UNIVERSITY OF SOUTHAMPTON

Automated Algorithmic Trading: Machine Learning and Agent-based Modelling in Complex Adaptive Financial Markets

by

Ash Booth

Supervisors: Dr. Enrico Gerding & Prof. Frank McGroarty
Examiners: Prof. Alex Rogers & Prof. Dave Cliff

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Business and Law
Southampton Management School

Faculty of Physical Sciences and Engineering
Electronics and Computer Science

Institute for Complex Systems Simulation

April 2016

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

Faculty of Business and Law
Southampton Management School

Faculty of Physical Sciences and Engineering
Electronics and Computer Science

Doctor of Philosophy

by Ash Booth

Over the last three decades, most of the world's stock exchanges have transitioned to electronic trading through limit order books, creating a need for a new set of models for understanding these markets. In this thesis, a number of models are described which provide insight into the dynamics of modern financial markets as well as providing a platform for optimising trading and regulatory decisions.

The first part of this thesis proposes an autonomous system that uses novel machine learning techniques to predict the price return over well documented seasonal events and uses these predictions to develop a profitable trading strategy. The DAX, FTSE 100 and S&P 500 are explored for the presence of seasonality events before an automated trading system based on performance weighted ensembles of random forests is introduced and shown to improve the profitability and stability of trading such events. The performance of the models is analysed using a large sample of stocks and the results show that the system described in this section produces superior results in terms of both profitability and prediction accuracy compared with other ensemble techniques.

The second part of this thesis explores price impact. For many players in financial markets, the price impact of their trading activity represents a large proportion of their transaction costs. This section of the thesis proposes an adaptation of the system introduced in the first part for predicting the price impact of order book events. The system's performance is benchmarked using ensembles of other popular regression algorithms including: linear regression, neural networks and support vector regression using depth-of-book data from the BATS Chi-X exchange. The results show that recency-weighted ensembles of random forests produce over 15% greater prediction accuracy on out-of-sample data, for 5 out of 6 timeframes studied, compared with all benchmarks. Finally, a novel procedure for extracting the directional effects of features is proposed and used to explore the features most dominant in the price formation process.

The final part of this thesis addresses the requirement for testing algorithmic trading strategies laid out in the Markets in Financial Instruments Directive (MiFID) II by describing an agent-based simulation. Five types of agent operate in a limit order market

producing a model that is able to reproduce a number of stylised market properties including: clustered volatility, autocorrelation of returns, long memory in order flow, concave price impact and the presence of extreme price events. The model is found to be insensitive to reasonable parameter variations. Finally, the model is used to explore how trading strategy affects the implementation shortfall of trading a large order. A number of execution strategies with various order types, are evolved and evaluated in the agent-based market. It is shown that the evolved strategies outperform the simple, well-known strategies significantly, suggesting that execution strategy plays an important role in determining the implementation shortfall of trading large orders.[1]

[1]This work was supported by an EPSRC Doctoral Training Centre grant (EP/G03690X/1).

For my family

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

D	A data set
N	The number of data-points in a data set
M	The number of features in a data set
X	The complete feature space
x_i	The feature vector of datapoint i
Y	The complete target variable space
y_i	The variable to be predicted
$\mathcal{W}_i$	The weight vector of datapoint i
$h(x)$	A weak prediction rule that maps x to y
$F(x)$	A strong prediction rule that maps x to y
σ_t^2	The variance at time t
Λ	Decay factor for exponential moving average calculation of variance
$r_{g,t}$	Rate of return of g at time t
VaR_t	Value at risk at time t
ν_t	The VaR scaling factor at time t
$\mathcal{VI}_j$	The importance of feature j in a single random forest (RF)
$\mathbf{Vi}_j$	The importance of feature j in the multiple RF model
$e_{\theta,j}$	The RMSE of tree θ before permuting feature j
$e_{\theta,\pi j}$	The RMSE of tree θ after permuting feature j
Θ	The number of trees in a random forest
$\mathfrak{I}$	The number of random forests in an ensemble
$\mathfrak{I}_{max}$	The maximum number of random forests allowed in an ensemble
$\hat{\sigma}$	The standard deviation of the differences between $e_{\theta,j}$ and $e_{\theta,\pi j}$
$S_{i,t}$	The prediction of random forest i at time t
$k_{i,t}$	The historical performance of random forest i at time t
λ	A smoothing parameter for controlling the recency weighting of experts
$w_{i,t}$	The weight of expert i at time t
$P_{g,t}$	The combined prediction of all experts for stock g at time t
n_t	The total number of experts at time t
Buy_t	The fraction of experts that suggest buying at time t
Sell_t	The fraction of experts that suggest selling at time t
$\mathcal{D}_t$	The difference between Buy_t and Sell_t

α_0	Threshold for ruling out weak predictions
$d_{g,t}$	The maximum drawdown of the algorithm in stock g at time t
$\mathbf{d}_t$	The vector of maximum drawdowns for the algorithm for all stocks
$\Phi_{g,t}$	Size of an order in stock g at time t
$q_{g,t}$	The size of the algorithm's position in stock g at time t
C_t	The algorithm's dollar wealth at time t
$\eta_{i,t-1}$	The RMSE of the last prediction made by RF i at time $t-1$
LO^0	A limit order at the current best price
LO'	A limit order within the best prices
MO^0	A market order whose volume $<$ outstanding volume at the best price
MO'	A market order whose volume $\geq$ outstanding volume at the best price
CA^0	A cancellation within the best price queue
CA'	A cancelation at the best price that removes all available volume
MN	A modification of an existing order at the best price.
τ	Agent type.
δ_τ	Probability of agent τ acting.
mm	Market maker agent.
lc	Liquidity consumer agent.
mr	Mean reversion agent.
mt	Momentum trading agent.
nt	Noise trader agent.
Φ	Volume available at the opposing best price.
v_{min}	Min. volume for market maker.
v_{max}	Max. volume for market maker.
v	Opposing volume.
h_{min}	Min. volume for liquidity consumer.
h_{max}	Max. volume for liquidity consumer.
n_r	Momentum length.
κ	ROC threshold.
v_{mr}	Order volume for mean reversion traders.
P_B	Probability of buying
λ_m	Market order probability
λ_l	Limit order probability
λ_c	Cancel order probability
μ_{mo}, σ_{mo}	market order size distribution parameters
μ_{lo}, σ_{lo}	limit order size distribution parameters
λ_{crs}	crossing limit order probability
λ_{inspr}	inside-spread limit order probability
λ_{spr}	spread limit order probability
λ_{offspr}	off-spread limit order probability

Declaration of Authorship

I, ASH BOOTH, declare that the thesis entitled

AUTOMATED ALGORITHMIC TRADING: MACHINE LEARNING AND AGENT-BASED MODELLING IN COMPLEX ADAPTIVE FINANCIAL MARKETS

and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is clearly attributed.
- Where I have quoted from the work of others, the source is given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed,

Acknowledgements

A slab of incredible good fortune fell right out of the sky and landed on my head when I was accepted for a place on the Institute for Complex Systems Simulation Doctoral Training Programme. First and foremost, I owe many thanks to my supervisors, Prof. Frank McGroarty and Dr. Enrico Gerding, for their expert advice and guidance throughout this project. Their continued support and passion for interdisciplinary research has been an inspiring force over the last few years.

It takes an entire village to keep an early career scientist's self-worth stoked (or in check, as appropriate). I could not have asked for more patient and supportive partisans than my parents, Sally and Phil Booth, nor would I have survived without the remarkable patience, positivity and passion of my phenomenal partner Jazmin Curzon.

Thanks also to friends near and far - there are too many of you to name but I wouldn't have made it through without you all.

Finally, I am extremely thankful for the depth-of-book data that was provided by *G-Research* and the insightful comments provided by various reviewers. I am also extremely grateful to all of the folks at the Institute of Complex Systems Simulation for providing a strong research foundation. In particular, Nicki Lewin, Jason Noble and Seth Bullock have been invaluable sources of support.

Abbreviations

ADT alternating decision tree

ABM agent-based model

ANN artificial neural network

ARIMA autoregressive integrated moving average

AI artificial intelligence

CDA continuous double auction

CFTC Commodity Futures Trading Commission

CME Chicago Mercantile Exchange

CV cross validation

DAX Deutsche Borse Ag German Stock Index

DE differential equation

DJIA Dow Jones Industrial Average

DMA direct market access

EC European Commission

ECN electronic communication network

EMA exponential moving average

ESMA European Securities and Markets Authority

ETF exchange-traded fund

EU European Union

FIFO first-in first-out

FPRD feature partition response differencing

FTSE 100 Financial Times Stock Exchange 100 index

FSA Financial Services Authority

GA genetic algorithm

HFT high frequency trading

IPO initial public offering

KOSPI Korea Composite Stock Price Index

KPSS Kwiatkowski, Phillips, Schmidt and Shin

LOB limit order book

MAPE mean absolute percentage error

MBC market-based control

MD max drawdown

MiFID Markets in Financial Instruments Directive

MLNN multi-layer feed forward neural network

MSE mean square error

MAD mean absolute deviate

MTF multi-lateral trading facilities

NYSE New York Stock Exchange

PLAT Penn-Lehman Automated Trading

RF random forest

RMSE root-mean-square error

RSI relative strength index

S&P 500 Standard & Poor's 500 index

SEC Securities and Exchange Commission

SVM support vector machine

SVR support vector regression

TAC Trading Agent Competition

UAV unmanned aerial vehicles

VaR value at risk

ZI Zero-Intelligence

ZIP Zero-Intelligence-Plus

Chapter 1

Introduction

The primary role of financial markets is to enable transactions between those that wish to buy and sell the same good and thus provide liquidity. As such, markets act as a means of price formation, taking into account all relevant information and determining prices for assets. The liquidity and price formation are emergent properties of the low level interactions of the buyers and sellers that make up the market.

The field of market microstructure research is dedicated to understanding the process by which the interaction of market participants leads to both liquidity and price formation. Specifically, in microstructure analysis, quantitative models of the trading process are devised in order to improve the efficiency of the allocation of goods, via financial markets, in the economy. The merit of microstructure analysis was highlighted in 1994 when Christie and Schultz (1994) found that the bid-ask spreads (the difference between the best bid and ask prices for a particular security) on the NASDAQ stock exchange were larger than was statistically likely, putting forward that dealers were colluding to widen them. A subsequent study confirmed this observation, leading to a lawsuit against NASDAQ and the introduction of new trading regulations. One such initiative, in the US, was regulation ATS, which permitted electronic communication networks (ECNs) - alternative off-exchange trading platforms - the option of either registering as stock exchanges or being regulated under a separate standard. In the EU, MiFID followed suit, sowing the seed for ECNs and multi-lateral trading facilities (MTF) to pop up across Europe. This was the beginning of a climatic shift in the landscape of global equity markets, reducing transaction costs and increasing competition.

