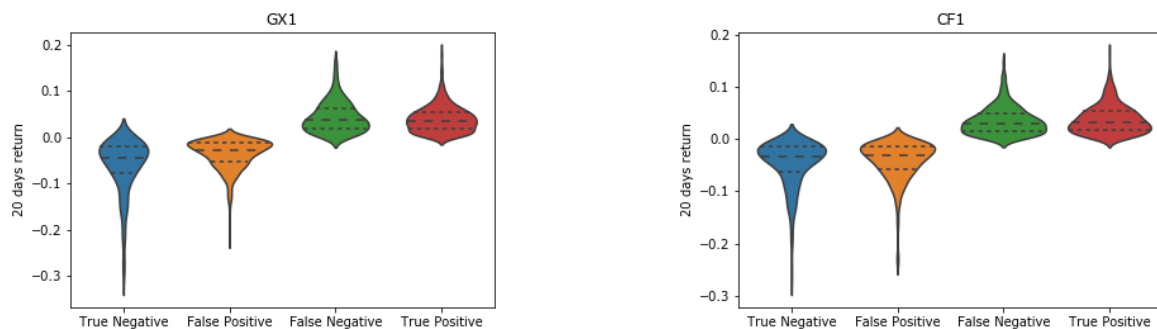


Figure 5.20: Two graphs showing the violin plots of the returns from each of the four groups true negative (blue), false positive (yellow), false negative (green) and true positive (red). Predictions are generated from the backtest corresponding to each respective asset: *GX1* and *CF1*. The dashed lines denotes the quantiles.

5.3.3 Combinatorial

Following the warm-up walk forward, it would be interesting to investigate how the models would have performed, had the history played out differently.

In figures 5.21 and 5.22, the performance metrics vary a lot with different paths. All assets except *Z 1* produced paths that both outperformed and under-performed its asset benchmark. Interestingly, *Z 1* yielded five paths in the ternary case which all outperformed the asset benchmark. On the contrary it only generated under-performing paths in the binary case.

The findings from the different combinatorial backtests suggested that there might be possible to find a holding period and threshold for certain assets that yields reliable results for future market movements. The results also indicates that if the parameters are not carefully chosen, the results can vary drastically. Note that the ternary models, once again, experienced an, on average, lower maximum drawdown compared to the binary models.

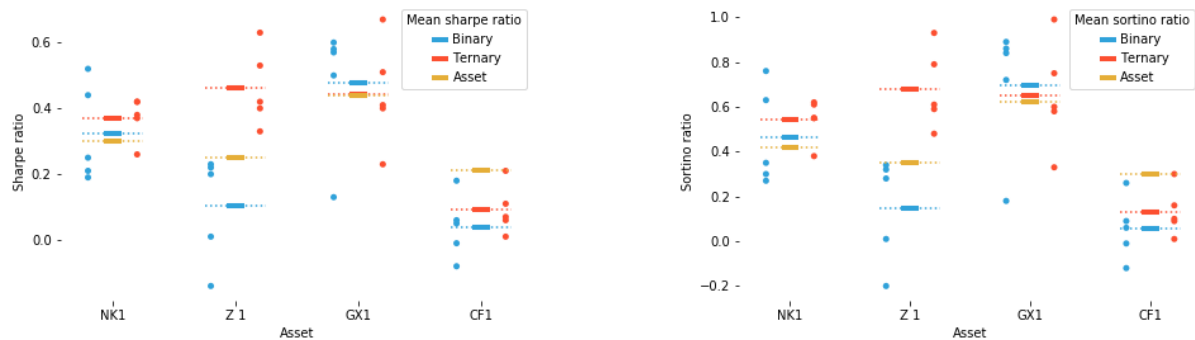


Figure 5.21: Sharpe ratio and Sortino ratio generated from five different combinatorial backtests in the binary as well as the ternary case.

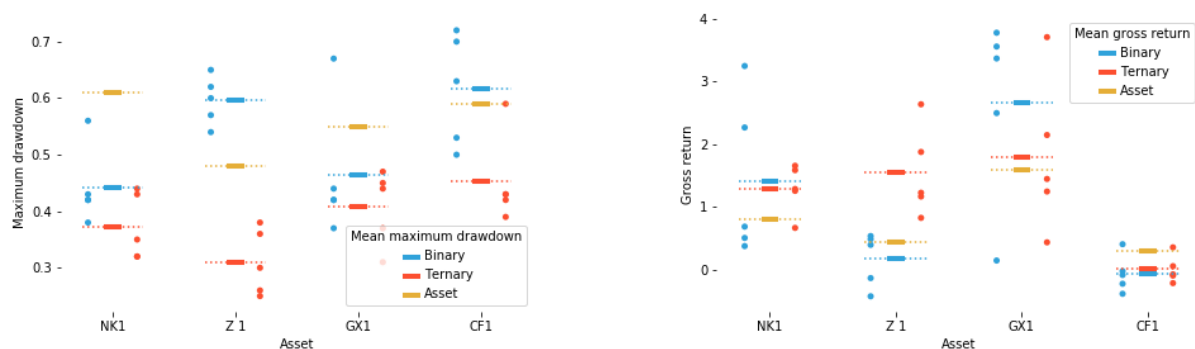


Figure 5.22: Maximum drawdown and Gross return generated from five different combinatorial backtests in the binary as well as the ternary case.

5.3.4 Consolidated training data

As depicted in table 5.8, when consolidating the data, the maximum drawdown was lower than the underlying asset across all contracts in the ternary case.

For all assets, there were no conclusive results other than the lower maximum drawdown. However, considering the performance over time for each asset, some behaviour stands out. First, all assets handled the financial crisis relatively well. Second, for each asset individually, the binary and ternary strategy seem to move in tandem over long periods of time, but have large divergences in small time frames. This was most likely an effect of making a false prediction at the wrong point in time. An example of this behaviour is depicted for *NK1* in figure 5.23.

Particularly when observing the middle of 2018 in figure 5.23. The ternary strategy stood outside the market and did not suffer the same amount of losses as the binary method did. This illustrates the importance of not being wrong at the wrong time. Furthermore, it highlights the value of being able to stay out of a indecisive market.

The opposite is of course true. There could be times when one is out of the market when the market spikes. An example was *NK1* during 2006. The ternary strategy remained flat while the asset and the binary strategy gained over 60%, see figure 5.23

Table 5.8: Confidence interval of 100 simulations for the performance metrics for all different assets. Shows each asset's own benchmarking value as well as the binary and ternary case. L = lower bound, M = mean, U = upper bound, B = binary, T = ternary, A = asset

		<i>Sharpe ratio</i>			<i>Sortino ratio</i>			Maximum drawdown			Gross return		
		<i>B</i>	<i>T</i>	<i>A</i>	<i>B</i>	<i>T</i>	<i>A</i>	<i>B</i>	<i>T</i>	<i>A</i>	<i>B</i>	<i>T</i>	<i>A</i>
Z1	<i>L</i>	0.04	0.19		0.05	0.27		0.37	0.19		-0.13	0.24	
	<i>M</i>	0.17	0.35	0.25	0.24	0.52	0.35	0.47	0.29	0.48	0.33	0.91	0.44
	<i>U</i>	0.29	0.51		0.42	0.77		0.57	0.39		0.78	1.58	
ES1	<i>L</i>	0.23	0.35		0.32	0.49		0.29	0.18		0.40	0.74	
	<i>M</i>	0.36	0.50	0.41	0.52	0.73	0.57	0.37	0.27	0.57	1.28	1.67	1.22
	<i>U</i>	0.50	0.65		0.72	0.97		0.46	0.36		2.16	2.59	
NQ1	<i>L</i>	0.38	0.27		0.55	0.39		0.25	0.25		1.22	0.49	
	<i>M</i>	0.53	0.43	0.62	0.78	0.64	0.87	0.31	0.31	0.54	2.97	1.42	3.35
	<i>U</i>	0.68	0.58		1.01	0.88		0.37	0.36		4.73	2.34	
GX1	<i>L</i>	0.53	0.40		0.77	0.57		0.30	0.23		2.57	1.08	
	<i>M</i>	0.66	0.54	0.44	0.97	0.79	0.62	0.36	0.30	0.55	5.00	2.27	1.60
	<i>U</i>	0.79	0.68		1.17	1.02		0.42	0.36		7.44	3.46	
NK1	<i>L</i>	0.23	0.19		0.33	0.27		0.46	0.38		0.42	0.15	
	<i>M</i>	0.36	0.34	0.30	0.53	0.50	0.42	0.55	0.50	0.61	1.58	1.25	0.81
	<i>U</i>	0.50	0.49		0.73	0.73		0.64	0.62		2.74	2.34	
CF1	<i>L</i>	0.36	0.01		0.51	0.01		0.46	0.38		1.20	-0.27	
	<i>M</i>	0.51	0.16	0.22	0.73	0.24	0.31	0.56	0.52	0.59	2.97	0.25	0.29
	<i>U</i>	0.65	0.32		0.95	0.46		0.67	0.66		4.75	0.77	
DM1	<i>L</i>	-0.07	-0.01		-0.09	-0.02		0.47	0.24		-0.36	-0.21	
	<i>M</i>	0.06	0.14	0.44	0.08	0.20	0.63	0.54	0.34	0.54	-0.07	0.16	1.35
	<i>U</i>	0.18	0.28		0.25	0.41		0.61	0.44		0.21	0.54	

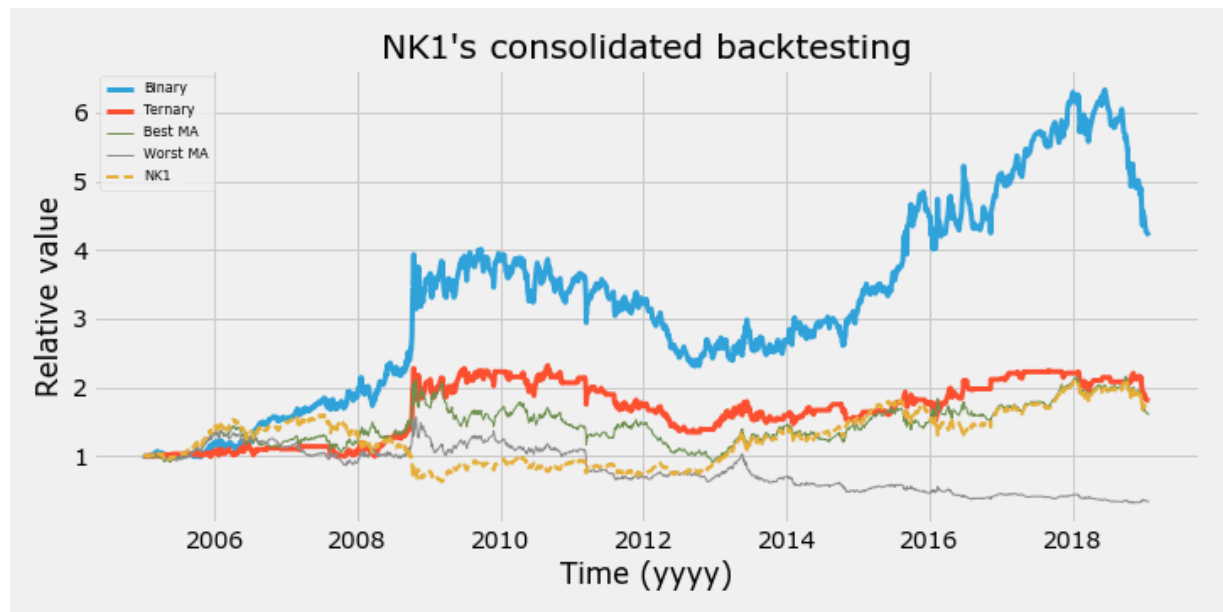


Figure 5.23: Consolidated backtesting of *NK1* for the binary and ternary cases. Also depicts the relative value of *NK1*'s contract value as well as the corresponding best and worst performing benchmarks

Chapter 6

Discussion

The discussion is structured as the workflow when developing a machine learning algorithm with added sections addressing the trading related subjects for this particular endeavour.

6.1 Feature selection

The feature selection process was somewhat complicated. While it is easy to find a multitude of trading indicators, homespun or otherwise, there was ambiguity as to which trading indicators were related to momentum strategies. In reviewing papers where machine learning is applied in a similar manner, the feature selection phase was never a focus and the general idea seemed to be to throw a lot of things at the model and see what sticks. Often, the plethora of features included trading indicators associated with momentum as well as ones associated with mean reversion. Mean reversion can be seen as the opposite to momentum where trades are initiated opposite to the trend under the assumption that the price will return to some kind of equilibrium. While one might assume that there are times for momentum and times for mean reversion, without an indicator telling which regime might be present, we thought such ambiguity would trip up even the best of pattern recognition algorithms. As such we strived to move away from inconsistent trading indicators and to have a relatively small group of features with rich variation within each type.

One natural way to approach the task of reducing an untidy universe of trading indicators would be to use feature importance in order to find out the usefulness of the features and as a extension the trading indicator. The hypothesis for the lacklustre result when attempting feature selection is that the market environment and the length of the trends have substantial impact on which features are useful to the model. Furthermore, in many applications of machine learning, an accuracy slightly above 50% would be a terrible result. For this problem anything above guessing rate is considered very much acceptable. The information contained in the features is thus limited and the ambiguity is significant as to whether there is anything to find in the feature space at times. This dynamic makes it complicated to disqualify features, who knows when they will be needed.

6.2 Model selection

6.2.1 Support Vector Machine

The model selection was an iterative process. As previously established, the SVM seemed promising but was to susceptible to the class imbalance while appearing to guess more than predict. The problem might lie in the failure to reduce the feature space to fewer and less correlated features. The fact that the SVM optimised to the same parameter values for different evaluation metrics is something worth discussing as well. A potential reason could be that the random search was initiated over the wrong set of parameter values. Perhaps the SVM could have been useful if more investigation was put into the matter. In hindsight, perhaps too much emphasis was put on the importance of a balanced confusion matrix.

6.2.2 Random forest

The parameter values of the Random forest used in the randomised search optimisation were chosen based partially on the our understanding of the algorithm. However, they were also subject computational limitations. The parameter $n_estimators$, which increases the bagging effect of the Random forest, (thus reducing variance in theory) had to be limited because large values had significant impact on computational time. Another interesting parameter was max_depth , which determines how many levels of nodes the tree can reach in total. A max depth of 8 in both optimised models meant that the model could consider a maximum of 8 different feature values before deciding what label to predict. While a max_depth of 8 theoretically allows the model to consider both long and short term features, there is no guarantee that an individual tree includes both types as the subset of features selected by the model is random. Speculation regarding the implicit feature selection by the Random forest illustrates the black-box nature of the problem and the lack of accountability as to what model behaviour the different features contributed with.

However, this built in feature selection based on Gini impurity, Equation ??, might be a crucial factor ensuring that the model worked at all.

6.2.3 Model evaluation metrics

The most suitable model was determined by randomised cross-validation combined with a grid search cross-validation and subsequently reviewing evaluation metrics. In the backtests, it became evident that these metrics alone were not comprehensive enough to find a model which performs well consistently over time. An important question was whether the metrics themselves were flawed or if the method should have been designed differently. The ultimate comparison was made with backtests, studying different performance metrics. As previously mentioned, high accuracy for predictions does not always lead to high returns. This lead us to believe that the evaluation should have been made differently.

Throughout the model selection phase, there was a reliance on confusion matrices in order to rank and evaluate models. While the accuracy of the model across the predictions is of upmost importance, the confusion matrix fails to illustrate the size of the price movements associated. For instance, if the model predicts downward moves with an accuracy of 53%, it makes a significant difference if the largest losses are included in those correct predictions. In fact, section 5.3.2 illustrates the decisive impact market timing have when large losses and gains occur. As such, it would be preferable to have an evaluation metrics which, just like the Matthews coefficient, is insensitive to class imbalance but also takes into account the losses and gains associated with the predictions.

For further studies, the Negative Log Loss metric could be examined further. Trying to establish the correlation between the probability of each predictions and the size of returns. If high returns are correlated with high probabilities, Negative Log Loss would serve as a great metric. Especially when considering section 5.3.2, where it was established that the models which performed well was able predict large losses and gains with a high degree of certainty.