Consequently, over the last two decades, trading mechanisms and regulation have evolved at a considerable pace. Computer technology has revolutionised financial markets, with trading moving from pits full of traders to warehouse-like data centres packed with racks of co-located servers running programs to process market data and submit orders in fractions of a second. As a result, financial exchanges have become more cost effective, leading to improved spreads, execution times and brokerage commissions (Chordia et al.,

2008; Angel et al., 2011). In addition, it has become possible to implement a range of trading strategies using completely automated algorithmic computer systems (so called "robot traders"). Such electronic trading now dominates most major financial markets. Although the anonymity of most platforms makes figures hard to glean, it is estimated that the volume of software executed trades in US equity markets has risen from 16% in 2000 to 82% in 2009 (Joyce et al., 2010) and continues to increase.

Although regulations and costs are the same for all traders, the explosion of computer technology has led to a diversification and specialisation of their strategies. Such rapid evolution has lead to a somewhat disjointed vocabulary for describing computer driven trading activities and, for the sake of clarity, we will adhere to the definitions recently laid out by the European Commission (EC).

In 2011, after advice from the European Securities and Markets Authority (ESMA), the EC published proposals to amend MiFID and recast it as a new directive MiFIDII. Intense political debate and a number of drafts and revisions meant that agreement between the European Union (EU) institutions was not reached until February 2014, with the final text published in June 2014 (European Union, 2014a).

MiFIDII came to be as a result of of increasing fears that algorithmic trading had the potential to cause market distortion over unprecedented timescales. Particularly, there were concerns over increased volatility, high cancelation rates and the ability of algorithmic systems to withdraw liquidity at any time. Thus, MiFIDII introduces tighter regulation over algorithmic trading, imposing specific and detailed requirements over those that operate such strategies. This increased oversight requires clear definitions of the strategies under regulation.

MiFIDII defines *algorithmic trading* as the use of computer algorithms to automatically determine the parameters of orders, including: trade initiation, timing, price and modification/cancelation of orders, with no human intervention. This definition specifically excludes any systems that only deal with order routing, order processing, or post trade processing where no determination of parameters is involved.

The level of automation of algorithmic trading strategies varies greatly. Brokers and large sell side institutions tend to focus on optimal execution, where the focus of the algorithmic trading is to minimise the market impact of orders. These algorithms focus on order slicing and timing. Other institutions, often quantitative buy-side firms, attempt to automate the entire trading process. These algorithms may have full discretion regarding their trading positions and encapsulate: price modelling and prediction to determine trade direction, initiation, closeout and monitoring of portfolio risk. This type of trading tends to occur via direct market access (DMA) or sponsored access.

Under MiFIDII, *high frequency trading (HFT)* is considered a subset of algorithmic trading. The EC defines HFT as any computerised technique that executes large numbers

of transactions in fractions of a second using:

- Infrastructure designed for minimising latencies, such as proximity hosting, colocation or DMA.
- Systematic determination of trade initiation, closeout or routing without any human intervention for individual orders; and
- High intra-day message rates due to volumes of orders, quotes or cancellations.

Such strategies seek opportunities on very small timescales, from milliseconds to seconds. Many high-frequency algorithms adopt a market-maker strategy, attempting to keep a relatively neutral position, providing liquidity (most of the time) while taking advantage of price discrepancies. Many other strategies exist, often invoking methods from time-series analysis, machine learning or artificial intelligence (AI) to predict movements and isolate trends. For HFT, monitoring the overall inventory risk and incorporating this information into pricing/trading decisions is vital.

The transition to electronic trading using limit order books (LOBs) and the subsequent rise of the computerised methodology listed above has created the need for a new breed of quantitative models to describe such markets. In particular, for large players that regularly transact in these markets, the price impact of their trading activity represents a large proportion of their transaction costs which are in turn considered a substantial determinant of investment performance (Guobuzaite et al., 2004). As well as affecting the performance of an active investment strategy, they also affect how rapidly assets may be converted into cash. As such, it is vital that firms are able to quantify and predict the potential impact of their trading activity; this is an integral part of algorithmic trading.

Although it began over thirty years ago, automated and algorithmic trading is by no means fully accepted or understood in an academic context. Since its introduction, recurring periods of high volatility and extreme stock price behaviour have plagued the markets. Such periods have been linked to trading algorithms, and their frequent occurrence has undermined investors' confidence in the current market structure and regulation (SEC and CFTC, 2010). So called "Flash Crashes" are becoming ever more frequent with evidence of over $18,000$ of them occurring between 2006 and 2011 in various stocks (Johnson et al., 2013). One of the more well know incidents of market turbulence is the Flash Crash of the 6^{th} May 2010.

At 14:32, began a trillion dollar stock market crash that lasted for a period of only 36 minutes (Kirilenko et al., 2014). Particularly shocking was not the largest ever intra-day loss of the Dow Jones but the sudden rebound of most securities to near their original values. This breakdown resulted in the second-largest intraday point swing ever witnessed, at 1,010.14 points. A mere two weeks after the crash, the Securities and Exchange Commission (SEC) and Commodity Futures Trading Commission (CFTC)

released a joint report that did little but quash rumours of terrorist involvement. During the months of silence that followed, there was a great deal of speculation about the events on May 6th with the identification of a cause made particularly difficult by the increased number of exchanges, use of algorithmic trading systems and speed of trading. Finally, the SEC and CFTC released their report on September 30th concluding that patient zero was a single algorithmic order that executed a large sale of futures contracts in an extraordinarily short amount of time from fund management firm Waddell & Reed (W&R) (SEC and CFTC, 2010).

The report was met with mixed responses. Though some studies seemed to agree with the report, stating that HFTs did not initiate the Flash Crash and that their responses to the unusually large selling pressure may simply have exacerbated market volatility (Kirilenko et al., 2014), others vehemently disagreed. Eric Hunsader, of Nanex LLC, a firm specialising in the distribution and analysis of high frequency data, has openly voiced his issues with the report:

> "Based on interviews and our own independent matching of the 6,438 W&R executions to the 147,577 Chicago Mercantile Exchange (CME) executions during that time, we know for certain that the algorithm used by W&R never took nor required liquidity. It always posted sell orders above the market and waited for a buyer; it never crossed the bid/ask spread. That means that none of the 6,438 trades were executed by hitting a bid. It is widely believed that the 'sell program' refers to the algo selling the W&R contracts. However, based on the statements above, this cannot be true. The sell program must be referring to a different algo, or Kirilenko's analysis is fundamentally flawed, because the paper incorrectly identifies trades that hit the bid as executions by the W&R algo." (Hunsader, 2012).

A number of established academics also disagree with the SEC's report. Menkveld and Yueshen (2013) analysed W&R's orderflow and identified an alternative narrative. They did not conclude that the crash was simply the price W&R were required to take for demanding immediacy in the S&P 500. Instead, they found that cross-market arbitrage, which provided e-mini sellers with increased liquidity from S&P 500 buyers in other markets, broke down minutes before the crash. As a result of the breakdown, W&R were forced to find buyers only in E-mini and so they decelerated their selling. An extreme response (in terms of price and selling behaviour) then resulted in W&R paying a disproportionately high price for demanding liquidity.

Easley et al. (2011) show that major liquidity issues were percolating over the days that preceded the Flash Crash. They note that immediately prior to the large W&R trade, volume was high and liquidity was low. Using a technique developed in previous research (Easley et al., 2010), they suggest that, during the period in question, order flow was

becoming increasingly toxic. They go on to demonstrate how, in a high-frequency world, such toxicity may cause market makers to exit - sowing the seeds for episodic liquidity. Of particular note, the authors express their concern that an anomaly like this is highly likely to occur, once again, in the future.

Another infamous crash occurred on the 23rd March 2012 during the initial public offering (IPO) of a firm called BATS. The stock began trading at 11:14 a.m. with an initial price of $15.25. Within 900 milliseconds of opening, the stock price had fallen to $0.28 and within 1.5 seconds, the price bottomed at $0.0007. Yet another technological incident was witnessed when, on the 1st August 2012, the new market-making system of Knight Capital was deployed. Knight capital was a world leader in automated market making and a vocal advocate of automated trading. The error occurred when testing software was released alongside the final market-making software. According to Knight's official statement:

> "Knight experienced a technology issue at the open of trading... this issue was related to Knight's installation of trading software and resulted in Knight sending numerous erroneous orders in NYSE-listed securities... which has resulted in a realised pre-tax loss of approximately 440 million [dollars]."

This 30 minutes of bogus trading brought an end to Knight's 17 year existence, with the firm subsequently merging with a rival.

The all-too-common flash crashes are a dramatic consequence of the growing complexity of modern financial markets and have not gone unnoticed by the regulators. In November 2011, the European Union (2011) made proposals for a revision of the Markets in Financial Instruments Directive (MiFID). Although this directive only governs the European markets, according to the World Bank (2012) (in terms of market capitalisation), the EU represents a market around two thirds of the size of the US. In the face of declining investor confidence and rapidly changing markets, a draft of MiFID II was produced. After nearly three years of debate [1], on the 14th January 2014, the European Parliament and the Council reached an agreement on the updated rules for MiFID II, with a clear focus on transparency and the regulation of automated trading systems (European Union, 2014b).

Specifically, MiFID II introduces rules on algorithmic trading in financial instruments. Any firm participating in algorithmic trading is required to ensure it has effective controls in place, such as circuit breakers to halt trading if price volatility becomes too high. Also, any algorithms used must be tested and authorised by regulators. We find the last

[1] Many of the articles in the original draft of the MiFID II proposal were met with accusations of tokenistic politics from both industry and academia (see http://www.bankingtech.com/81761/mifid-ii-is-a-dogs-dinner-says-former-uk-government-advisor/).

requirement particularly interesting as MiFID II is not specific about how algorithmic trading strategies are to be tested.

From the arms race in trading technology and the rapidly changing regulatory landscape arises a complexity that presents enormous challenges for understanding, analysis and prediction in this brave new market. How do changes in the order book affect price impact? What drives price movement? How do we quantify and predict liquidity? Fortunately, electronic markets also generate vast amounts of data.

Traditionally, such problems have been approached using differential equation (DE) models. Such models have a deep-rooted history in the social sciences, from business cycles (Samuelson, 1939) and innovation diffusion (Bass, 1969) to epidemiology (Andersson and Britton, 2000) and options pricing (Black and Scholes, 1973). The DE class of models is broad, encompassing a wide range of feedback effects but, for the sake of mathematical tractability, typically aggregating agents into a relatively small number of states. Importantly, agents within each compartment are assumed to be homogeneous and well mixed with the transitions among states modelled as their expected value. In finance, such assumptions are rarely true.

Contrastingly, in agent-based models (ABMs), the model consists of a set of agents that encapsulate the behaviours of the various individuals that make up the system, and execution consists of emulating these behaviours (Parunak et al., 1998). Thus, ABMs explore heterogeneity of agents and the network structure of their interactions. In such a new, ill-explored and data-rich complex system, an agent-oriented approach, with its emphasis on autonomous actions and interactions, is an ideal method for addressing theses questions. Specifically, in such systems, a constituent agent is capable of local decision making with only incomplete and imperfect knowledge of the environment. The market system, viewed as the interacting of various autonomous agents, behaves as a computational ecosystem in which the agents interact and strategically compete for resources.

1.1 Research Aims

Against this background, our aims are threefold. First, we aim to investigate whether it is possible to utilise novel machine learning methods to understand and predict daily market price movements. It should be noted that we are not trying to develop a state-of-the-art trading system. Instead, we aim to identify the minimum elements necessary to produce an autonomous trading system and to use the system to further our understanding of the drivers of market dynamics. Such a trading system must: generate signals indicating whether to buy, sell or hold an asset (trading signals) while efficiently managing its risk. We approach the first of these tasks by developing time series analysis and machine learning techniques for finding trends or indicators in historical data. A

risk management layer separates the decision to trade from the trading recommendations made by the machine learning layer. This risk management layer will evaluate the strength of the machine learning layer given current market conditions and, if it deems necessary, stop the agent from trading further or even unwind its position (sell if it is holding assets or buy back if it is short).

Second, we wish to explore whether the methods described above can be successfully applied to high-frequency market-depth data to help quantify how changes in order flow affect the price impact. We devise an adaptation of the system used for daily price prediction to explore the effects of incoming orders on market dynamics.

Finally, given the shift in structure and participant behaviour in modern financial market and the clear need for an environment for analysing automated trading strategies, it is likely that the simple statistical models of old will no longer suffice. We therefore propose an interactive agent-based modelling environment to understand the effects of various trading behaviours and strategies on the dynamics of the limit order book (LOB). Such a model must include agents that represent common market behaviours and must accurately reproduce empirically observed values for the well known stylised facts of limit order markets. Such abilities are a crucial step towards a viable platform for the testing of trading algorithms as outlined in MiFIDII.

1.2 Contributions

To address the challenges involved in understanding and explaining financial market phenomena, we present novel algorithms in Chapters 3 and 4 that provide the ability to make predictions about a number of financial metrics. In Chapter 5 we develop a novel agent-based modelling environment for understanding algorithmic interactions in limit order book markets.

In more detail, in Chapter 3, we describe an automated trading system that, for the first time, uses performance-weighted ensembles of random forests to predict the price return during well documented seasonality events. While others have investigated the performance of ensemble systems and random forest algorithms for predicting using unfiltered daily stock prices, as yet, no study has explored the use of ensembles of random forests for building expert systems for trading empirical regularities in stock markets. Specifically, this section makes the following contributions to the state of the art:

- We present the first ensemble learning system for trading seasonal trends and demonstrate its effectiveness on equity market data.
- The highly active fields of online ensemble generation and random forest predictors are fused to produce a novel expert system for stock trading formed from

performance weighted ensembles of random forests.

- Using experiments based on real data, we show that our recency biased performance-weighted ensembles of random forests, used for trading seasonal events, outperforms other ensemble methods. These include cumulative weighting systems, simple averaging of regressors as well as a number of non-ensemble regression techniques.

These contributions are described in the following paper:

A. Booth, E. Gerding, and F. McGroarty. Automated trading with performance weighted random forests and seasonality. *Expert Systems with Applications*, 41(8):3651–3661, 2014a

In Chapter 4, we present an empirical model for predicting the short term price impact in a limit order book of events that alter the best available prices in the book. Such events include any market orders or limit orders at the current best prices, as well as cancelations that remove all volume at the best quoted price. Specifically, we develop a model, based on performance weighted ensembles of random forests, to forecast the relative change in price 1, 5, 10, 60 and 600 seconds after an event. While other studies have investigated the predictive power of more tradition regression techniques, currently, no study has explored the use of performance weighted ensemble to predict the short term price impact of order book events. We demonstrate that an ensemble of recency-biased, performance-weighted random forests is able to predict the price impact of events more consistently and with greater accuracy than linear regression, neural networks, support vector regression as single algorithms or combined as ensembles. Also, a novel methodology for extracting the directional effects of features in random forest ensembles is proposed and used to explore the most important features in the price formation process. This work is described in full in the following papers:

A. Booth, E. Gerding, and F. McGroarty. Predicting equity market impact with performance weighted ensembles of random forests. In *IEEE Symposium on Computational Intelligence for Financial Engineering and Economics*, pages 1–8. IEEE, 2014c

A. Booth, E. Gerding, and F. McGroarty. Performance-weighted ensembles of random forests for predicting price impact. *Quantitative Fiance*, (ahead-of-print):1–13, 2014b

Finally, in Chapter 5, we describe, for the first time, an agent-based simulation environment that is realistic and robust enough for the analysis of algorithmic trading strategies. In detail, we describe an agent-based market simulation that centres around a fully functioning LOB and populations of agents that represent common market behaviours and strategies: market makers, fundamental traders, high-frequency momentum traders,

high-frequency mean reversion traders and noise traders. We demonstrate that the model accurately reproduces empirically observed values for: the auto-correlation, volatility clustering, kurtosis and variance of price return and order-sign time series; the price impact function and the occurrence of extreme price events.

The model described in this chapter includes agents that operate on different timescales and whose strategic behaviours depend on other market participants. The decoupling of actions across timescales combined with dynamic behaviour of agents is lacking from previous models and is essential in dictating the more complex patterns seen in high-frequency order-driven markets. Consequently, this chapter presents a model that represents more trading behaviours and is able to replicate more of the empirically observed empirical regularities than any previous work. Such abilities provide a crucial step towards a viable platform for the testing of trading algorithms as outlined in MiFID II. The work described in this chapter is described in the following paper, which is currently under review:

A. Booth, E. Gerding, and F. McGroarty. A brave new model for a brave new market. *European Journal of Finance*, (Under-Review), 2015

1.3 Outline of Report

In Chapter 2 we cover some background on LOBs and summarise the current literature on machine learning for predicting daily price changes in equity markets, the current state of the art for modelling price impact in LOBs, as well as the key research on the agent-based modelling of LOBs. In Chapter 3 we describe an automated trading system that uses performance-weighted ensembles of random forests to predict the price returns during well documented seasonality events. Chapter 4 then goes on to propose an adaptation of the random forest ensemble algorithm for predicting the short term price impact in a LOB of events that alter the best available prices. Chapter 6 concludes and proposes future work develop and agent-based modelling environment for understanding LOB dynamics.

Chapter 2

Literature Review

The nature of interdisciplinary research is such that ideas and terminology from a variety of disciplines must be called upon. Thus, although this review may appear to cover a broad spectrum of topics, each provides a relevant background to the research question.

This chapter begins with an introduction to limit order books before outlining the current state of the art in terms of predicting daily price returns using machine learning algorithms. Next, we address the market microstructure literature before discussing the current state of the art in agent-based modelling of financial markets.

2.1 Background - Limit Order Books

For many years, the majority of the worlds financial markets have been driven by a style of auction, very similar to the basic process of haggling, known as the continuous double auction (CDA). In a CDA a seller may announce an offer or accept a bid at any time and a buyer may announce a bid or accept an offer at any time. This continuous and asynchronous process does away with any need for a centralised auctioneer, but does need a system for recording bids and offers and clearing trades. In modern financial markets, this function is performed by a uniform trading protocol known as the limit order book (LOB), whose universal adoption was a major factor in the transformation of financial exchanges (Jain, 2005).

The most common type of order submitted to a LOB is the limit order - an instruction to buy or sell a given quantity of an asset, that specifies a limit (worst acceptable) price which cannot be surpassed. Upon receiving a limit order, the exchange's matching engine compares the order's price and quantity with opposing orders from the book. If there is a book order that matches the incoming order then a trade is executed. The new order is termed aggressive (or marketable) because it initiated the trade, while the existing order from the book is deemed passive. If, on the other hand, there are no

matches for the incoming order it is placed in the book along with the other unmatched orders, waiting for an opposing aggressive order to arrive (or until it is cancelled). A visualisation of the structure and mechanism of a LOB is given in Figure 2.1.

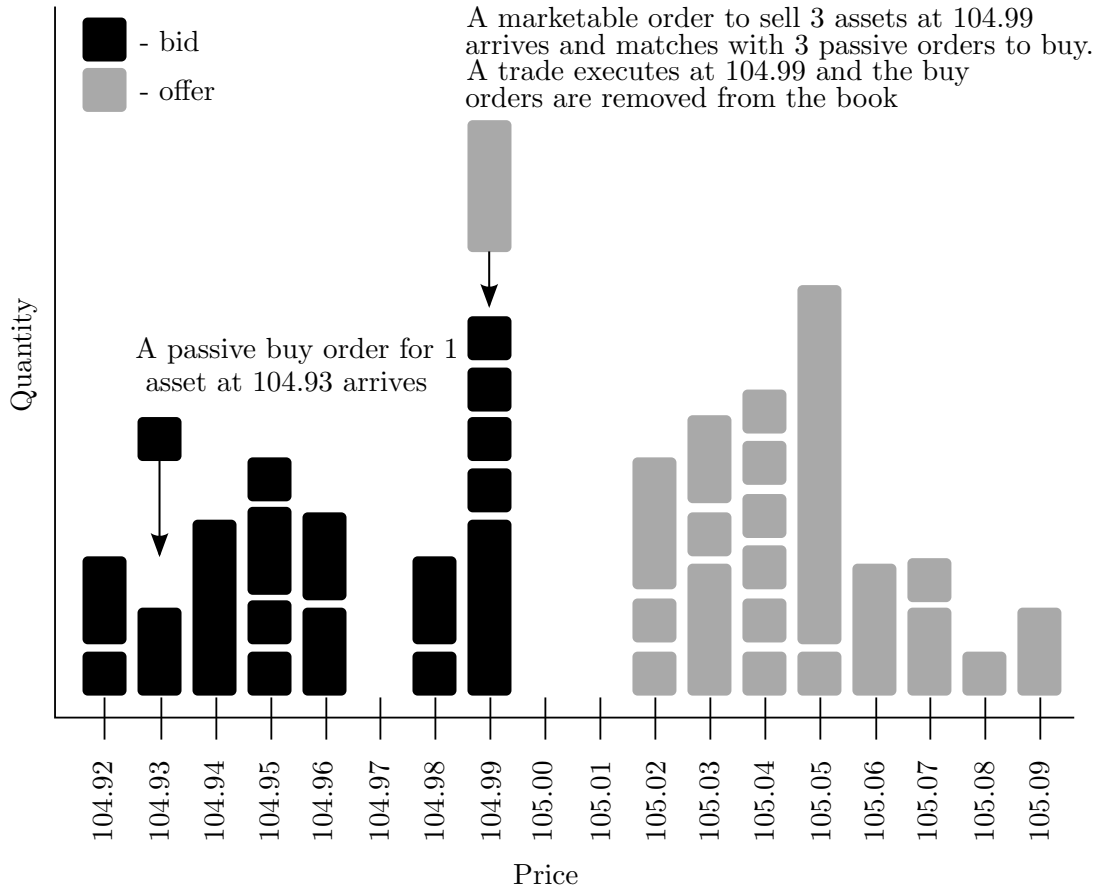


FIGURE 2.1: An illustration of LOB structure and dynamics.