6.2.4 Holding period and threshold decision

The holding period and threshold was chosen in a somewhat arbitrary manner because the choice was not meant to be a deciding factor for the model. Furthermore, the optimal values most likely fluctuates substantially over time. Optimising over the training period could be a futile exercise as the model might not generalise well out-of-sample. The main concern was to achieve a reasonable class balance. Furthermore, having a homogeneous holding period and threshold across the contracts enabled for combining the training data from all contracts in the consolidated backtest.

An alternative to arbitrarily choosing the parameters could be a dynamic labeling process where the holding period fluctuates and a label is assigned when some dynamic threshold is broken. Such a solution would carry over to the backtest where the rebalancing would be continuous as opposed to being limited to rebalancing days.

Another option would be to, instead of a two-sided threshold, further simplify the model by designing it to

only predict large downward movements. Such an approach could have made use of evaluation metrics such as F1 and Recall.

As the backtests indicates, the models struggle to generalise and it would be ignorant to rule out the holdingperiods and thresholds as a potential contributor to this problem.

6.3 Backtesting and benchmarking

As the backtesting and benchmarking section covers several types of backtests for different assets as well as a binary and ternary case this discussion aims to create a summarising view of the results as a whole, starting off with a list of what seemed to have been more and less successful.

6.3.1 More successful

6.3.1.1 Crisis identification

The financial crisis of 2008 was handled well by the algorithm for all assets across all backtests except for the traditional walk-forward. This suggests that the model has in fact found characteristics for such market movements. The simple trading strategies depict similar behaviour, illustrated in Figures F.1:F.6. The behaviour of the machine learning algorithm is not unexpected as the Simple trading strategies, among other trading indicators, are included in the feature space. However, the fact that the models outperformed the Simple trading strategies indicates that the model has made use of its information advantage.

6.3.1.2 Lower drawdowns

The only performance metric which outperformed all assets in both the binary and ternary case for the consolidated backtest was the maximum drawdown. Even when our strategies performed well, significantly better than the underlying asset, the maximum drawdown was low. Between the two, the ternary strategy was most effective in limiting the maximum drawdown.

6.3.1.3 Outperforming simple MA benchmarks

Across all assets there was only one Simple trading strategy which outperformed our strategies and the underlying asset, *DM1*. Interestingly, this is the asset where our strategy performed the worst. In general it is fair to claim that this machine learning model has indeed gained insights beyond those of the Simple trading strategies.

6.3.2 Less successful

6.3.2.1 Computational restraints

Due to computational restraints, only the consolidated backtests were conducted as a series of tests combined in a confidence interval. All other backtesting methods rely on refitting and predicting frequently, which quickly becomes computationally challenging. The consolidated backtest is only fitted to the data once for the entire data set and predicts once for every asset which is why the test could be repeated 100 times.

When writing this thesis, computational power has been a frequent hindrance and the reason to why some ideas and methods were excluded from the paper or never implemented.

6.3.2.2 Traditional walk-forward

If neglecting the extreme performance of the binary model for *GXI*, the traditional backtests performed poorly when compared to the other backtests. As discussed, this was expected and validates the importance of training the models on relevant data.

6.3.2.3 Generalise results

From the confidence interval of the consolidated backtest, it is clear that the best performing model varies between binary, ternary and the two benchmarks. As such, no distinct generalisation can be made for this type of machine learning strategy. Two questions inevitably arise. What is the main reason for the models failing to generalise and what can be done to mitigate this. Given the overall positive results for *NK1*, it may be as simple as; there needs to be a model selection phase for each individual asset. Due to computational restraints and a lack of data for some assets, the approach was not pursued in the thesis.

6.3.2.4 Method approach

In retrospective, it would have been sound to split the data into three parts, training, validation and testing. This would have enabled a more iterative selection process, where actual performance metrics and backtests could have been generated on a validation set before evaluating the results on a test set. The reason for avoiding this methodology was the fear of not training the models on a sufficient amount of data.

6.3.3 Transaction costs

For the more frequently traded strategies an analysis on the effect of trading costs would be necessary. However, the strategies did in general not seek to rebalance very often which meant a limited impact of transaction costs on the results.

Chapter 7

Conclusion

As Marcos Lopez de Prado a professor within the field of computational finance says "*Many investment managers believe that the secret to riches is to implement an extremely complex ML algorithm. They are setting themselves up for a disappointment. If it was as easy as coding a state-of-the art classifier, most people in Silicon Valley would be billionaires. A successful investment strategy is the result of multiple factors.*" [26]

Ultimately we can conclude that the model is too simplistic when aiming to only determine the direction of market movement because taking the wrong position at a few points in time can destroy the long-term results of a trading strategy. We believe that the approach of optimising the model using evaluation metrics is sound but that it failed in practise due to poor execution. In hindsight it is clear that there would need to be a measure, or a group of measures, which bridges the gap between overall accuracy and actual returns. Measures which takes into account class imbalances, probabilities as well as returns. Perhaps a combination of two machine learning algorithms, one optimised with Matthews Correlation coefficient for predicting direction of movements in combination with an algorithm optimised with Negative Log Loss for predicting the size of bets. Furthermore, with the results at hand, it might would have been wise to focus on reinforcing the strategy's ability to limit drawdowns by optimising its ability to identify negative trends.

Even though the overall results seems disappointing there is still a lot to be learned from the results. The model does in fact manage to find some information in the data, having outperformed the simple moving average benchmarks in almost every scenario. Furthermore, the strategies experienced low maximum drawdowns. As previously established, capital preservation by minimising drawdowns is an important aspect of managing money. While portfolio allocation is outside the scope of this thesis, financial products which perform well in times of crisis has been in high demand since 2008. Furthermore, products which are negatively correlated, or uncorrelated, with common asset classes can improve the performance metrics of a portfolio by reducing volatility. The strategy developed here exhibit such tendencies, especially when it matters, in times of crisis.

Bibliography

- [1] Gerald Appel. *Technical analysis: power tools for active investors*. FT Press, 2005.
- [2] Nick Baltas and Robert Kosowski. Momentum strategies in futures markets and trend-following funds. In *Paris December 2012 Finance Meeting EUROFIDAI-AFFI Paper*, 2013.
- [3] Marianne Baxter and Robert G King. Measuring business cycles: approximate band-pass filters for economic time series. *Review of economics and statistics*, 81(4):575–593, 1999.
- [4] Paul Beekhuizen and Winfried G Hallerbach. Uncovering trend rules. *Available at SSRN 2604942*, 2015.
- [5] Graham Benjamin, David L Dodd, and S Cottle. Security analysis. *Mc Graw Hill Inc, New York*, 1934.
- [6] Ron Bird, Xiaojun Gao, and Danny Yeung. Time-series and cross-sectional momentum strategies under alternative implementation strategies. *Australian Journal of Management*, 42(2):230–251, 2017.
- [7] Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PloS one*, 12(6):e0177678, 2017.
- [8] William Brock, Josef Lakonishok, and Blake LeBaron. Simple technical trading rules and the stochastic properties of stock returns. *The Journal of finance*, 47(5):1731–1764, 1992.
- [9] B Bruder, TL Dao, JC Richard, and T Roncalli. Trend filtering methods for momentum strategies, *ssrn*. 2011.
- [10] KC Chan. On the contrarian investment strategy. *Journal of business*, pages 147–163, 1988.
- [11] Alfred Cowles 3rd and Herbert E Jones. Some a posteriori probabilities in stock market action. *Econometrica, Journal of the Econometric Society*, pages 280–294, 1937.
- [12] Werner FM De Bondt and Richard Thaler. Does the stock market overreact? *The Journal of finance*, 40(3):793–805, 1985.
- [13] Eugene F Fama. The behavior of stock-market prices. *The journal of Business*, 38(1):34–105, 1965.
- [14] Benjamin Graham. *The Intelligent Investor: The Classic Text on Value Investing*. Collins, 1949.
- [15] James Grant. *The Great Metropolis*, volume 1. T. Foster, 1838.
- [16] Gerwin AW Griffioen. Technical analysis in financial markets. 2003.
- [17] Richard DF Harris and Fatih Yilmaz. A momentum trading strategy based on the low frequency component of the exchange rate. *Journal of Banking & Finance*, 33(9):1575–1585, 2009.
- [18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, volume 2. Springer, 2017.
- [19] Robert J Hodrick and Edward C Prescott. Postwar us business cycles: an empirical investigation. *Journal of Money, credit, and Banking*, pages 1–16, 1997.

- [20] John D. Hunter. matplotlib: A 2d graphics environment, 90-95 (2007). [Online; accessed 2019-05-09].
- [21] Brian Hurst, Yao Hua Ooi, and Lasse Heje Pedersen. A century of evidence on trend-following investing. 2017.
- [22] Narasimhan Jegadeesh and Sheridan Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of finance*, 48(1):65–91, 1993.
- [23] Göran Kauermann, Tatyana Krivobokova, and Willi Semmler. Filtering time series with penalized splines. *Studies in Nonlinear Dynamics & Econometrics*, 15(2), 2011.
- [24] Edwin Lefevre. Reminiscences of a stock operator. new york: George h. *Doran and Company*, 1923.
- [25] Ari Levine and Lasse Heje Pedersen. Which trend is your friend? *Financial Analysts Journal*, 72(3):51–66, 2016.
- [26] Marcos López de Prado. *Advances in Financial Machine Learning*, volume 1. John Wiley Sons, Inc., Hoboken, New Jersey, 2018.
- [27] Burton G Malkiel and Eugene F Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.
- [28] Benoit Mandelbrot. New methods in statistical economics. *Journal of political economy*, 71(5):421–440, 1963.
- [29] Ben R Marshall, Nhut H Nguyen, and Nuttawat Visaltanachoti. Time-series momentum versus moving average trading rules. *Available at SSRN 2225551*, 2014.
- [30] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.
- [31] Wes McKinney. pandas data structures for statistical computing in python, 51-56 (2010). [Online; accessed 2019-05-09].
- [32] Tobias J Moskowitz, Yao Hua Ooi, and Lasse Heje Pedersen. Time series momentum. *Journal of financial economics*, 104(2):228–250, 2012.
- [33] Fotis Papailias and Dimitrios D Thomakos. An improved moving average technical trading rule. *Physica A: Statistical Mechanics and its Applications*, 428:458–469, 2015.
- [34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [35] William F Sharpe. Mutual fund performance. *The Journal of business*, 39(1):119–138, 1966.
- [36] Frank A Sortino and Robert Van Der Meer. Downside risk. *Journal of Portfolio Management*, 17(4):27, 1991.
- [37] S. Chris Colbert Stéfan van der Walt and Gaël Varoquaux. NumPy the numpy array: A structure for efficient numerical computation, 22-30 (2011). [Online; accessed 2019-05-09].
- [38] Ryan Sullivan, Allan Timmermann, and Halbert White. Data-snooping, technical trading rule performance, and the bootstrap. *The journal of Finance*, 54(5):1647–1691, 1999.
- [39] Manoj Thakur and Deepak Kumar. A hybrid financial trading support system using multi-category classifiers and random forest. *Applied Soft Computing*, 67:337–349, 2018.
- [40] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

- [41] Oliphant Travis E. A guide to NumPy. USA: Trelgol Publishing, (2006).
- [42] Michael Waskom, Olga Botvinnik, Drew O’Kane, Paul Hobson, Joel Ostblom, Saulius Lukauskas, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, John B. Cole, Jordi Warmenhoven, Julian de Ruiter, Cameron Pye, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Pete Bachant, Marcel Martin, Kyle Meyer, Alistair Miles, Yoav Ram, Thomas Brunner, Tal Yarkoni, Mike Lee Williams, Constantine Evans, Clark Fitzgerald, Brian, and Adel Qalieh. mwaskom/seaborn: v0.9.0 (july 2018), July 2018.
- [43] Miao Yu and Jinguo Song. Volatility forecasting: Global economic policy uncertainty and regime switching. *Physica A: Statistical Mechanics and its Applications*, 511:316–323, 2018.

Appendices

Appendix A

Hyper-parameter optimised values

Table A.1: Hyper-parameter optimised values for a Random forest using a holding period of ten days in the binary case.

Parameter/Model	<i>Accuracy</i>	<i>F1</i>	<i>Negative Log Loss</i>	<i>Matthews Correlation Coefficient</i>	<i>Recall</i>
<i>n_estimators</i>	110	110	200	160	160
<i>min_samples_split</i>	6	9	5	4	5
<i>min_samples_leaf</i>	1	1	2	4	3
<i>max_features</i>	auto	sqrt	auto	auto	sqrt
<i>max_depth</i>	8	8	8	8	10
<i>bootstrap</i>	False	False	True	False	True

Table A.2: Hyper-parameter optimised values for the Random forest using a ten day holding period and a two percent threshold

Parameter/Model	<i>Accuracy</i>	<i>F1</i>	<i>Negative Log Loss</i>	<i>Matthews Correlation Coefficient</i>	<i>Recall</i>
<i>n_estimators</i>	130	180	150	160	110
<i>min_samples_split</i>	5	4	4	4	10
<i>min_samples_leaf</i>	5	5	4	4	3
<i>max_features</i>	sqrt	sqrt	sqrt	sqrt	auto
<i>max_depth</i>	75	85	8	8	10
<i>bootstrap</i>	True	True	True	True	True

Appendix B

Graphs from simple trading strategy backtesting

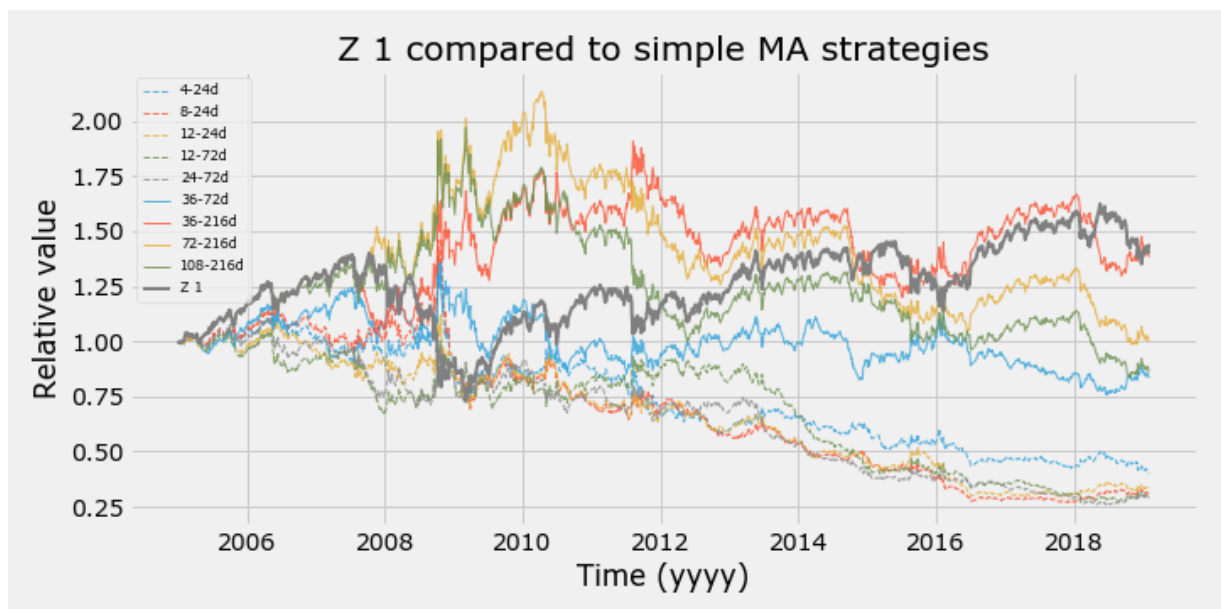


Figure B.1: Depicts the different backtest generated by applying a simple moving average difference trading strategy. Asset tested on was $Z 1$