The details of order matching vary across exchanges and assets classes. However, most modern equity markets operate using a price-time priority protocol. That is, the lowest offers and highest bids are considered first, while orders of the same price are differentiated by the time they arrive (with priority given to orders that arrive first). Thus, limit orders with identical prices form a first-in first-out (FIFO) queue.

Most LOB-driven exchanges offer many more order types than the simple limit order. Another particularly common order type is the market order, which ensures a trade executes immediately at the best available price for a given quantity. As a result, market orders demand liquidity and risk uncertainty. Many more order types are available that allow control over whether an order may be partially filled, when an order should become active and how visible the order is. Such order types include: conditional orders, hybrid orders, iceberg orders, stop orders and pegged orders, but the intricacies of these order types are beyond the scope of this report.

Traders interact with a screen-based LOB that summarises all of the "live" (outstanding) bid and offers that have not yet been cancelled or matched. The LOB has two sides: the

ask book and the bid book. The ask book contains the prices of all outstanding asks, along with the quantity available at each price level, in ascending order. The bid book, on the other hand shows the corresponding information for bids but in descending order; this way traders see the "best" prices at the top of both books. A simplified example of what a trader may see when looking at a LOB is given in Figure 2.2.

Stock ABC			
Bids		Asks	
8	104.99	5	105.02
3	104.98	6	105.03

FIGURE 2.2: A simplified example of how a trader would view the limit order book shown in Figure 2.1. We can see that the spread is \$104.99-105.02 and the the volume available at the best prices are 8 on the bid book and 5 on the ask book.

The amount of information available about the LOB at any given time depends on the needs and resources of the traders. Usually the only information that is publicly available (in real time) is the last traded price or the mid-price (the point between the current best prices). Professional traders may chose to subscribe to receive information on the price and size for the best prices, along with the price and size of the last recorded transaction, of an asset of interest; this is known as "level 1" market data. The most informative information, "level 2" or "market depth" data, includes the complete contents of the book (except for certain types of hidden orders) but this comes at a premium. For individual subscribers, the current cost of receiving real time level 2 data for equities from just the New York Stock Exchange (NYSE) exchange is \$5000/month (NYSE EURONEXT, 2013). For a complete picture, One may also want to subscribe to the 15 other US national stock exchanges currently listed with the SEC [1].

At first glance, the rules of limit order trading seem simple but trading in a LOB is a highly complex optimisation problem. Traders may submit buy and/or sell orders at different times, prices, quantities and - in today's highly fragmented markets - often to multiple order books. Orders may also be modified or cancelled at any time. The complexity of LOB strategies presents significant challenges for those attempting to model, understand and predict behaviours. Nonetheless, the well-defined framework and the vast volumes of data generated by the use of LOBs presents an exciting and valuable opportunity for computational modelling.

[1] A list of the stock exchanges registered with the SEC is available at http://www.sec.gov/divisions/marketreg/mrexchanges.shtml.

2.2 Forecasting Price Returns

As we touched upon in the introduction, though an interesting endeavour, our goal here is not to create the world's foremost automated trading system. Instead, we aim to produce a simple, viable and consistently profitable system that enables us to unpick the elements that are important in driving price formation. Thus, we take a data scientist's approach, using novel methods from machine learning to build predictive models and then to analyse those models in order to improve our understanding of the system being forecast. Before proposing a predictive model, we first summarise a number of popular predictive methods from machine learning an discuss their previous applications to the prediction of price returns.

Many attempts have been made to devise a consistently profitable autonomous trading system. Inspiration for such trading systems comes from a variety of fields, ranging from fundamental analysis and econometric modelling to evolutionary computation, machine learning and even news mining (Nuij et al., 2013). The trading and execution algorithms that we will discuss in the next section are intrinsically reactive, purely responding, using predefined rules, to market conditions. In order to be used effectively, such trading algorithms must be combined with advanced techniques that are able to deal with the sudden unexpected shifts that can occur in highly volatile markets. Machine learning offers the opportunity for short term prediction of many market variables in volatile markets. It effectively seeks to find and verify relationships and/or trends and as such the majority of these techniques are entirely statistical in nature.

In this report we will focus on the literature surrounding the use of machine learning to generate trading rules using technical analysis, which uses statistically notable short-term opportunities captured by technical indicators, such as momentum and trending. However, the absence of a solid mathematical foundation for technical analysis has meant that its presence in the academic literature is very limited. Furthermore, during the 1960s, trading rules based on technical indicators were studied and found not to be profitable (Fama and Blume, 1966; Alexander, 1961). It was this work that led notable academics to dismiss technical analysis and support the *efficient market hypothesis* (Fama, 1970). However, the major problem with the early studies of technical analysis was the ad hoc specifications of the trading rules that suggested the use of data dredging.[2] Despite this, profitable technical trading strategies for inter-day trading in the S&P 500 have been found using genetic algorithms, although the strategies did not fair any better than simple buy and hold strategies (where the asset is bought at the start of the test period and sold at the end) when presented with out of sample data (Allen and Karjalainen, 1999). More recently, there has been a surge in work generating trading strategies by using technical indicators as inputs to machine learning models. Below we

[2]Data dredging represents the statistical bias that arises from the misuse of statistics. It implies the deliberate inappropriate use of statistics to uncover misleading relationships in data.

review some of the wide ranging machine learning techniques used to formulate trading strategies.

2.2.1 Artificial Neural Networks

Neural networks have been a particularly popular field of study originating from the desire to better understand the human brain. The field is generally considered to have began with McCulloch and Pitts's (1943) mathematical model of the basic processing unit of the brain - the nerve cells known as neurons. Their model of a neural consists of three simple parts:

Weighted inputs, w_i - that corespond to synapses in the brain.

An Adder - that sums all input signals. This is analogous to the membrane of a neuron, wich collects electrical charge.

An Activation function - that determines whether a neuron fires for a given set of inputs.

The analogy suggests that each of the inputs, x_i, represents the output of other neurons and that each of those neuronal firings flowed along synapses of different strengths. The strength of a synapse is thus captured by its weight, w_i. Now, as all signals arrive into a neuron they are summed to determine if there is enough signal strength to make it fire. This was captured mathematically by McCulloch and Pitts (1943) as:

$$y = \mathbb{1}\left(\sum_i w_i x_i > \Theta\right) \tag{2.1}$$

for some threshold Θ. Thus, the McCulloch and Pitts neuron is a binary threshold device. Though this is a very simple model, Rosenblatt (1958) suggested the perceptron learning algorithm as a way to estimate the weights of a McCulloch and Pitts neuron.

2.2.1.1 The Perceptron

The Perceptron is simply a set of McCulloch and Pitts neurons and the associated weights that bind each input to each neuron. An illustration of a toy example of a perceptron is given in Figure 2.3. In this example, the number of inputs is equal to the number of neurons but this needn't be the case. It is worth noting that neurons in a perceptron act completely independant of one another. That is, each neuron multiplis the inputs by its own weights before summing the result and comparing it to its own threshold. Training a perceptron is therefore an exercise in enabling a perceptron to

learn to reproduce a particular output for a given input. The learning rule laid out by Rosenblatt (1958) is as follows:

$$w_{i,j} \leftarrow w_{i,j} + \eta(t_j - y_j) \cdot x_i \tag{2.2}$$

Where $w_{i,j}$ is the weight that feeds input i to neuron j, t_j is the target output of neuron j (the output that we were hoping for from neuron j), y_j is the actual output from neuron j, x_i is the value of the input, and η is the learning rate that determines how fast the network learns.

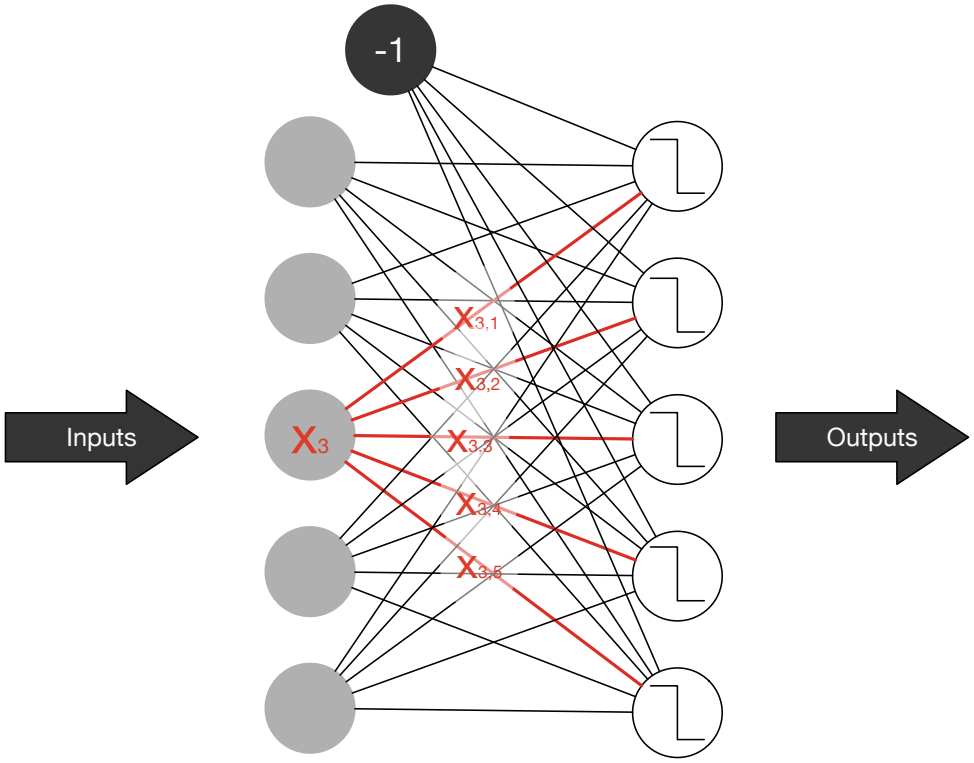


FIGURE 2.3: An example perceptron network comprising a set of inputs (grey circles) connected to a set of McCulloch and Pitts Neurons (white circles) with weighted connections.

In a particularly thorough study by Minsky and Papert (1969), it was shown that what a perceptron learning algorithm does is to find a straight line (or plane or hyperplane) that best separates data where the neurons fire on one side of the line but not the other. Such a line is often referred to as a decision boundary or a discriminant function. Unfortunately, linear models, with no hidden layers, have very limited predictive capabilities and, as such, this greatly reduced interest in this field of study.

2.2.1.2 The Multi-Layer Perceptron

In order to make a neural network more powerful, one can add hidden layers of neurons to enable the approximation of any smooth functional mapping forming a MLNN. However, training networks with hidden layers is much more difficult than with the Perceptron. It was a couple of decades before Rumelhart et al. (1986) discovered how to solve the problem using the backpropagation algorithm [3]. Backpropagation is a form of gradient decent where we compute the gradient of the error s with respect to the weights of the network so that we can adjust the weights and minimise the error function. As this differentiation cannot be done directly, the chain rule of differentiation is used. This produces an update function for each layer, which are applied backwards through the network. The full MLNN algorithm is given in Algorithm 1 below. It was this algorithm which sparked a decade of intense interest in neural network models.

Algorithm 1 The basic summary of an MLNN algorithm using backpropagation

Initialisation
    Initialise all weights to small random values
Training
while learning **do**
    for each input vector **do**
        Forwards Phase
            Compute the activation of each neuron in the hidden layer(s)
            Work through the network and calculate the output activations
        Backwards Phase
            Compute the error at the output
            Compute the error in the hidden layers
            Update the output layer weights
            Update the hidden layer weights
    end for
    Randomise the order of the input vectors
end while
Recall
    Use the Forwards phase described above

2.2.1.3 Applications in Price Prediction

Artificial neural networks (ANNs) are likely the most commonly studied approach to predicting financial time series. They have been used for over two decades for this purpose. ANNs were first applied to stock prediction in 1988 when a feed forward network was used to analyse daily stock returns of IBM (White and Diego, 1988). Since then, a plethora of research has attempted to find predictive rules for US (Refenes et al., 1995; Enke and Thawornwong, 2005), Japanese (Kamijo and Tanigawa, 1990)

[3] In fact, a solution to this problem was already known by Bryson et al. (1979) and Werbos (1974) but they were not aware that this was a problem in neural networks

and many other stock prices. Neural network prediction has by no means been limited to stock price prediction; predicting stock indices (Chenoweth et al., 1995; Kulkarni, 1996; Fernández-Rodríguez, 2000), bonds (Dunis and Morrison, 2004), FX (Dunis et al., 2011; Sermpinis et al., 2012), futures (Trippi and DeSieno, 1992; Kim, 2004; Witkowska and Marcinkiewicz, 2005) and options (Meissner and Kawano, 2001; Mitra, 2006) have also been popular areas of research.

A particularly prominent early study of forecasting price returns with neural networks was performed by Kimoto et al. (1990). Several technical and economic indicators were used as inputs to a modular neural network prediction system. Using an adaptation of backpropagation termed 'supplementary learning' and a stand feed forward neural network with one hidden layer, predictions were found accurate enough to generate consistent profit.

Chenoweth et al. (1995) were among the first to use technical indicators as inputs to an artificial neural network (ANN). They used two distinct ANNs that were trained with either upward or downward trending data of the S&P 500 index and suggested that their prediction system could yield an annual return of approximately 15%.

Throughout the early nineteen nineties, neural network applications in finance for such tasks as pattern recognition, classification, and time series forecasting dramatically increased. In a noteworthy reflection of the popularity of neural network prediction at the time, Kaastra and Boyd (1996) provided an eight-step procedure to design a neural network forecasting models including a discussion of the tradeoffs faced in parameter selection, some common pitfalls, and points of disagreement among practitioners.

Of course, ANNs may be combined with other learning techniques. Lebaron (1998) applied bootstrapping to capture arbitrage opportunities in the foreign exchange market, and then used an ANN where its network architecture was determined through an evolutionary process. The inputs for LeBaron's network were based on technical indicators - they were: two price/moving average ratios (15 and 5 days); a lag of the interest adjusted return series; a local average of squared returns (10 days) as a volatility estimate; the interest differential at each time step; and the interest rate differential less its 30 week moving average. It was found that the combination of methods used clearly dominated some simpler forecasting methods, and random walk guessing.

In another study combining ANNs and evolutionary learning, Kuo et al. (2001) developed a genetic algorithm based fuzzy neural network to generate fuzzy inference rules combining qualitative effects (such as social and political) and technical indicators. Running simulations on the Taiwan stock market, they found that the neural network considering both the quantitative and qualitative factors outperformed the neural network considering only the quantitative factors both in the identification of buying-selling points and buying-selling performance.

More recently, Menezes and Nikolaev (2006) devised a neural network architecture termed Polynomial Genetic Programming based on the polynomial neural network first developed by Ivakhnenko (1971). The model used genetic algorithm to estimate ANN parameters including starting polynomials and weight estimation demonstrating superior results compared with standard multi-layer perceptrons over a number of timescales.

Guresen et al. (2011) performed a particular thorough comparison of ANN methodologies including: multi-layer perceptrons, dynamic artificial neural networks, and hybrid neural networks which use generalized autoregressive conditional heteroscedasticity (GARCH) to extract new input variables. Interestingly, using mean square error (MSE) and mean absolute deviate (MAD) to measure performance on daily data from the NASDAQ stock index, they found that classical multi-layer perceptrons significantly outperformed dynamic nets and GARCH-hybrids.

Wang et al. (2011) proposed a novel method for forecasting stock prices using a wavelet de-noising-based backpropagation neural network. Using monthly closing price data from the Shanghai Composite Index over a period from 1993 to 2009, they show significantly improved performance compared to a standard multi-layer perceptron. They attribute the improved performance to de-noising and preprocessing using wavelet transforms.

Using daily market prices and technical indicators as inputs, Ticknor (2013) used a novel method of Bayesian regularized artificial neural networks to predict the one day future closing price of individual stocks. In experiments using daily data on Microsoft and Goldman Sachs stock, they found that their model performs as well as more complex models but without the need for preprocessing of data, seasonality testing, or cycle analysis. They attribute this to the assignment of probabilistic network weights, allowing the network to automatically penalize complex models and reduce the potential for overfitting and overtraining.

2.2.1.4 A Note on Deep Learning

Continuing the theme of biologically inspired learning, it is thought that a great many processes in the human brain involve many layers of processing with each level learning features at a great level of abstraction (Hubel and Wiesel, 1965). In the visual cortex, for example, it is thought that the brain first identifies edges, followed by patches, then surfaces and finally objects (Serre et al., 2005; Kandel et al., 2000). This observation, along with the discovery by Hinton (2002) of a method for training with deep networks by training one layer at a time in unsupervised mannor, has spawned a resurgence in neural network learning over the last few years (Bengio, 2009).

Recently, there has been an explosion of interest in deep learning in both academia and industry, with Google's acquisition on London-based Deepmind and IBM's acquisition

of AlchemyAPI standing as telling examples. The peaked interest has been attributed to considerable progress in training deep networks due to: improved optimization techniques, e.g. Martens (2010); innovation in learning algorithms (Bengio, 2009); the adoption of GPU computing (Cirean et al., 2010); improved techniques for weight initiation (Bengio, 2012); and a technique termed *generative pre-training* which initialises the parameters of a network through with unsupervised learning (Glorot and Bengio, 2010).

Deep learning has demonstrated fantastic results in fields such as: feature extraction (Lee et al., 2009; Chen et al., 2010), autoencoding (Krizhevsky and Hinton, 2011), handwriting recognition (Hinton et al., 2006), speak recognition (Hinton et al., 2012), sentiment analysis (Glorot et al., 2011), and even playing the Atari (Mnih et al., 2013). However, we will not document nor explore the application of deep learning to forecasting market return as our goal is to create a simple, transparent system for exploring the underlying drivers of market dynamic. Though powerful, deep learning techniques are far too complex and opaque for our needs.

Although the studies discussed in this section indicate that neural networks are able to outperform relevant benchmarks, the ANNs approach has a number of significant problems:

- Neural networks are inherently unstable. That is, small changes in training data can causes substantial changes in the model generated and thus its ability to generalise (Cunningham et al., 2000).
- ANNs are a bit of a black box when it comes to interpretation. Often neural networks are too complex to be analysed mathematically and it can be difficult to relate network weights to physical parameters, should this be a requirement.
- It has been shown that in order to overcome their tendency to overfit, a particularly large volume of training data is required to capture the underlying structure of the data(Hippert et al., 2001).
- Large, effective neural nets require intense compute power and as such, the use of distributed computing using GPUs is often required to train networks in a reasonable timescale (Krizhevsky et al., 2012).

These drawbacks suggest that neural networks may not be the ideal model for exploring market dynamics. Below, we discuss support vector machines (SVMs), which use the concept of a margin to attempt to overcome problems with overfitting.

2.2.2 Support Vector Machines

The support vector machine was first described by Boser et al. (1992). While SVMs provide similar prediction accuracy to neural networks, thanks to their convex objective

function, they are much easier to train. The birth of the SVM gave rise to a decade of intense study of kernel methods.

2.2.2.1 The Margin

Like multi layer perceptrons, SVMs make use of the insight that data can be more easily separated when it is mapped into higher dimensions. In addition, SVMs leverage the concept of *optimal separation*. Specifically, and SVM classifier aims to maximise the size of the largest region, known as the margin, that separates classes without their being any points inside. An example for a two dimensional dataset is given in Figure 2.4, in three dimensions the margin is a cylindrical and in greater dimensions is a hypercylinder.

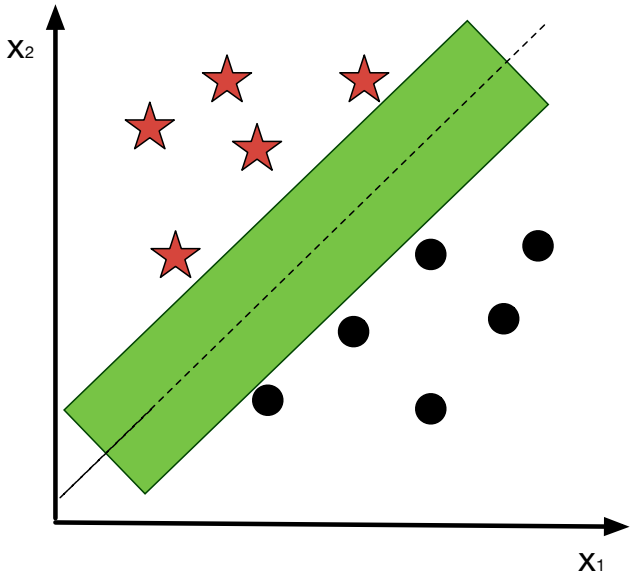


FIGURE 2.4: The margin - the largest region that can separate the classes without points falling inside.