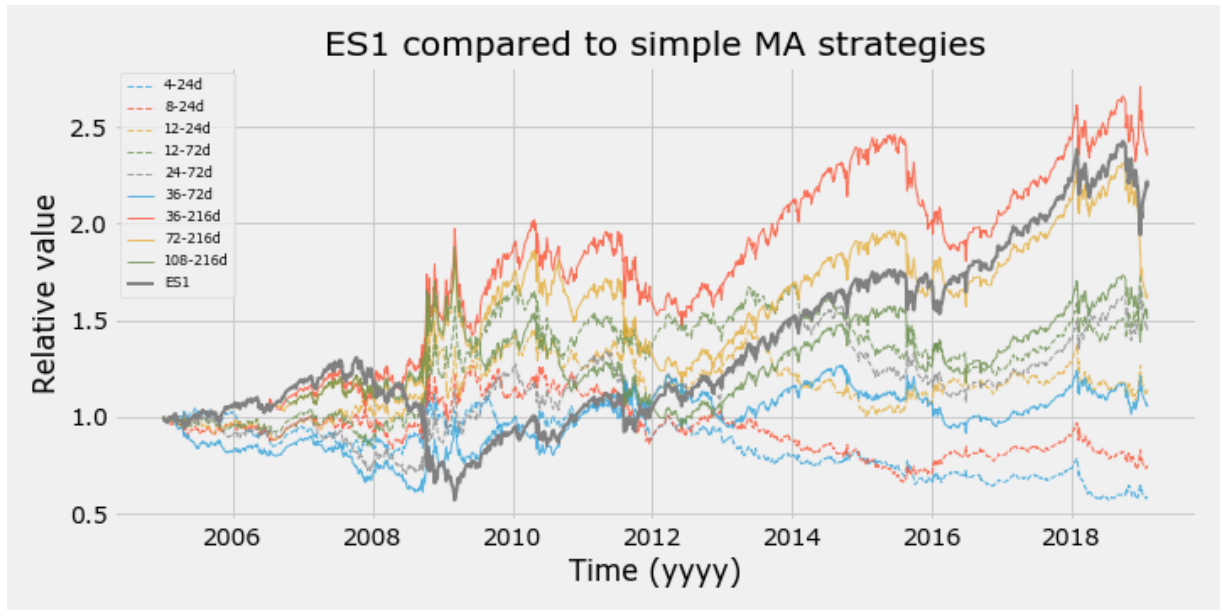


Figure B.2: Depicts the different backtest generated by applying a simple moving average difference trading strategy. Asset tested on was *ES1*

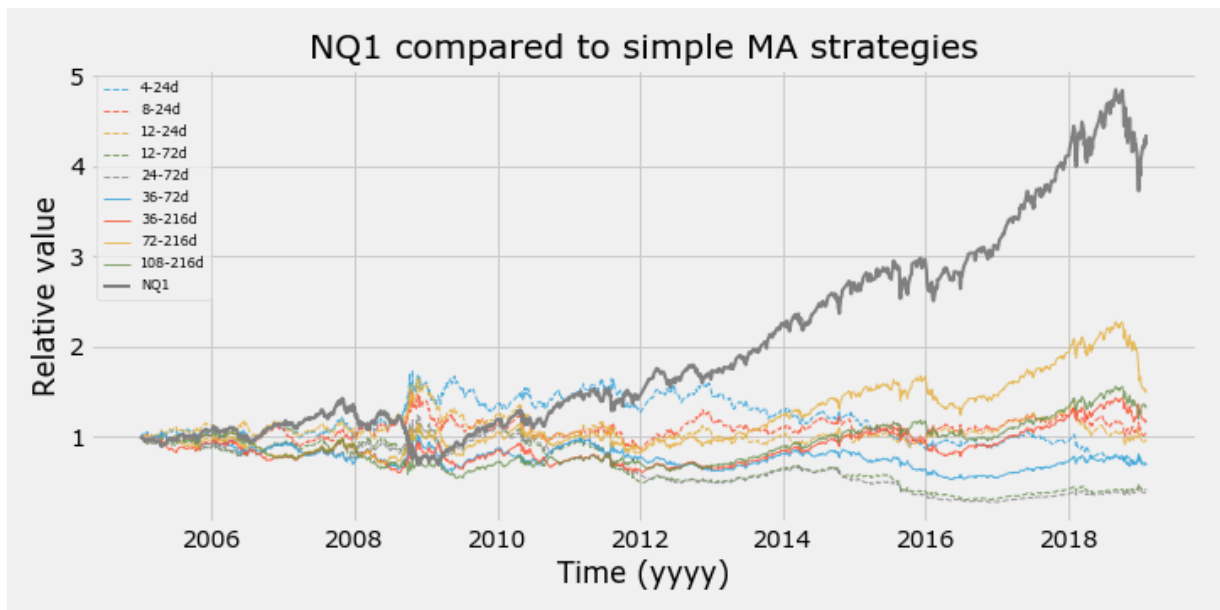


Figure B.3: Depicts the different backtest generated by applying a simple moving average difference trading strategy. Asset tested on was *ES1*

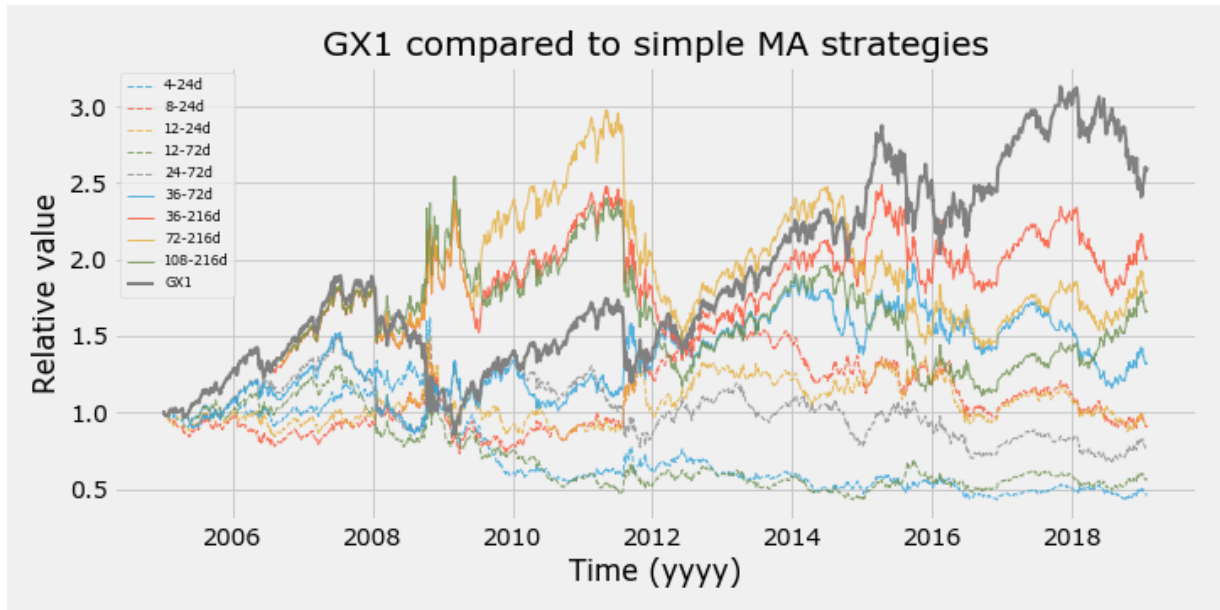


Figure B.4: Depicts the different backtest generated by applying a simple moving average difference trading strategy. Asset tested on was *GX1*

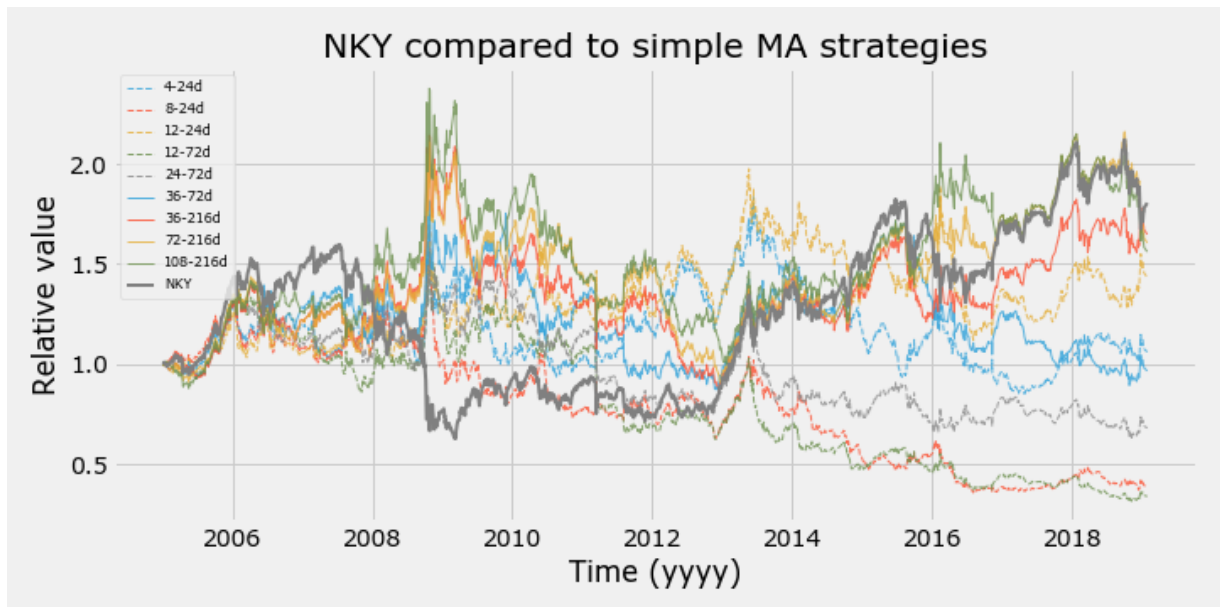


Figure B.5: Depicts the different backtest generated by applying a simple moving average difference trading strategy. Asset tested on was *NK1*

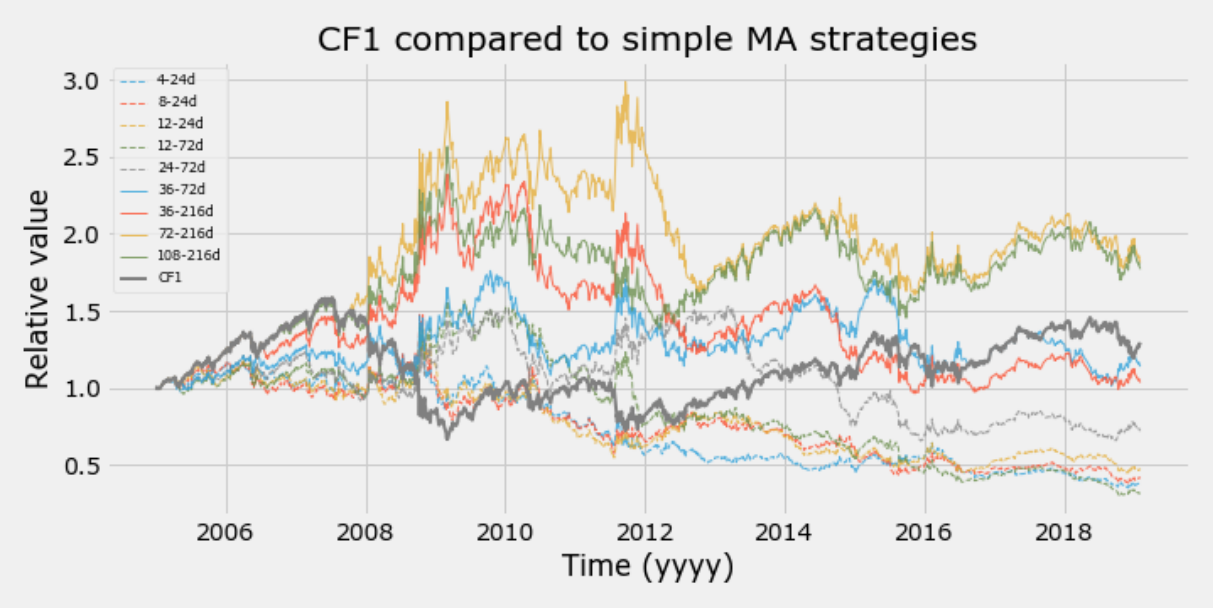


Figure B.6: Depicts the different backtest generated by applying a simple moving average difference trading strategy. Asset tested on was *CF1*

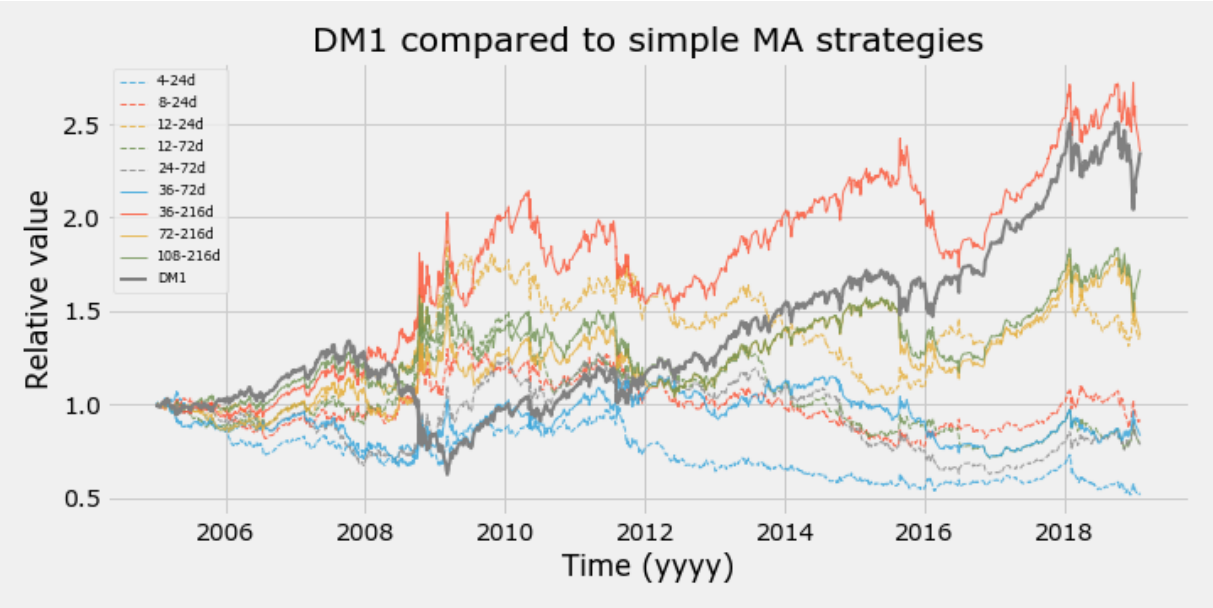


Figure B.7: Depicts the different backtest generated by applying a simple moving average difference trading strategy. Asset tested on was *DM1*

Appendix C

Graphs from traditional walk-forward backtesting

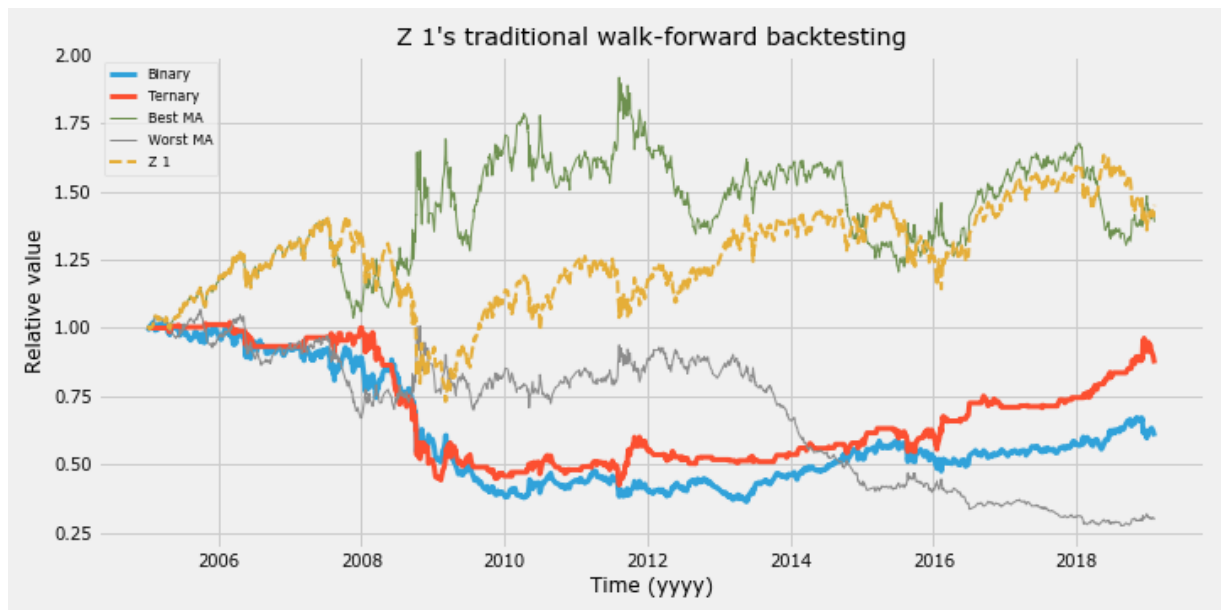


Figure C.1: Traditional walk-forward backtesting of $Z1$ for the binary and ternary cases. The relative value of $Z1$'s contract value as well as the corresponding best and worst performing benchmarks are also depicted.