The datapoints that sit closest to the separating line are known as the *support vectors*. Given that the most effective classifier is the one that goes through the center of the margin, we can make two arguments: firstly, we should aim to maximise the margin, and second we should be most interesting the the support vectors as these are most likely to be incorrectly classified. More formally, SVMs adopts a structural risk minimization principle of function estimation by minimizing an upper bound of the generalization error (Vapnik, 1999).

The mathematical formalisation of the maximisation of the margin with soft constrains and the subsequent optimisation problem are beyond the scope of the review. Instead, let us discuss the next important concept of SVMs.

2.2.2.2 Kernels

Now, although the concept of a margin may enable us to find a decision boundary that is superior to that identified by a perceptron, we still need a method for overcoming the dependance on linear separability. SVMs use a basis function to map the data into higher dimensions such that they can be separated by a hyperplane. Thanks to the *kernel trick* SVMs avoid the computationally expensive process of computing the dot products of all the extended basis vectors and instead simply compute the dot product of the orignial vectors. These kernels are the basis for why support vector machines work. Although they transform the data into a higher dimensional space, the datapoints only appear inside those inner products and so there's no need to actually perform any computation in the higher dimensional space.

Choosing an appropriate kernel and the parameters for it is a difficult problem that ideally requires in-depth domain knowledge to create the best transform. Though there are techniques for identifying a kernel (Blumer et al., 1989), many researchers tend to identify a kernel function through experimentation. Although, in theory, any symmetric function that is positive definite can be used as a kernel, there are three particularly commonly used basis functions that have mathematically friendly kernels that correspond to them:

Polynomials of degree s in the elements x_k of the input vector with the kernel:

$$\mathbf{K}(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^s \tag{2.3}$$

Radial Basis Function expansions of the $x_k s$ with parameter σ and kernel:

$$\mathbf{K}(\mathbf{x}, \mathbf{y}) = e^{\frac{-(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}} \tag{2.4}$$

Sigmoid Functions of the $x_k s$ with parameters κ and δ, and kernel:

$$\mathbf{K}(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x} \cdot \mathbf{y} - \delta) \tag{2.5}$$

2.2.2.3 Applications in Price Prediction

Since their inception, SVMs have been used in many studies that have used closing prices as inputs for forecasting prices and shown them to outperform ANNs (Huang et al., 2005; Chen and Shih, 2006).

Tay and Cao (2001) used lagged closing prices as well as moving averages to predict the next day's closing price for five futures contracts on the Chicago Mercantile Market. In comparison with neural networks they found that the SVMs produced better forecasts.

In 2002 they modified their original SVM model to by using a C-ascending SVM (Tay and Cao, 2002). This SVM variant contains a modified regularised risk function, whereby the recent errors are penalised more heavily than the distant ones. This procedure was based on the idea that, in the non-stationary financial time series, the dependency between input variables and output variable gradually changes over the time. Specifically, they thought that recent past data could provide more important information than the distant past data. They found that C-ascending SVMs consistently forecast better than the standard SVMs and that they also used fewer support vectors than those of the standard SVMs, resulting in a sparser representation of solution.

Kim (2003) used 12 technical indicators, including ROC, Williams' %R, A/D oscillator, stochastic D%, stochastic K%, stochastic slow D%, momentum, disparity5, disparity10, OSCP, CCI and RSI, as inputs for an SVM to predict the direction of change of the daily Korea Composite Stock Price Index (KOSPI). It was found that SVMs outperformed back-propagation neural networks and case based reasoning. Kim suggested that the results could be attributed to the fact that SVMs implement the structural risk minimisation principle, which leads to better generalisation than the two alternative techniques. It was also noted that the SVM was highly sensitive to the upper bound and kernel parameters and that finding optimal parameter values is essential.

In a comparative study, Kumar and Thenmozhi (2005) analysed the performance of SVMs and random forests in predicting daily price fluctuations in the S&P 500. Using the sam technical indicators as Kim (2003), they found the SVM to outperform both neural networks and random forests. Further comparative experiments investigated the performance of autoregressive integrated moving average (ARIMA), random forests, ANNs and SVMs in a live trading experiment finding that, in the short term, the SVM model outperformed all other models in the study.

Kara et al. (2011) compare the ability of SVM and ANN classification techniques in predicting the direction of daily movement in the Istanbul Stock Exchange (ISE) National 100 Index. Using just 10 technical indicators as inputs, their experiments found the average prediction accuracy of the SVM model to be only 71.52% compared the the neural networks performance accuracy of 75.74%.

More recently, Kao et al. (2013) use a novel feature extraction method named nonlinear independent component analysis to extract features that serve as inputs to a SVM for forecasting the price returns of two of the major Asian stock markets: the Shanghai Stock Exchange Composite (SSEC) and Nikkei 225 stock indexes. They compare their novel approach with more traditional preprocessing methods including: linear ICA and principal component analysis finding their nonlinear independent component analysis to improve prediction accuracy all other methodologies studied.

Kazem et al. (2013) proposed a predictive model based on SVMs, chaotic mapping and the firefly algorithm for predicting changes in stock prices. Their model consisted of

three stages. In the first, phase space dynamics are constructed using a delay coordinate embedding method, from this, a chaotic firefly algorithm is used to optimize the hyperparamters of an SVM. Finally, the tuned SVM is used to forecast the price returns of a number of stocks that trade on the NASDAQ. The model was compared with ANNs, adaptive neuro-fuzzy inference systems, genetic algorithm-based SVMs, and firefly-based SVMs and found to outperform all models in term of mean absolute percentage error (MAPE) and MSE.

Though they have clearly been demonstrated to be powerful, SVMs may not be the perfect approach for improving our understanding. Unfortunately, the complex data transformations that occur in training an SVM and multi-dimensional boundary planes that result are very difficult to interpret. This is why they are often referred to as a black box. This is in contrast to general linear models and decision trees which are very easy to interpret. Also, SVMs are particularly sensitive to the scale of the features and, often, with datasets that contain very heterogenous features, it is much easier to use a decision tree based model where there is no concern for feature scaling. Next, we discuss some less opaque techniques for modelling price returns.

2.2.3 Evolutionary Learning

In a recurring theme, computer scientists treated evolution in much the same way that they did neuroscience, by cherrypicking some interesting ideas and filling in the blanks with computation techniques so as to create useful learning methodologies. Evolution, as a process, may be viewed as a search problem, with organisms competing with one another to further their survival. Evolution occurs in environments that bias 'fitter' organisms. That is, organisms that are better suited to an environment and are able to live longer are moer attractive and have a better chance of producing offspring.

The genetic algorithm (GA), first developed by Holland (1975), models the process that gives rise to evolution. Specifically, it models: mutation, where individual characteristics have some probability of changing slightly through time; and sexual reproduction, where parents can mix genetic information to produce offspring. In most biological instances, each parent has a 50% chance of passing a given chromosome to their offspring and so only one allele (chromosomal variation) will be inherited (Agrawal, 2001). Coupled with random mutations that occur during the copying process Kondrashov (1988), offspring share traits with their parents while also displaying new phenotypes.

In order to create a computational approximation to the process of evolution, a number of conditions must be met. First, solutions to problems must be abe to be represented in the form of a chromosome. Second, for each potential solution, it must be possible to compute its *fitness*. Next, a method for selecting *parent* solutions must be described. Finally, a process for introducing variation into chromosomes is required. Solutions to

these requirements are discussed in the sections below.

2.2.3.1 Chromosomal Representation

Before anything, individuals must be represented in analogy to a chromosome. Traditionals claim that true GAs must use a string representation with each element of the string (the chromosome) represented by a letter from some alphabet yet, in practice, a variety of string representations may be used (Wright, 1991). Examples of applications and their associated chromosomal representations include:

- The travelling salesman problem (Goldberg and Lingle, 1985; Braun, 1991; Larrañaga et al., 1999). Here, cities are represented by letters and routes are easily represented as strings of those letters, e.g. BHGADCEF.
- Finding solutions to a function optimisation problem (Deb et al., 2000; Houck et al., 1995; Goldberg and Richardson, 1987). If the function takes binary arguments, string representations are trivial, e.g. 00101110101000010111.
- Learning weights in a neural network (Leung et al., 2003; Whitley et al., 1990). Here, real variables make up the chromosome. e.g. 3 fully connected nodes = 9 weights (with self connections) - [0.01, 1.24, 3.42, -0.91, -2.03, 1.71,-0.08, 4.22, 1.11]
- Computer programs (Koza, 1992; Miller and Thomson, 2000). This is known as genetic programming and involves representing computer programs as trees. The LISP language leds itself well to this field, e.g. (div (+ 2 a) (sin (* - a 2) b)))

Not matter the application, the goal is to create a set of random chromosomal representations to act as an initial population. Note, these representations do not, in themselves, tell us whether the solution is good or even feasible. To determine feasibility, it is necessary to provide a way to quantify how well each individual solves the problem, known as the *fitness* of the individual.

2.2.3.2 Fitness Functions

Put simply, the fitness function takes an individual (genome, string, etc.) as an input and outputs a value that represents the goodness of fit for that individual. This is the only part of a genetic algorithm that is application specific. For a fitness function, the best possible individual should have the highest fitness and fitness should diminish as individuals do less well as solutions to a problem. For the travelling salesman problem, the fitness may simply be the length of the tour; while for a neural network, the fitness may be the prediction accuracy of the model on a given problem. In general, fitness should always be a positive function (Whitley, 1994).

2.2.3.3 Selecting Parents

For each generation individuals are selected to generate new offspring. The idea is to exploit the current population by selecting strings that are fit compared with their peers. Specifically, individuals are chosen with a probability than is proportional to their fitness. There are a number of methods commonly used to select parents:

Truncation Selection - Pick top fraction f of the population and discard the rest. If, for example, $f = 0.2$, the fittest 20% of the population are put into the mating pool and chosen with equal probability. Note, this limits the exploration potential of a GA and biases the algorithm towards exploitation (Alba and Dorronsoro, 2005).

Rank Based Selection - Individuals are chosen with a probability proportional to their rank.

Fitness Proportionate Selection - Here individuals are chosen with a probability that is proportional to their fitness. Formally:

$$p_\alpha = \frac{F_\alpha}{\sum_\beta F_\beta} \tag{2.6}$$

where F_α is the fitness of individual α.

Boltzmann Selection - If fitness is not always positive then Boltzmann selection may be used to make them so:

$$p_\alpha = \frac{e^{sF_\alpha}}{\sum_\beta F_\beta} \tag{2.7}$$

where s is a parameter that controls the selection strength (de la Maza and Tidor, 1993).