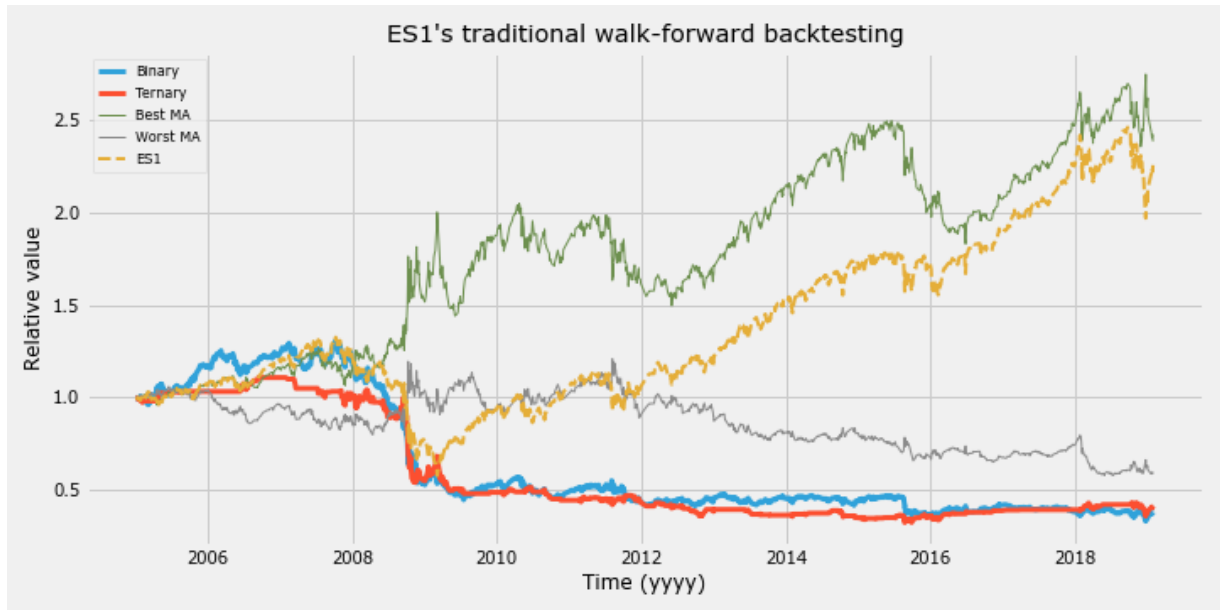


Figure C.2: Traditional walk-forward backtesting of *ES1* for the binary and ternary cases. The relative value of *ES1*'s contract value as well as the corresponding best and worst performing benchmarks are also depicted.

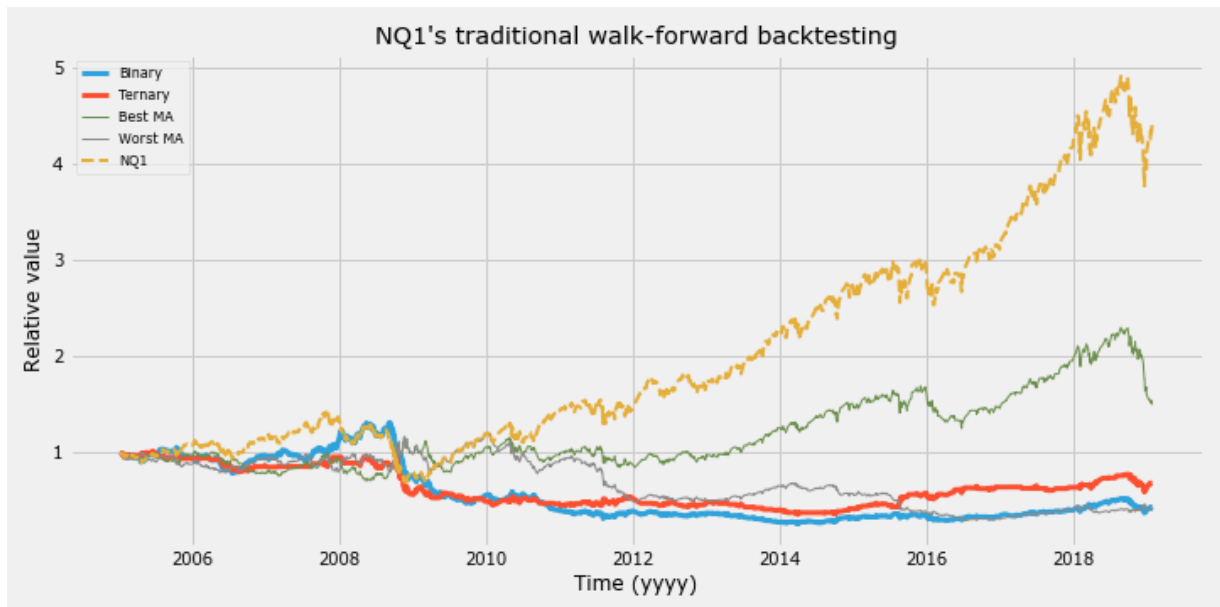


Figure C.3: Traditional walk-forward backtesting of *NQ1* for the binary and ternary cases. The relative value of *NQ1*'s contract value as well as the corresponding best and worst performing benchmarks are also depicted.

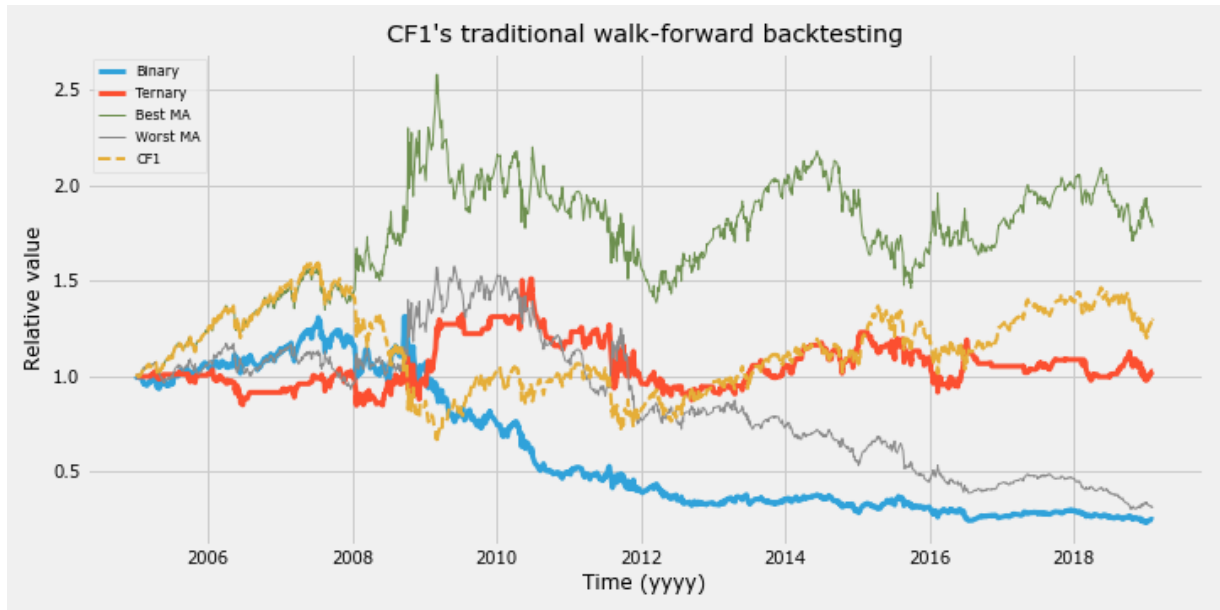


Figure C.4: Traditional walk-forward backtesting of *CF1* for the binary and ternary cases. The relative value of *CF1*'s contract value as well as the corresponding best and worst performing benchmarks are also depicted.

Appendix D

Graphs from warm-up walk-forward backtesting

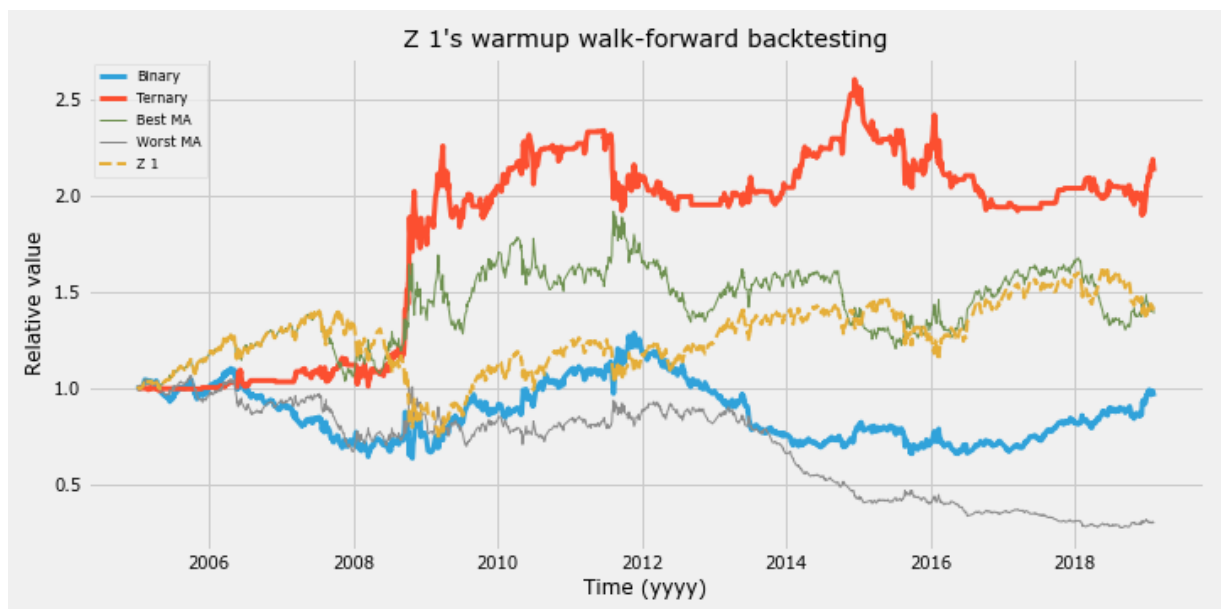


Figure D.1: Warmup walk-forward backtesting of $Z 1$ for the binary and ternary cases. The relative value of $Z 1$'s contract value and the corresponding best and worst performing Simple trading strategies are also depicted.

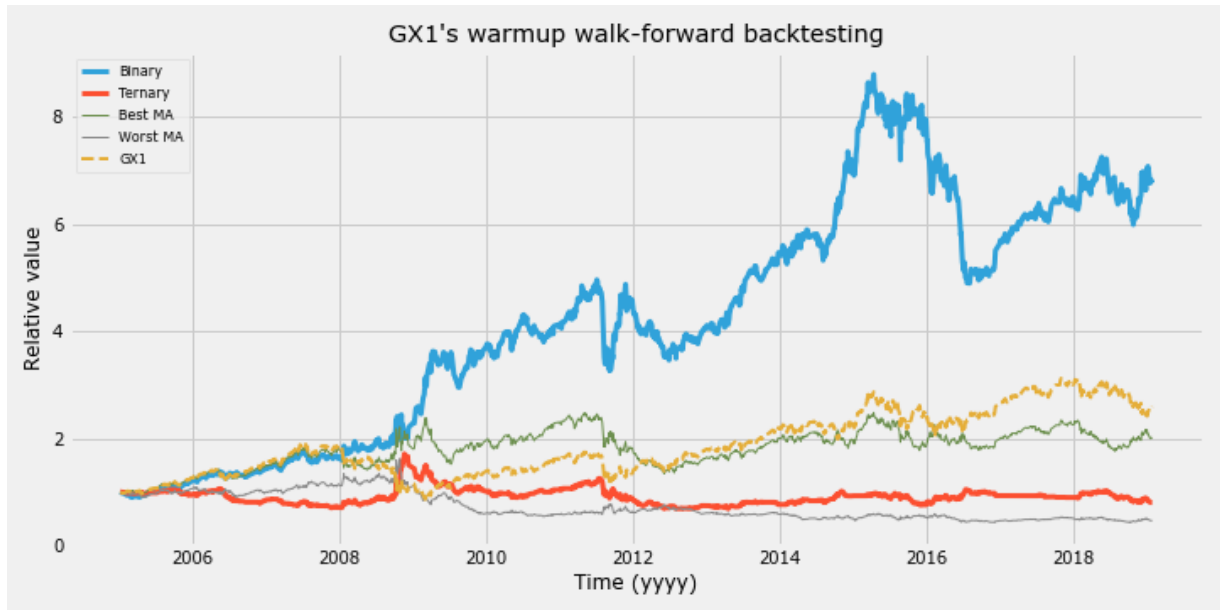


Figure D.2: Warmup walk-forward backtesting of *GX1* for the binary and ternary cases. The relative value of *GX1*'s contract value and the corresponding best and worst performing Simple trading strategies are also depicted.

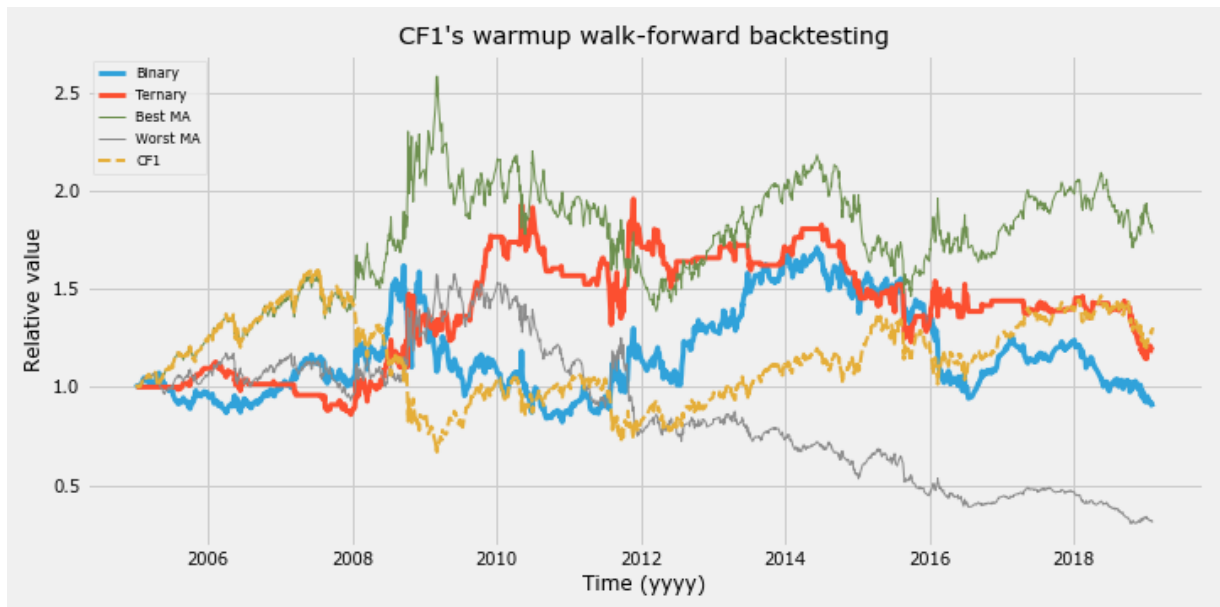


Figure D.3: Warmup walk-forward backtesting of *CF1* for the binary and ternary cases. The relative value of *CF1*'s contract value and the corresponding best and worst performing Simple trading strategies are also depicted.