Tournament Selection - Picking k individuals and returning the individual with the greatest fitness. Though tournament selection can ensure that good solutions are not lost, it has been shown to encourage premature convergence Miller and Goldberg (1995).

Once a method is chosen for selecting parents, all that is left is to identify a method for producing offspring.

2.2.3.4 Producing Offspring

After selecting the parents for creating the next generation, a process is required for combining their chromosomes to generate offspring. This is the genetics part of the algorithm and two more concepts are borrowed from biology: crossover and mutation. These concepts are discussed below.

Crossover

With pairs of parents selected, we must generate offspring using genetic material (e.g. individual letters, bits, nodes, weights or numbers) from each parent in a pair. One very common method for achieving this is known as *single point crossover* (Deb and Agrawal, 1995). Here, a point in the chromosome is chosen randomly and all alleles before this crossover point come from parent one and all alleles after come from parent two. Of course, this only creates one offspring and the other offspring receives all the genetic material before the crossover point from parent two and after from parent one. A logical extension of the crossover process is *multi-point crossover* where the chromosome is split at a greater number of points (De Jong and Spears, 1992). In the extreme, we find *uniform crossover* where each allele is randomly selected from one of the parents.

Crossover is the element of the GA that performs global explorations. The hope is that it may take good traits from each parent solution and produce an offspring with greater numbers of good traits that were previously not seen in one individual. Now, we can also expect bad traits to come together but individuals with lower fitness will be less likely to be selected for reproduction in the next generation. Thus the GA aims to combine and amplify traits that increase fitness.

Mutation

The mutation operation is the portion of the GA that performs local search Srinivas and Patnaik (1994). It enables the exploitation of good solutions while exploring local areas of the fitness landscape. For mutation, the value of each allele is altered with a small probability, p, where often $p = \frac{1}{L}$ where L is the number of alleles in the genome. In genomes that are represented as an alphabet of binary string, mutation is a simple bit flip, while with real numbers the mutation operation tends to either add or subtract a random number from the allele. This form of local search trades off the refinement of good solutions with the possibility of disrupting them.

The genetic operators described above are arguable the most powerful elements of the genetic algorithm and form a fundamental part of the building block hypotheis that describes why GAs work (Fogel, 1994). Generally, genetic algorithms work well on problems where good solutions intuitively come from putting together lots of smaller solutions such different sections of the genome assemble a separate building block, with crossover assembling building blocks to produce a final solution. As such, GAs have been successfully applied across a number of domains including: scheduling (Gonçalves et al., 2005; Cheng et al., 1996), multi-objective optimisation (Horn et al., 1994; Deb et al., 2000), hardware design (Yang et al., 2008; Robinson et al., 2002) and even the optimisation of poker strategies (Noble, 2002; Barone and While, 1999).

2.2.3.5 Applications in Price Prediction

When applied to price prediction, GAs are generally used to generate evolving trading rules. The rules tend to be represented as binary trees where the leaves are technical indicators and the non-leaves are Boolean functions. Together they represent simple decision functions. Shin et al. (1998) followed this approach and used genetic algorithms to mine trading rules from 9 technical indicators for KOSPI 200 futures. The GA used the technical indicators to create a set of five rules for each trading day that determined whether a buy or sell signal was produced. They reported substantial profits generated in simulated experiments even while the underlying index hit a 58% downturn.

Mahfoud and Mani (1996) developed a novel GA based system for predicting quarterly stock returns based on 15 technical and fundamental attributes. Their algorithms differed from traditional GAs, which perform optimisation, in that it performed inductive machine learning. They compared their system with neural networks and found GAs to outperform neural networks (5.47% relative return versus 4.40%). In addition, they explored a hybrid GA-ANN system and found a 21% improvement over GA only system.

In an experiment using 67 years of daily data from the S&P 500 index, Allen and Karjalainen (1999) attempted to use a GA to find profitable technical trading rules. They found that, after transaction costs, the rules generated did not create any excess returns over a simple buy-and-hold strategy in the out-of-sample test periods. It was stated, however, that the rules learned by GAs could also be tested on liquid markets with low transaction costs, e.g. financial futures, commodities, and foreign exchange markets. They suggested that the methodology could be developed further as the genetic algorithm used was a relatively simple one.

Dempster et al. (2001) compared four methods for foreign exchange trading (reinforcement learning, GAs, Markov chain linear programming, and a simple heuristic) and found that a combination of technical indicators using a GA leads to better profit based performance than using only individual indicators for the foreign exchange market. They found that, although all of the methods were able to generate significant in-sample and out-of-sample profits when transaction costs were zero, none of the methods produced significant profits at realistic transaction costs.

A hybrid model for predicting stock indices was proposed by Asadi et al. (2012). The model proposed is a combination of various preprocessing techniques combined with a genetic algorithm and the LevenbergMarquardt algorithm for learning the weights of a feed forward neural network. Through training and testing on a number of global stock indices, they find their approach to maintain steady returns through periods of high volatility and to yield consistently strong prediction accuracy. The authors attribute the successof their model to the ability of the GA to traverse the complex space parameter space and of the neural network to model the complex relationships between the technical

indicators and the index.

Bodas-Sagi et al. (2013) also address the optimisation of parameter values for technical indicators. They describe a novel method of Multi-Objective Evolutionary Algorithms to identifying the optimal parameter settings for a collection of technical indicators. The results showed that their technique was able to discover parameters settings that weremore profitable than those suggested as standard in the literature. Furthermore, they showed that evolution learning techniques lend themselves well to parallelisation and that considerable runtime speed increases can be achieved.

In a move away from optimising technical indicators, Sun et al. (2013) proposed an evolutionary morphological-rank-linear method for adjusting the time phase distortions that can appear when modeling financial time series. The method presented is a hybrid approach whereby a modified genetic algorithm is used to identify the optimal parameters of the morphological-rank-linear filter. In an experiment comparing the forecasting power of their technique to multi-layer perceptrons and time-delay added evolutionary forecasting, they find their method to give improved predictions across a number of stocks and markets.

2.2.4 Predicting with Ensembles

Ensemble learning is built upon the idea that predicting with lots of models that each get slightly different results on a dataset and combining their predictions, produces significantly better results than any one model on its own.

With ensemble learning, there are three main questions to consider: what kind of models should we use, how can we ensure that they learn different things, and how should we combine their predictions? Ensuring that the underlying models learn different things is the primary difference between the various types of ensemble algorithms.

In recent years, the ensemble approach has been applied with great success in expert learning systems such as decision trees (Breiman, 1996) and ANNs (Hansen and Salamon, 1990). Ensembles of expert systems have already been successfully applied in many areas such as face recognition (Gutta and Wechsler, 1996), character recognition (Mao, 1998) and medical diagnosis (Zhou et al., 2002) among others.

2.2.4.1 Boosting

One of the more well-studied ensemble techniques is known as boosting. Boosting came about through the desire to turn a collection of weak classifiers into a strong ensemble committee. It is a method for improving the performance of any traditional learning algorithm. Schapire (1990) first proposed boosting in the computational learning theory

literature before Freund and Schapire (1997) went on to solve many of the practical difficulties of earlier boosting algorithms with the creation of the general discriminative learning algorithm Adaboost.

Adaboost

Adaboost works by repeatedly applying a simple learning algorithm, termed the weak or base learner, to various weightings of the same training set. The elementary form of Adaboost is intended for binary classification problems in which a training set consists of paired data $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, where x_i represents the features of the training data and $y_i \in \{-1, +1\}$ corresponds to the binary label to be predicted. A *weighting* of the training examples represents the assignment of a non-negative real value w_i to each example (x_i, y_i).

On iteration t of boosting, the weak learner is applied to the training set with a set of weights $\mathcal{W}_1^t, \mathcal{W}_2^t, \ldots, \mathcal{W}_m^t$ and generates a prediction rule h_t that maps x to $\{0, 1\}$. It is required that the weak learner for $h_t(x)$ has a *small* but significant correlation with the labels y using the current weighting. Once a weak prediction rule is generated, the weights are then changed such that the predictions $h_t(x)$ and the example labels y are decorrelated. The process is then repeated as the weak learner is applied to the same training examples with the new weights. Once the process has been repeated several times, all of the prediction rules $h_{1-t}(x)$ are combined into a single *strong* rule using a weighted majority vote. It can be shown that so long as the weak prediction rules generated at each iteration are all somewhat correlated with the labels, the the strong rule will have very high correlation with the labels.

One way to view the entire process is to think of a function $F(x)$ that approximates the mapping of an feature space X to the target variable Y and is repeatedly improved by adding to it small corrections from the new weak prediction functions. Continuing with this notation, we describe Adaboost in Algorithm 2. The strong prediction rule that Adaboost generates is defined as $sign(F(x))$.

Algorithm 2 The general Adaboost algorithm (Freund and Schapire, 1997). x_i corresponds to the input features of instance i; y_i is the binary label to be predicted; $\mathcal{W}_i^t$ is the weight at time t of instance i; and $sign(F_t(x))$ is the prediction at time t.

$F_0(x) \equiv 0$
for $t = 1 \rightarrow T$ **do**
  $\mathcal{W}_i^t = e^{-y_i F_{t-1}(x_i)}$
  Use weak learner to generate h_t
  $\alpha_t = \frac{1}{2} \ln \frac{\sum_{i:h_t(x_i)=1, y_i=1} \mathcal{W}_i^t}{\sum_{i:h_t(x_i)=1, y_i=-1} \mathcal{W}_i^t}$
  $F_{t+1} = F_t + \alpha h_t$
end for

A particularly interesting feature of Adaboost is that the test error[4] of the strong rule tends to continue to fall after the training error[5] sinks to zero. Schapire et al. (1998) relates this to the concept of a margin, that is: while $y \cdot F(x) \geq 0$ denotes a correct prediction[6], $yF(x) > b > 0$ represents a confident prediction and the confidence will increase in line with b.

Logitboost

Friedman et al. (2000), shortly followed by Collins et al. (2000), described a modification of Adaboost, called Logitboost. Logitboost has an alternative method for calculating the weights that means it can be interpreted as a step-wise logistic regression algorithm; it is described in Algorithm 3.

Algorithm 3 The general Logitboost algorithm (Friedman et al., 2000). x_i corresponds to the input features of instance i; y_i is the binary label to be predicted; $\mathcal{W}_i^t$ is the weight at time t of instance i; and $sign(F_t(x))$ is the prediction at time t.