Appendix E

Graphs from combinatorial backtests

E.1 NK1

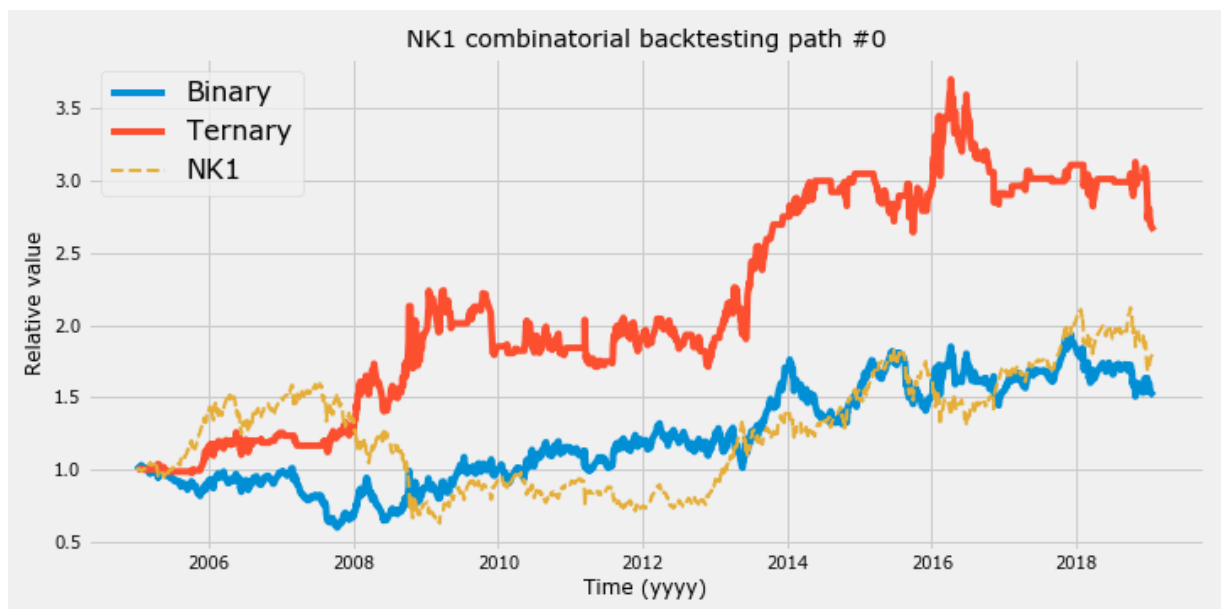


Figure E.1: Depicts the first paths generated from the combinatorial backtest for the asset *NK1*.

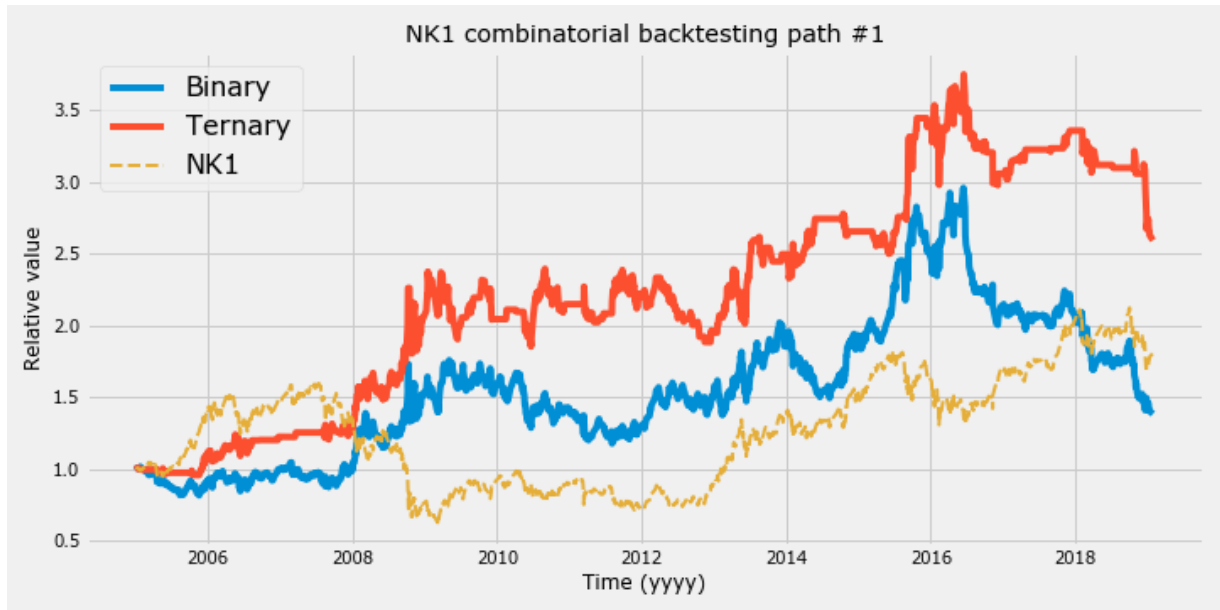


Figure E.2: Depicts the second paths generated from the combinatorial backtest for the asset *NK1*.

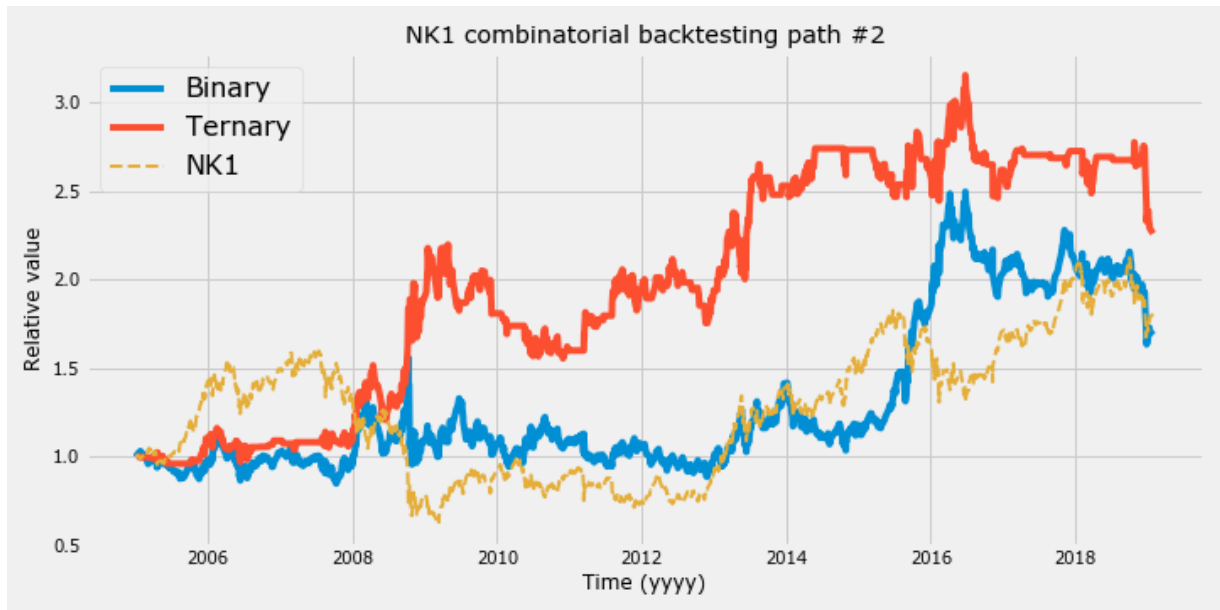


Figure E.3: Depicts the third paths generated from the combinatorial backtest for the asset *NK1*.

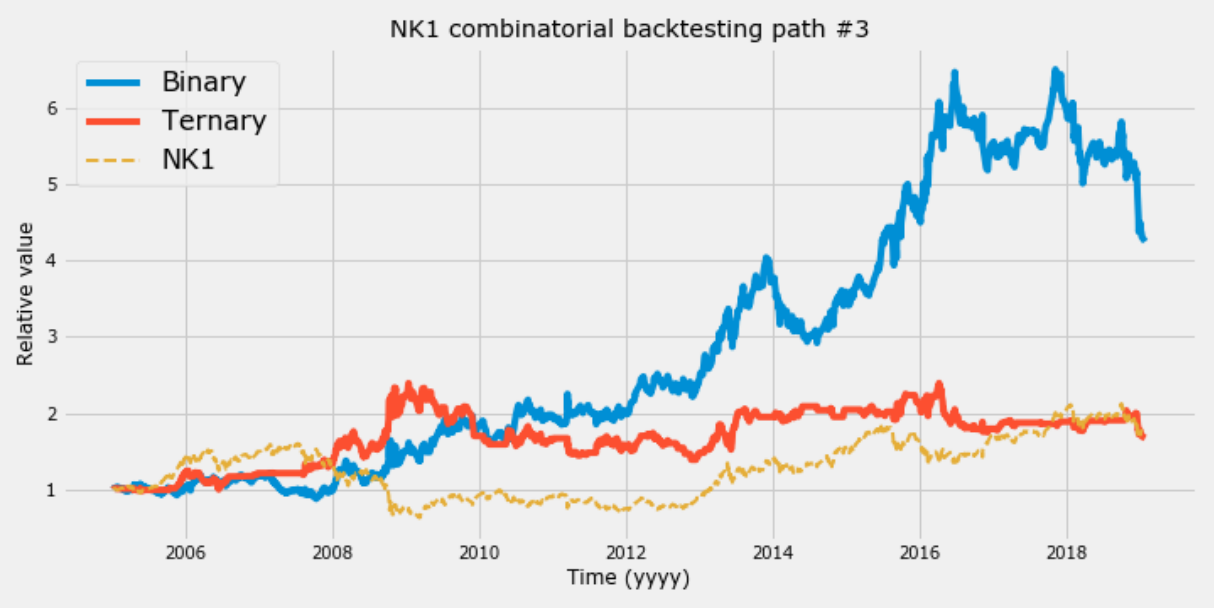


Figure E.4: Depicts the fourth paths generated from the combinatorial backtest for the asset *NK1*.

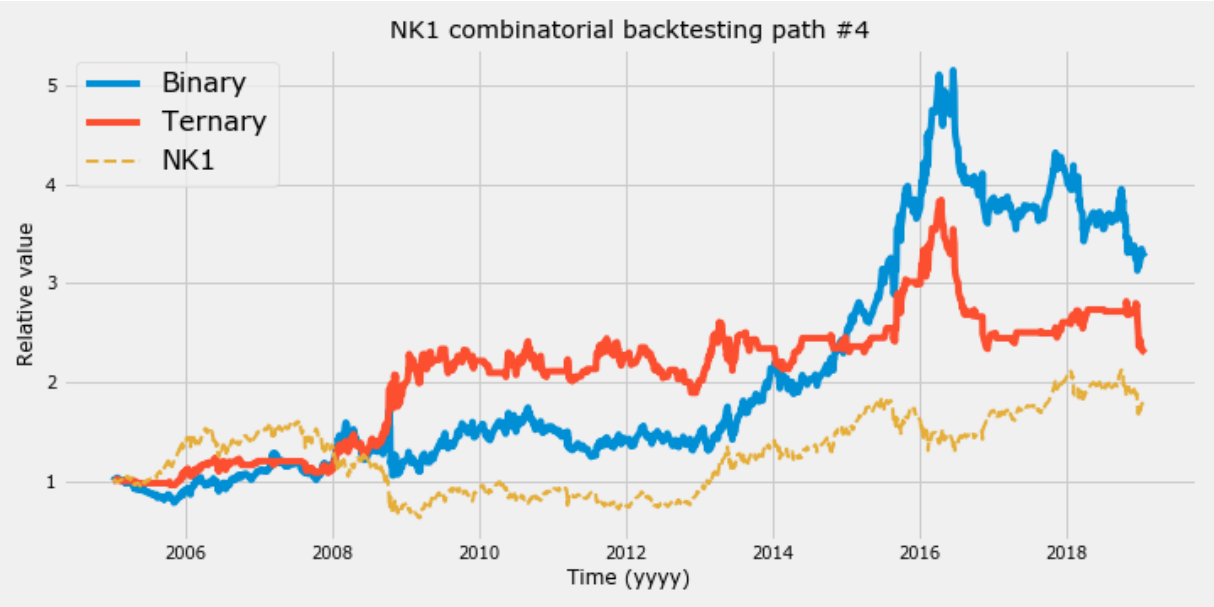


Figure E.5: Depicts the fifth paths generated from the combinatorial backtest for the asset *NK1*.

E.2 Z 1

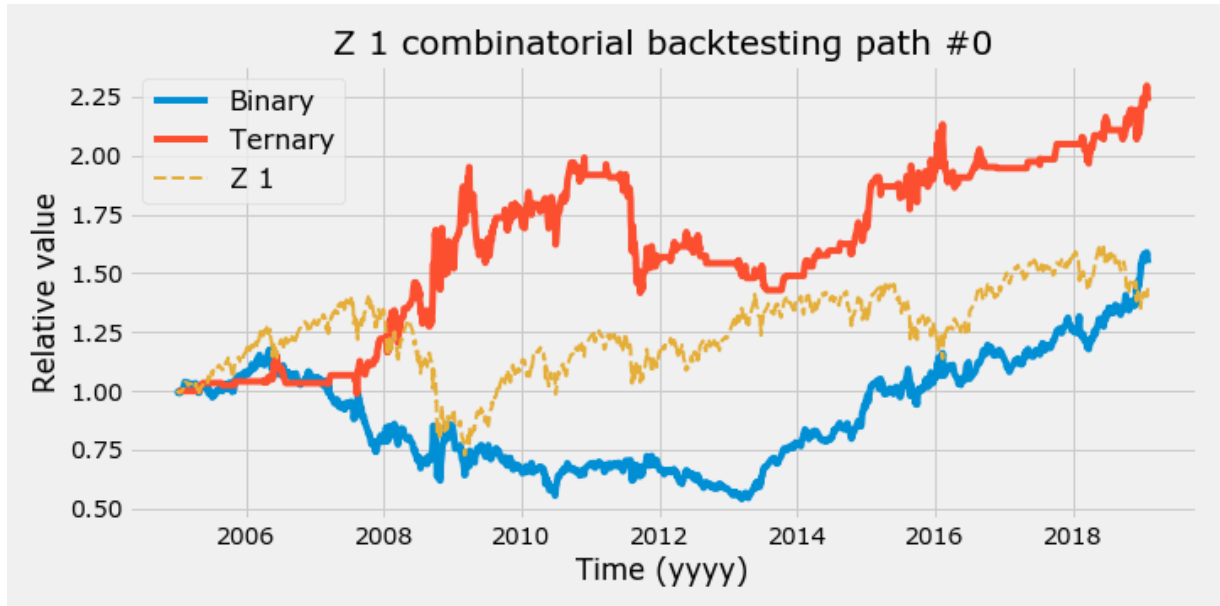


Figure E.6: Depicts the first paths generated from the combinatorial backtest for the asset *Z 1*.

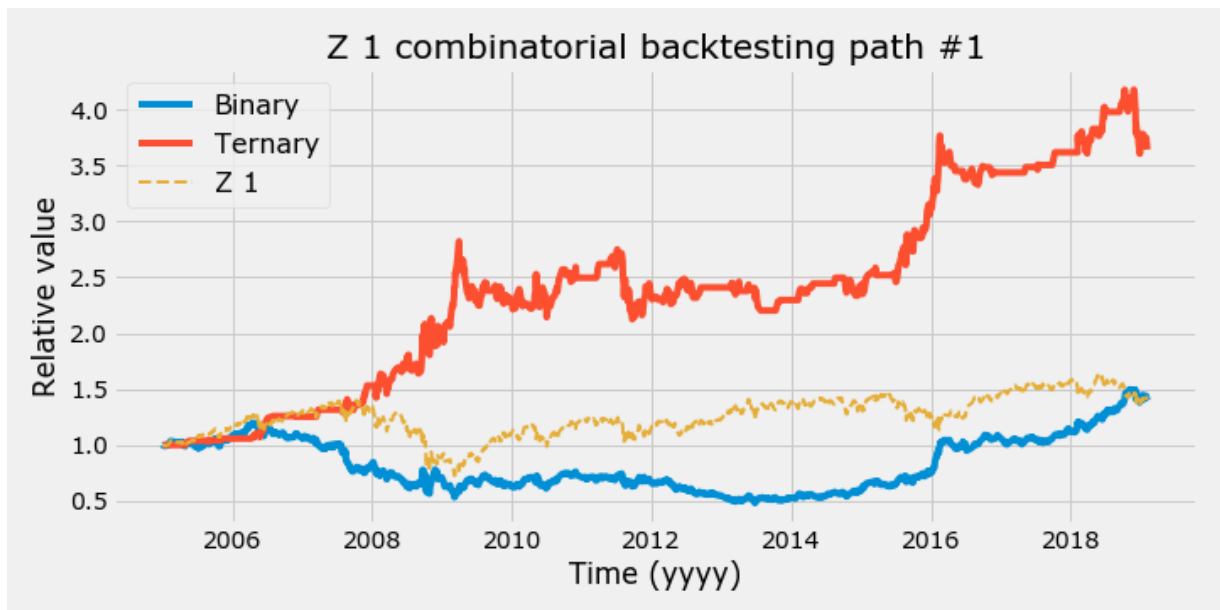


Figure E.7: Depicts the second paths generated from the combinatorial backtest for the asset *Z 1*.

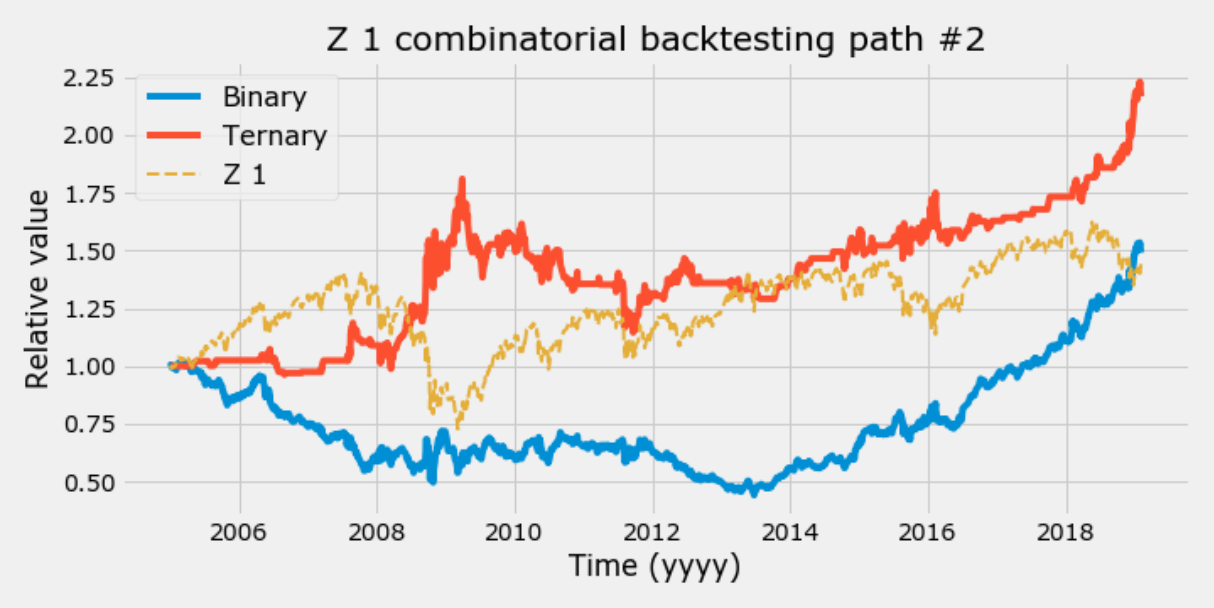


Figure E.8: Depicts the third paths generated from the combinatorial backtest for the asset $Z 1$.

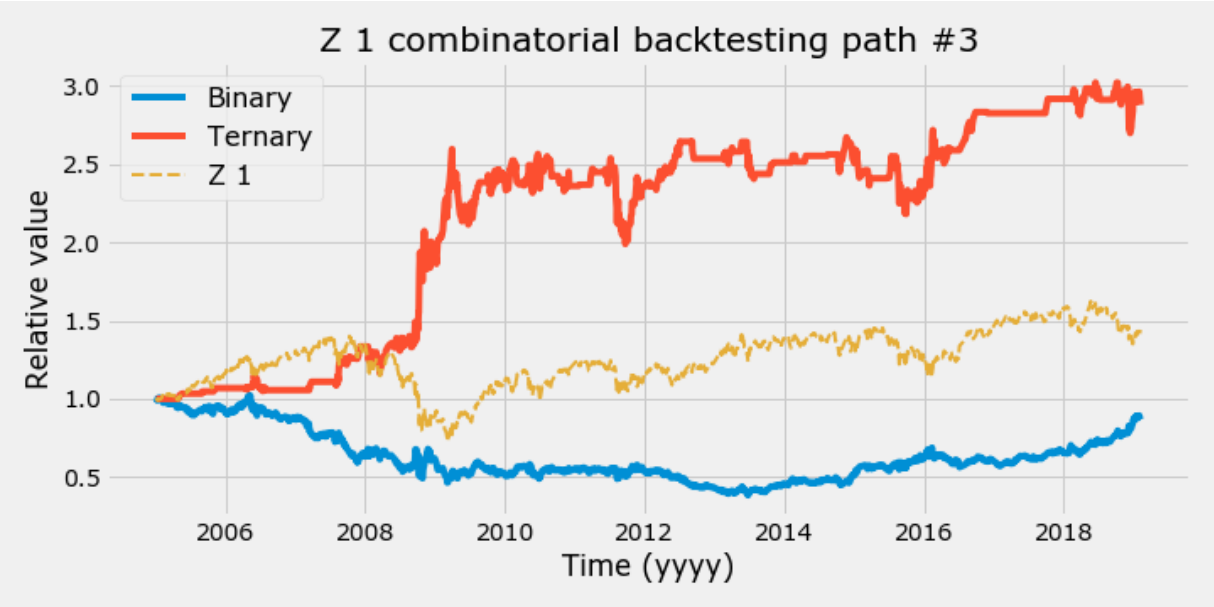


Figure E.9: Depicts the fourth paths generated from the combinatorial backtest for the asset $Z 1$.

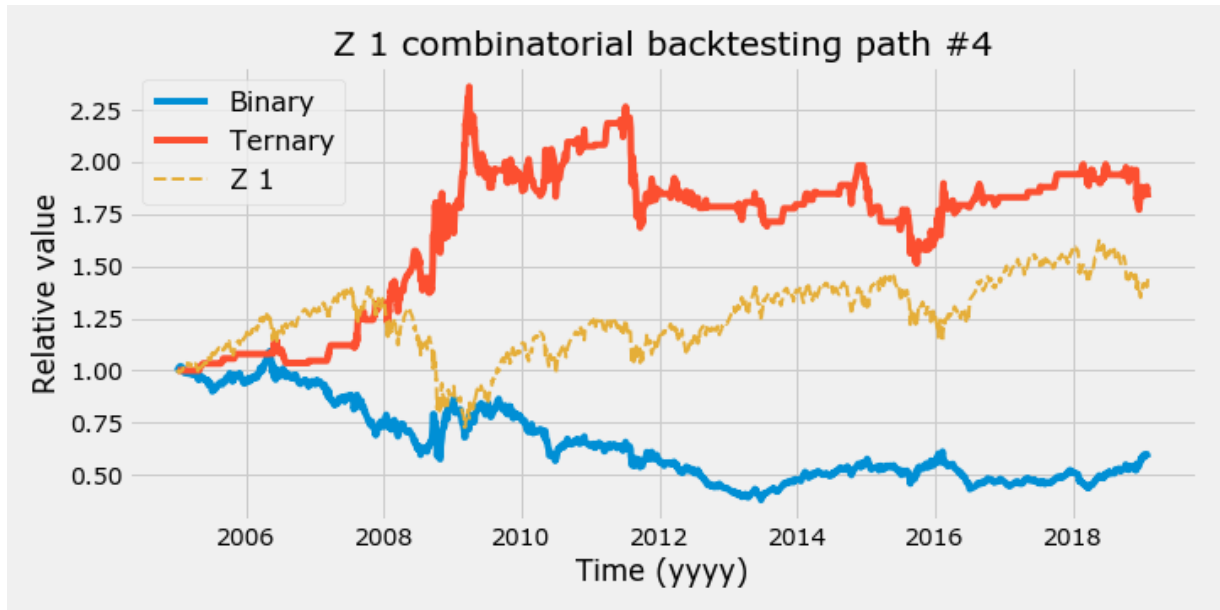


Figure E.10: Depicts the fifth paths generated from the combinatorial backtest for the asset *Z 1*.

E.3 GX1

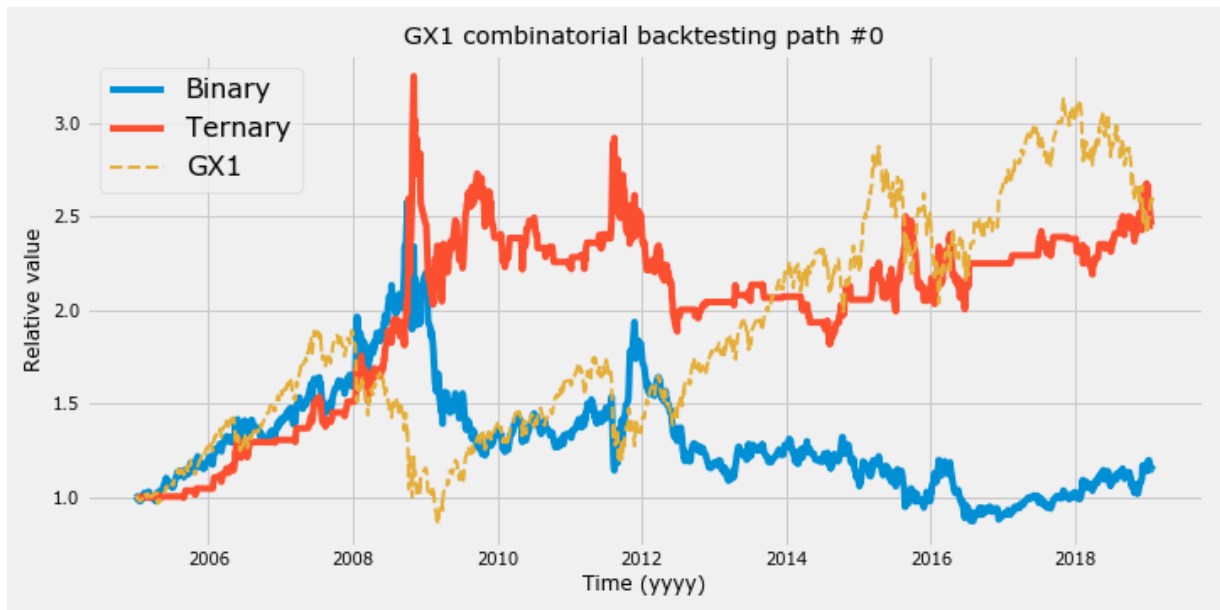


Figure E.11: Depicts the first paths generated from the combinatorial backtest for the asset *GX1*.

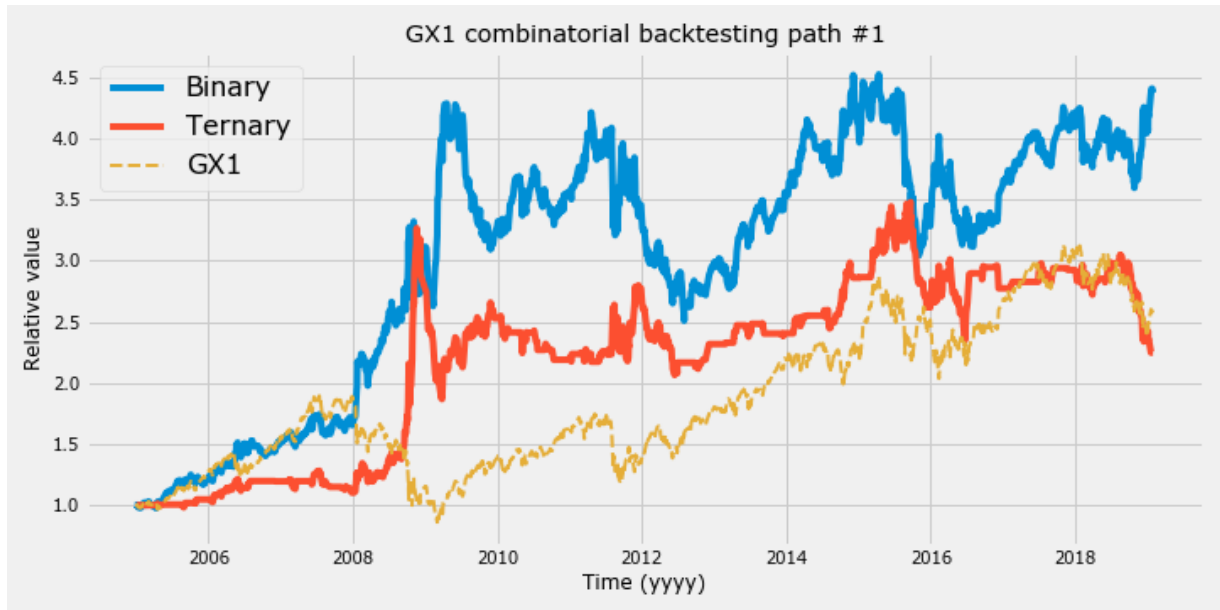


Figure E.12: Depicts the second paths generated from the combinatorial backtest for the asset *GX1*.

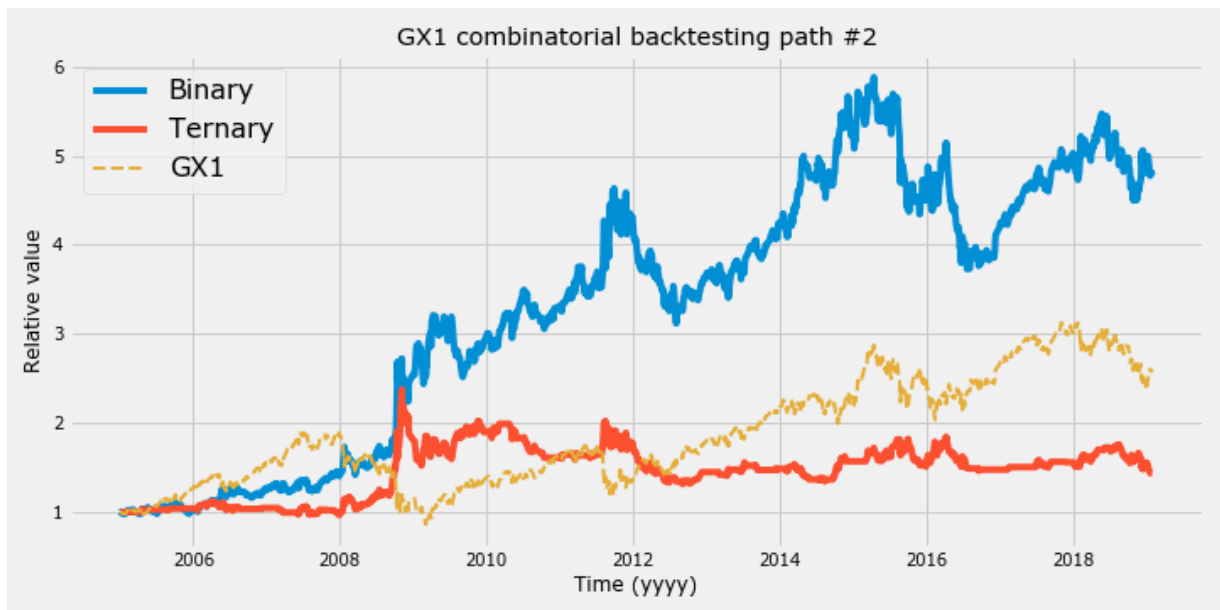


Figure E.13: Depicts the third paths generated from the combinatorial backtest for the asset *GX1*.

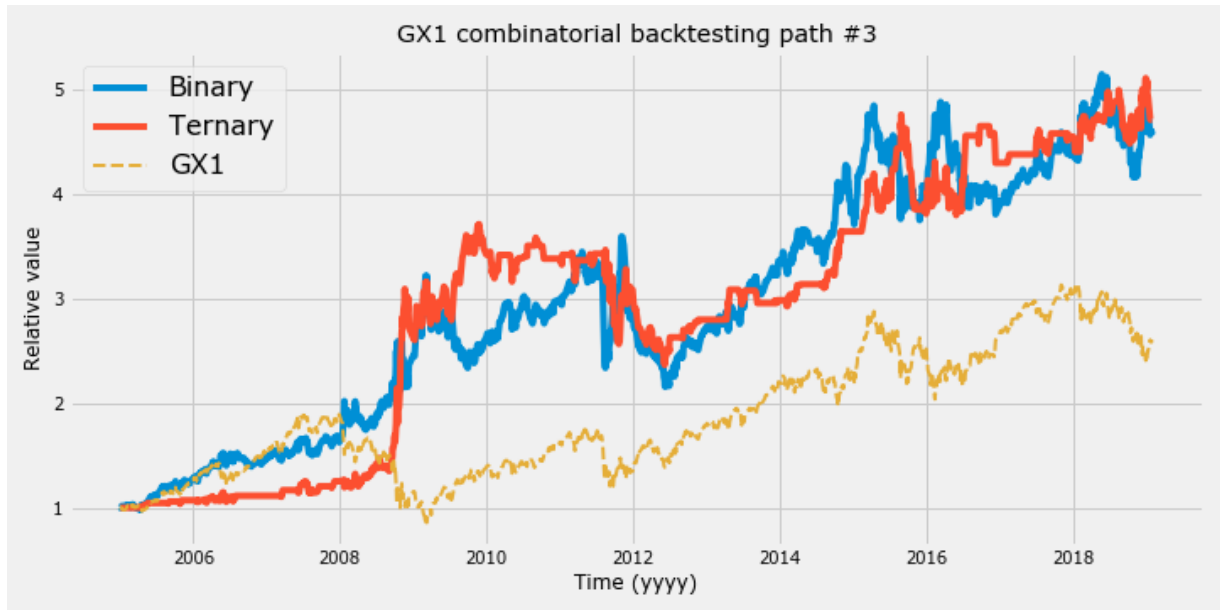


Figure E.14: Depicts the fourth paths generated from the combinatorial backtest for the asset *GX1*.

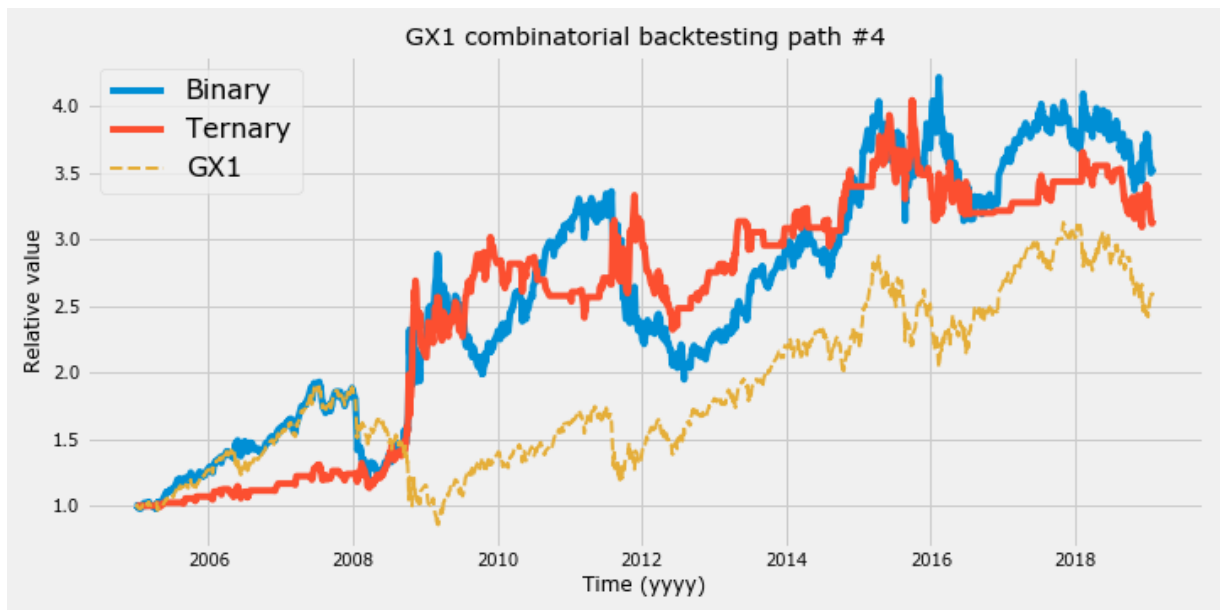


Figure E.15: Depicts the fifth paths generated from the combinatorial backtest for the asset *GX1*.

E.4 CF1

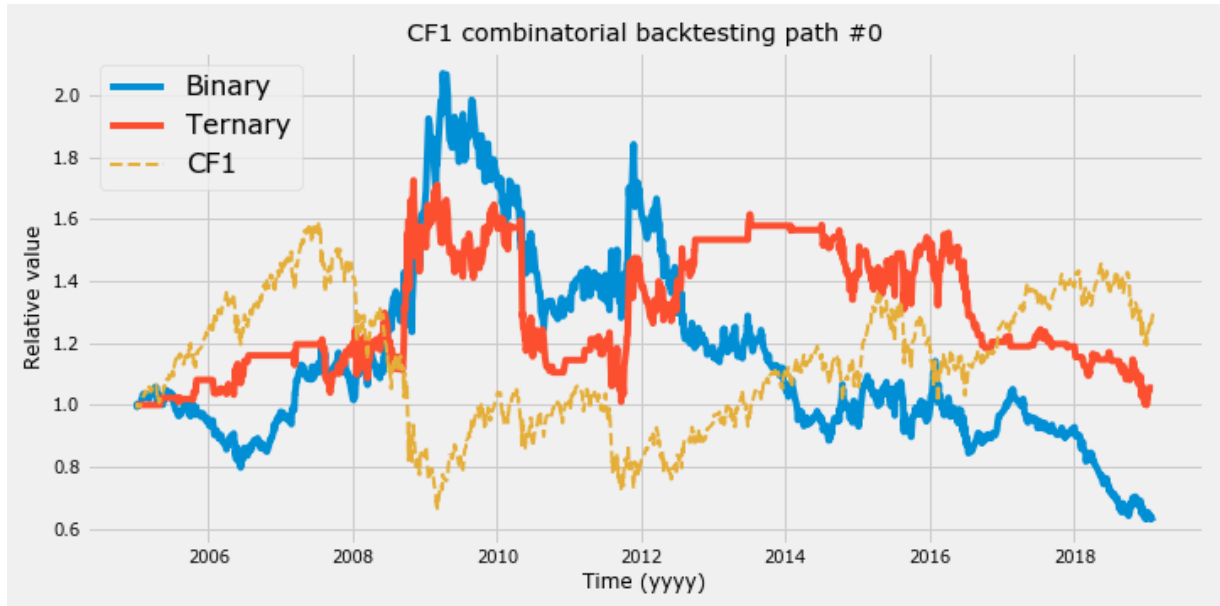


Figure E.16: Depicts the first paths generated from the combinatorial backtest for the asset *CF1*.

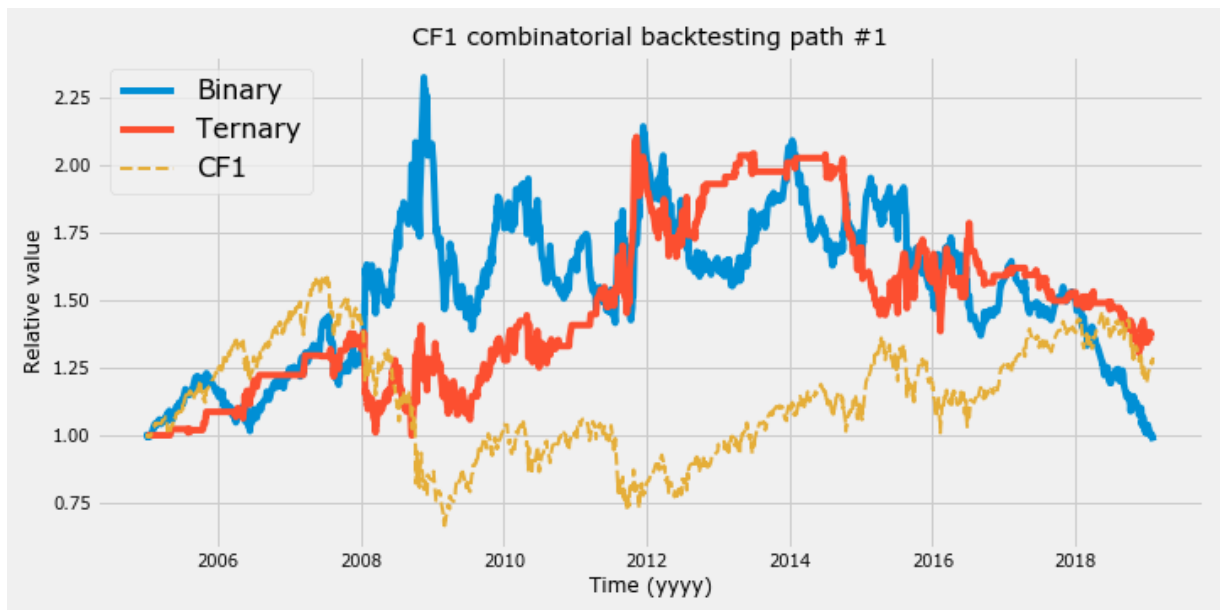


Figure E.17: Depicts the second paths generated from the combinatorial backtest for the asset *CF1*.

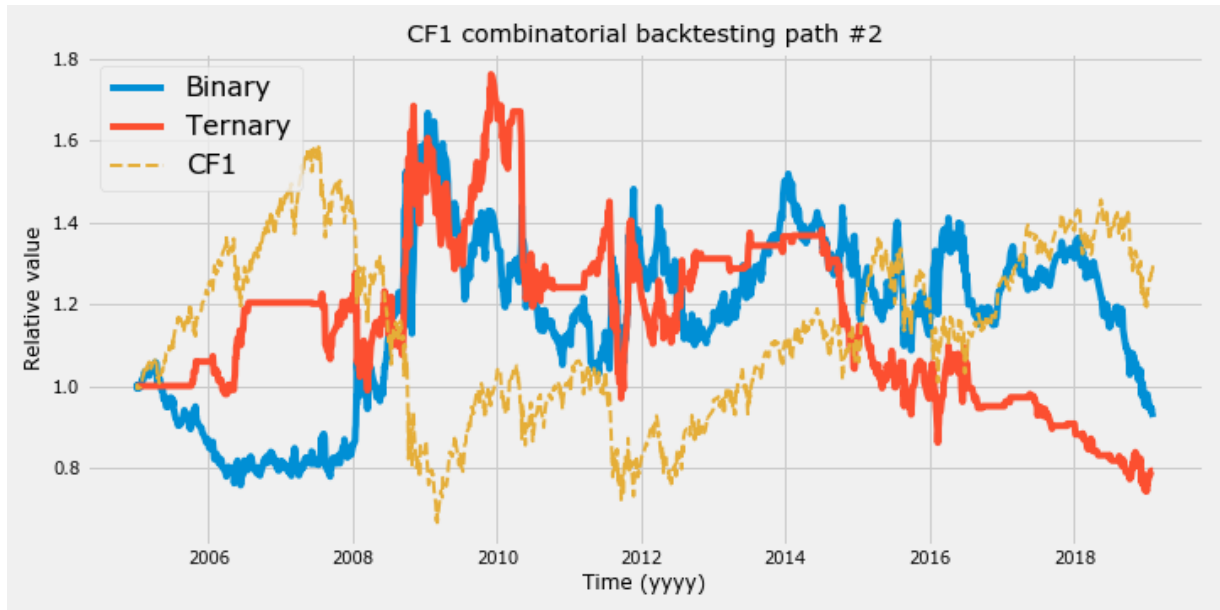


Figure E.18: Depicts the third paths generated from the combinatorial backtest for the asset *CF1*.

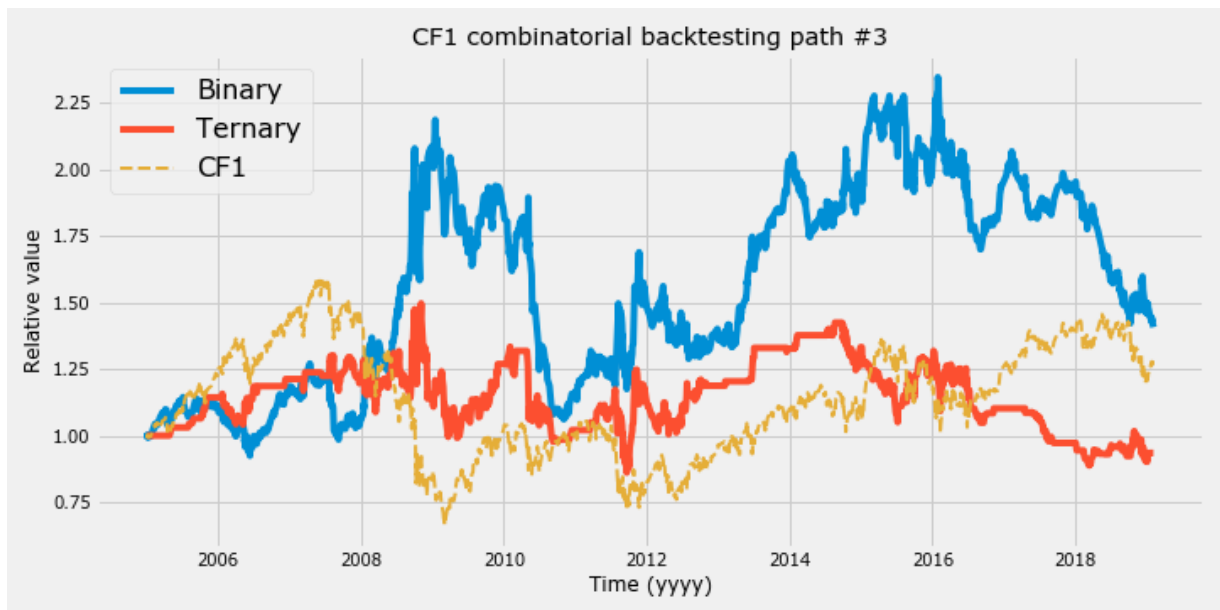


Figure E.19: Depicts the fourth paths generated from the combinatorial backtest for the asset *CF1*.

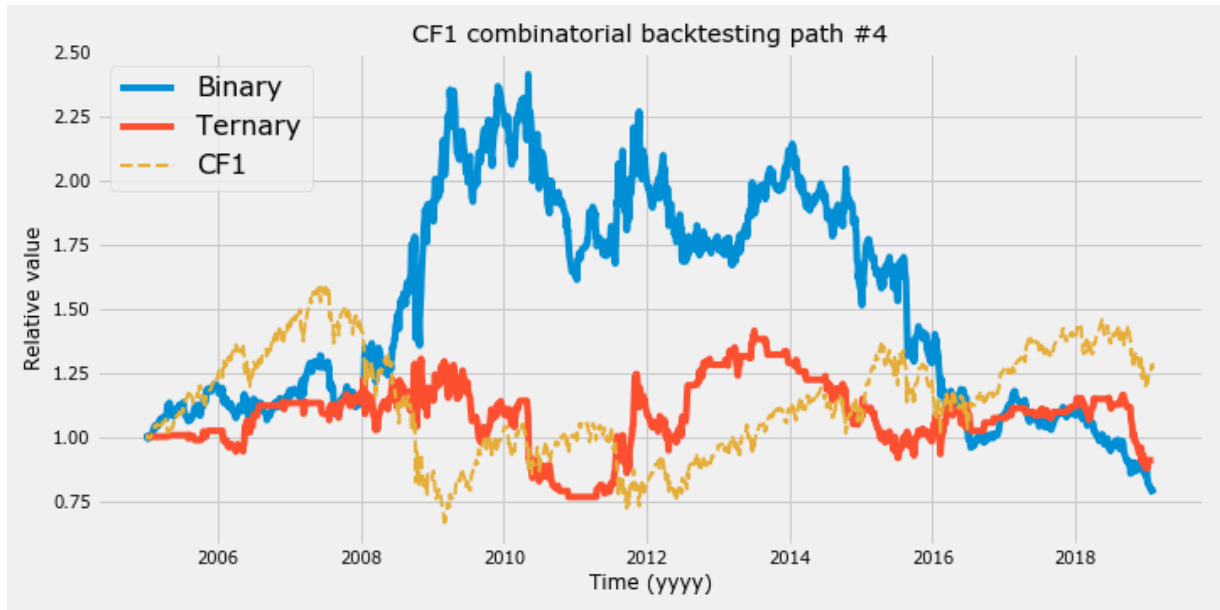


Figure E.20: Depicts the fifth paths generated from the combinatorial backtest for the asset *CF1*.

Appendix F

Graphs from consolidated backtests

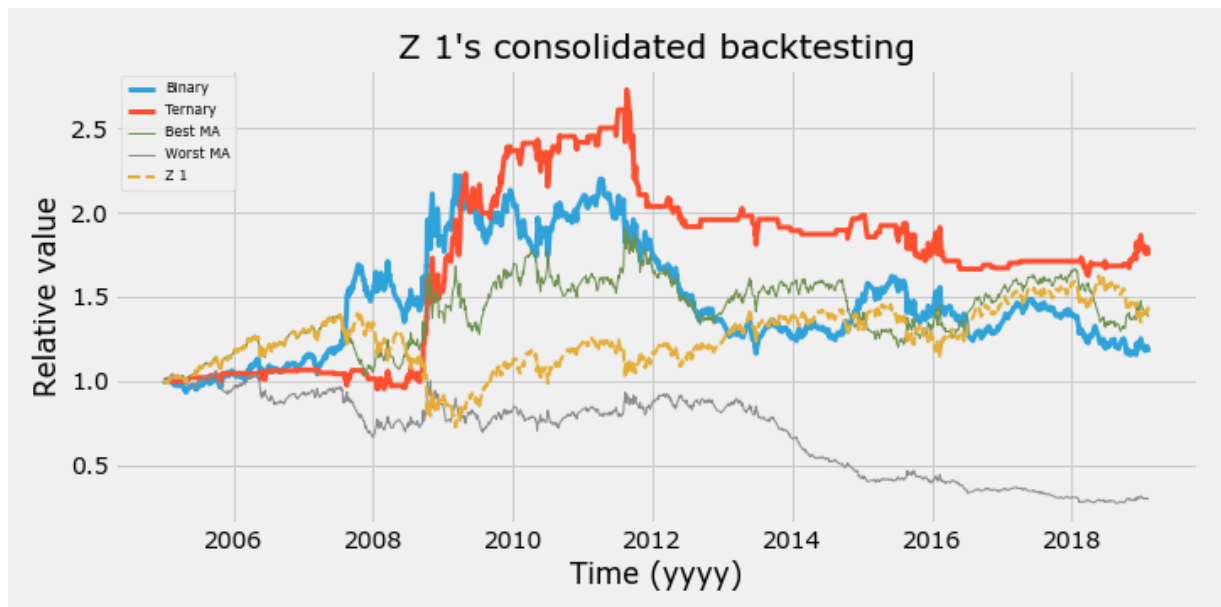


Figure F.1: Consolidated backtesting of $Z 1$ for the binary and ternary cases. Also depicts the relative value of $Z 1$'s contract value and the corresponding best and worst performing Simple trading strategies.

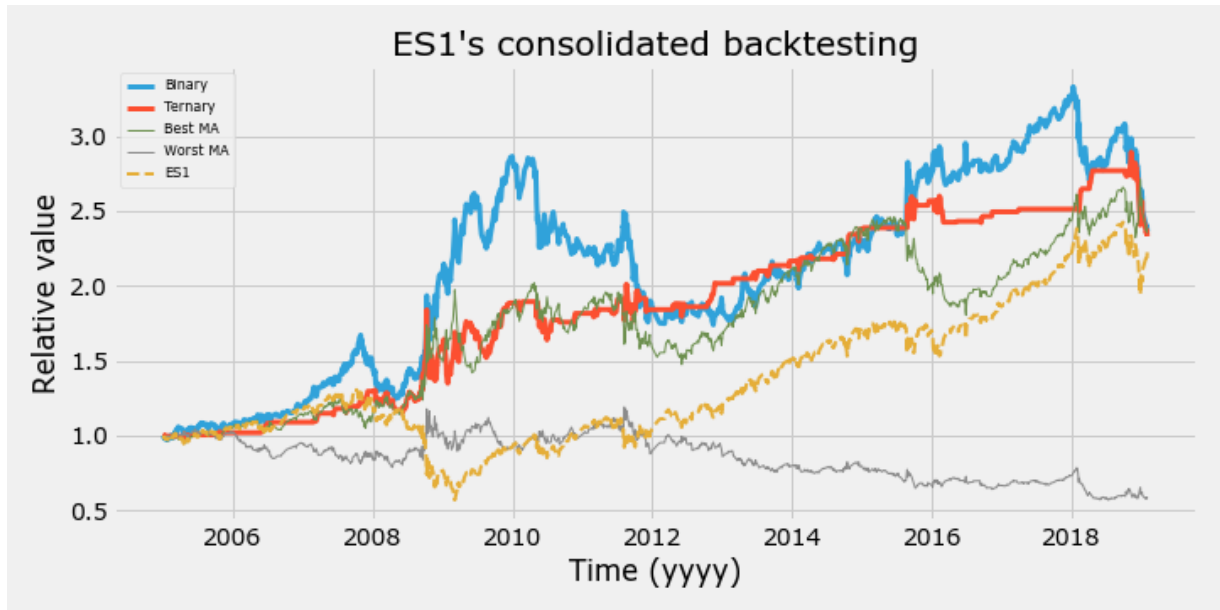


Figure F.2: Consolidated backtesting of *ES1* for the binary and ternary cases. Also depicts the relative value of *ES1*'s contract value as well as the corresponding best and worst performing benchmarks

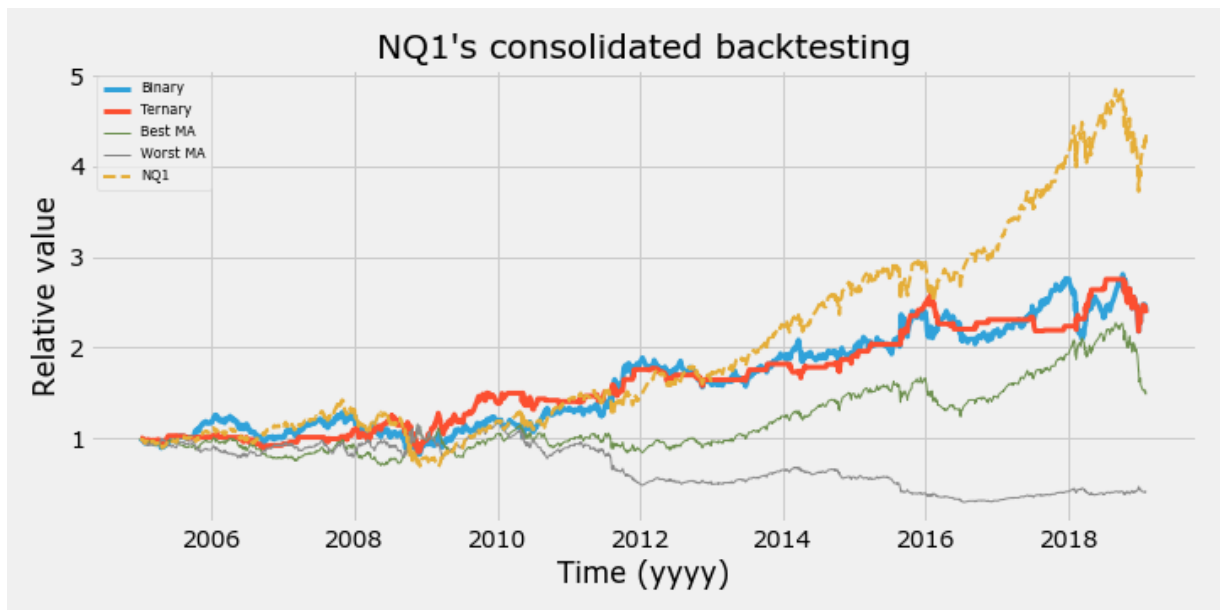


Figure F.3: Consolidated backtesting of *NQ1* for the binary and ternary cases. Also depicts the relative value of *NQ1*'s contract value as well as the corresponding best and worst performing benchmarks

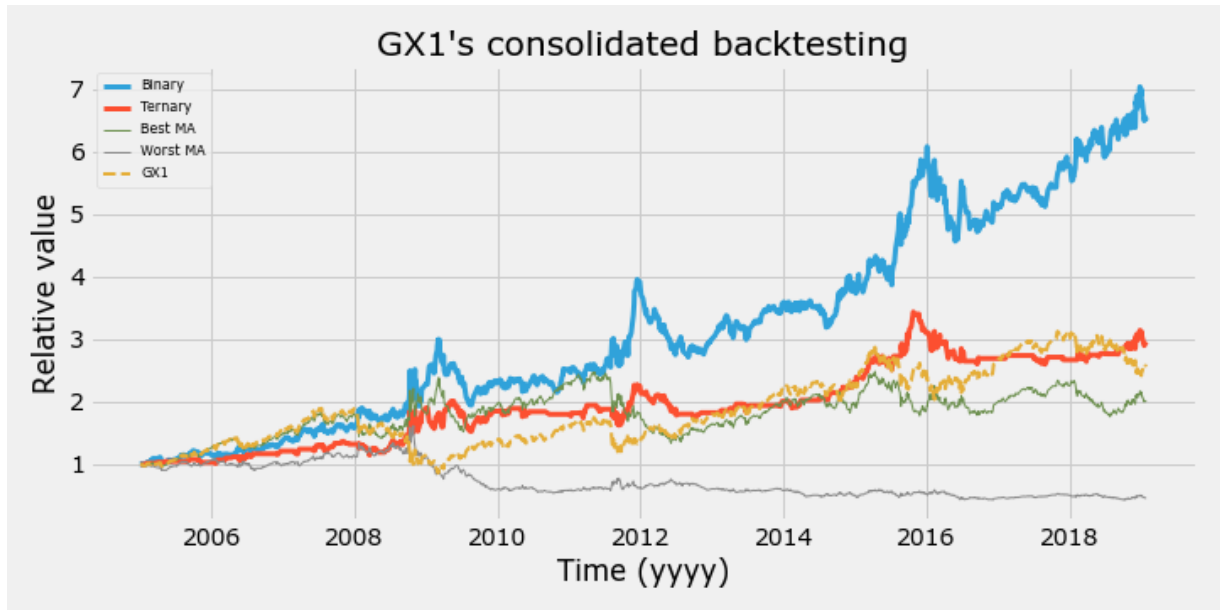


Figure F.4: Consolidated backtesting of *GX1* for the binary and ternary cases. Also depicts the relative value of *GX1*'s contract value as well as the corresponding best and worst performing benchmarks

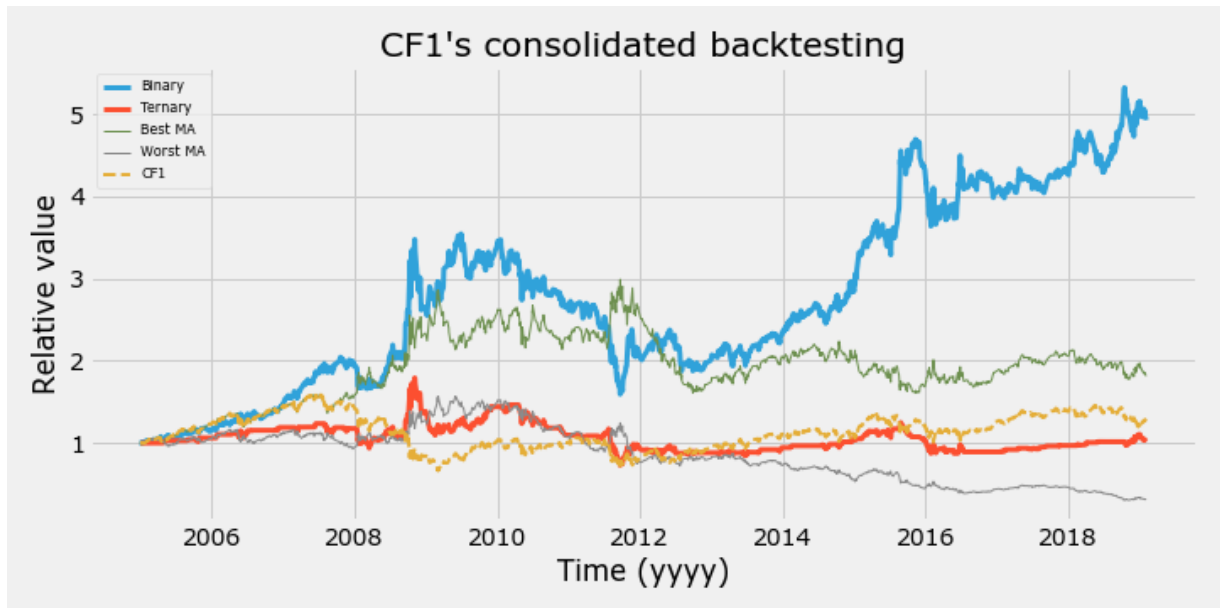


Figure F.5: Consolidated backtesting of *CF1* for the binary and ternary cases. Also depicts the relative value of *CF1*'s contract value as well as the corresponding best and worst performing benchmarks

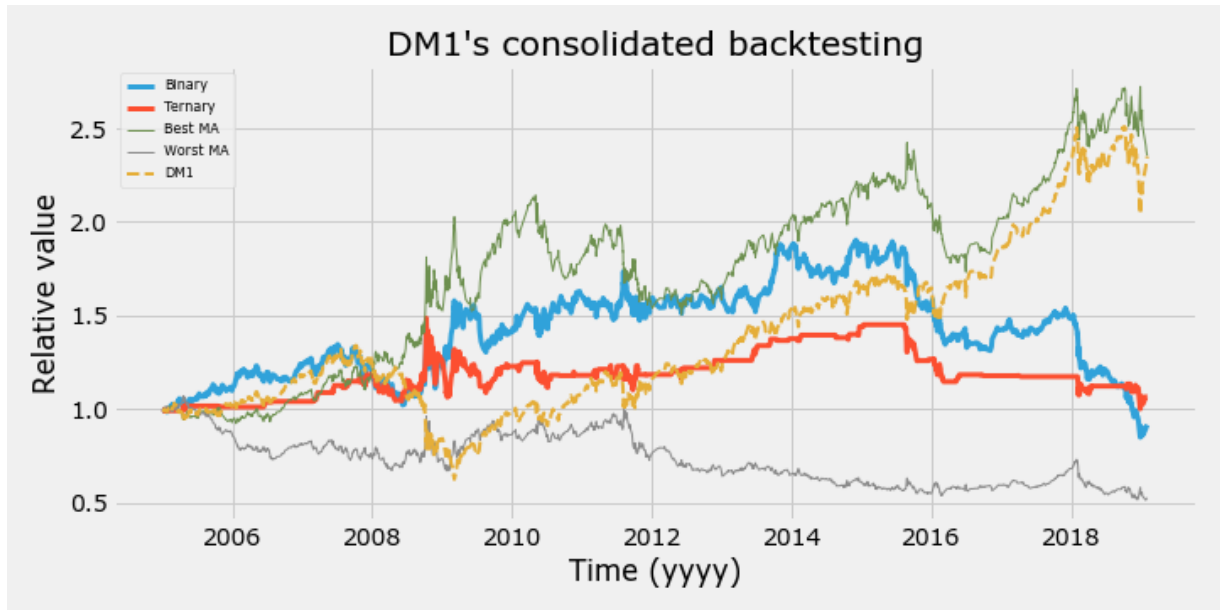


Figure F.6: Consolidated backtesting of $DF1$ for the binary and ternary cases. Also depicts the relative value of $DM1$'s contract value as well as the corresponding best and worst performing benchmarks

Master's Theses in Mathematical Sciences 2019:E31
ISSN 1404-6342
LUTFMS-3368-2019
Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden
<http://www.maths.lth.se/>