$F_0(x) \equiv 0$
for $t = 1 \rightarrow T$ **do**
  $\mathcal{W}_i^t = \frac{1}{1+e^{-y_i F_{t-1}(x_i)}}$
  Use weak learner to generate h_t
  $\alpha_t = \frac{1}{2} \ln \frac{\sum_{i:h_t(x_i)=1, y_i=1} \mathcal{W}_i^t}{\sum_{i:h_t(x_i)=1, y_i=-1} \mathcal{W}_i^t}$
  $F_{t+1} = F_t + \alpha h_t$
end for

2.2.4.2 Random Forests

The random forest (RF) is a nonparametric and nonlinear classification and regression algorithm first proposed by Ho (1995) and further developed by Breiman (2001). As the name suggests, an RF is an ensemble of many classification or regression trees designed to produce accurate predictions that do not overfit the data (Breiman, 2001). Regression trees use a tree structure to recursively partition a feature-space until the subsets of that feature-space are manageable enough to fit simple models to. The tree model therefore has two parts: the recursive partition and a simple model for each cell of the partition. Each of the terminal nodes (leaves) of a tree represents a small subset of the feature space, and is attached to a simple model which applies only to that subset. The cell to which a datapoint belongs is identified by starting at the root node of the tree, and answering a sequence of questions about the feature values. An example of the structure of a regression tree is given in Figure 2.5.

[4]The test error represents the fraction of mistakes made on unseen data examples.
[5]The training error is the percentage of mistakes made on the training data.
[6]remember that we are predicting the binary variable $y_i \in \{-1, +1\}$, so a correct prediction is made when $y = 1 \wedge F(x) \geq 0$

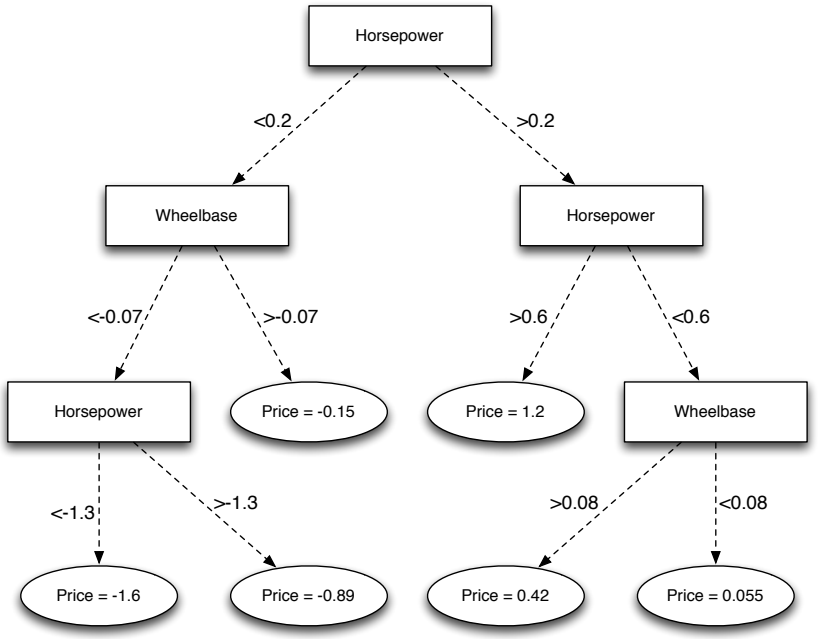


FIGURE 2.5: Example of a regression tree for predicting price of cars built in 1993. Both the target and the features have been standardised to have zero mean and unit variance.

In training an RF, bootstrap samples are drawn from the training data to construct multiple regression trees, where each tree is grown using a randomised subset of features. Specifically, we produce forests of regression trees using the procedure below, where the training set consists of feature vectors, $\mathbf{x}$, and targets, y, and is defined as $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$. The aim of the RF is to find a function $F : X \to Y$, where X is the feature space and Y is the output space.

In growing each tree without pruning, and selecting the best split among a random subset at each node, RFs maintain prediction strength while inducing diversity among trees. Furthermore, the random selection of features reduces correlation among the unpruned trees and keeps overall model bias low. With an ensemble of such trees, variance is also reduced.

Now, for any new observation, the RF produces an overall prediction by taking the average of the predictions of the individual trees in the forest. A formalisation of the RF algorithm is given in Algorithm 4.

Compared with other prediction algorithms, RFs have been shown to be less prone to overfitting (Breiman, 2001) and as such have proved successful in a number of domains including image classification (Bosch et al., 2007), ecological prediction (Prasad et al., 2006) and microarray data classification (Díaz-Uriarte and Alvarez De Andrés, 2006).

Algorithm 4 The random forest algorithm.

Training
for $\theta = 1$ to Θ **do**
  Select n observations from D with replacement to form a bootstrap sample.
  Grow a regression tree T_θ on the bootstrap sample by recursively following:
  for each terminal node **do**
    Randomly select $m << M$ variables.
    Pick the best split among the m as measured by the sum of squares eror.
    Split the node into two child nodes according to the best split.
  end for
end for
Output the ensemble of trees.
Recall
The prediction, $S_i = \frac{1}{\Theta}\sum_{\theta=1}^{\Theta} t_\theta(x)$

2.2.4.3 Applications in Price Prediction

In relation to financial markets, an ensemble of ANNs was investigated by Abdullah and Ganapathy (2000) as a method for classifying trends of the Kuala Lumpur Stock Exchange composite index. They compared three relatively simplistic methods of combining the classifiers - max, average and median - finding that median produced the higher classification score and mean return per trade. Chen et al. (2007) used a similar approach of combining the outputs of an ensemble of ANNs but instead used particle swarm optimisation to determine the weight that each predictor was given.

In a similar vein, Chun and Park (2005) used an ensemble of case-based-reasoning models to predict the daily movement of the KOSPI. They found that choosing the expert with the best historical performance gave the best out-of-sample results, significantly outperforming a random walk model with $p < 0.01$.

In a particularly extensive study, Tsai et al. (2011) used classifier ensembles to forecast stock returns. Specifically, they compared the hybrid methods of majority voting and bagging finding that, although multiple classifiers outperformed single ones in terms of prediction accuracy and returns on investment, there was no significant difference between majority voting and bagging. Furthermore, performance using 'homogeneous' ensembles (e.g. an ensemble of neural networks) and 'heterogeneous' ensembles (e.g. an ensemble of neural networks, decision trees and logistic regression) was analysed to find that heterogeneous ensembles offered slightly better performance than homogeneous ones.

With similar goals in mind, Creamer and Freund (2010) used an alternating decision tree (ADT) learning algorithm, which was implemented with Logitboost, to generate buy and sell signals for a multitude of stocks in the S&P 500 index. Additionally, they implemented an on-line learning layer to combine the output of several ADTs and suggests a short or long position. They found that their algorithm was able to generate

consistently positive returns and the the online layer enabled the algorithm to constantly adapt in line with the market.

Rodríguez and Sosvilla-Rivero (2006) used Adaboost with decision tree learning algorithms to find direction-of-change patterns for the S&P 500 index with daily prices from 1962 to 2004. They found that periods characterised by high first-order serial correlation in stock returns allowed both in and out-of-sample direction-of-change predictability. However, it was noted that functions induced in periods with a lack of autocorrelation in stock returns were able to obtain in-sample predictability but failed to detect out-of-sample predictability.

Creamer and Freund (2010) used an alternating decision tree (ADT) learning algorithm, which was implemented with Logitboost, to generate buy and sell signals for a multitude of stocks in the S&P 500 index with high frequency data from 2003-2005. Their algorithm predicts by taking an average of the predictions of all the ADTs weighted by their training error. Their formula for the exponential weights is derived from a weighted majority algorithm introduced by Littlestone and Warmuth (1989). Logitboost enables their algorithm to select the best combination of rules derived from well-known technical analysis indicators and is used to select the best parameters of the technical indicators themselves. Additionally, they implemented an on-line learning layer to combine the output of several ADTs and suggests a short or long position. They found that their algorithm was able to generate abnormal returns during the test period and their experiments showed that the boosting approach was able to improve the predictive capacity when indicators were aggregated as a single predictor. Also, the combination of indicators of different stocks was able to to reduce the need for computational power, while still maintaining adequate predictive capacity.

Creamer and Freund (2004) used random forests to successfully predict performance of companies and to measure corporate governance risk in Latin American banks. In their study, the performance of RFs was compared to logistic regression and Adaboost, finding that RFs consistently produced superior results. The merits of RFs in financial prediction were also demonstrated by Lariviere and Vandenpoel (2005) who demonstrated that random forest regression could be used for exploring both customer retention and profitability. They analysed a sample of 100,000 customers using data from a large European financial services company finding that RF techniques provided better prediction results for the validation and test samples compared to linear regression and logistic regression models.

More recently, there has been a surge in interest in he use of RFs for stock market prediction. Maragoudakis and Serpanos (2010) used a method called Markov Blanket random forest to make predictions on the direction of stock markets. They report their proposed strategy to outperform a simple buy-and-hold investment strategy by an average of 12.5% to 26% for the initial period and from 16% to 48% for the remainder,

as well as outperforming linear regression, SVMs and ANNs. With similar goals in mind, Qin et al. (2013) used gradient boosted random forests to make predictions on the direction of the Singapore Exchange. Using boosting to weight the individual trees of the forest and a forgetting factor to address market changes, their empirical results showed that their proposed methods were able to generate excess returns compared to a buy-and-hold strategy. Also, Zbikowski and Grzegorzewski (2013) used a novel online-adaptation method to allow RFs to adapt to non-stationary financial time series, while Xu et al. (2013) demonstrated the RF algorithmss ability to select features for trend prediction in stock prices.

The impressive out-of-sample results reported in these studies form the basis of our decision to use random forests as a baseline expert predictor as it is clear that RFs are the most appropriate approach for applications with large, noisy datasets. Although many of the trading models mentioned above are shown to produce an acceptable risk-return behaviour on recent historical data, there is no guarantee that such systems will continue generating the same returns through all market phases in the future without higher level adaptation. Algorithms that are unable to adapt to the changing market conditions will not succeed in the long run. To this end, we propose a novel approach that sees the online training of random forests to produce an ensemble of experts that is able to produce accurate predictions in non stationary financial time series data.

2.2.5 Precautions for Data Mining

Zemke (2002) provides an excellent review of the potential pitfalls to consider when using financial data for prediction. This report highlights the importance of rigorous preprocessing and, in particular, the need for: sufficient good quality data, outlier detection; and adjustments for missing data. Another import issue is that of overfitting. In essence this refers to fitting the model so closely to the training data that the out of sample performance is reduced dramatically.

After models are fitted, relationships must be checked to ensure they conform to common/economic thought. Particularly, high correlation and R^2 do not imply a causal relationship. Leinweber (2007), demonstrates this danger by fitting the S&P 500 returns, butter production in Bangladesh and the sheep population of both countries with an R^2 of 0.99.

Throughout this section we've seen a number of models that have demonstrated solid prediction accuracy across a number of financial markets. However, let us take a step back and consider our motives for selecting a model: transparency. We aim to build a predictive model so as to greater improve our understanding of the drivers of price dynamics. As such, many of the more 'black box' techniques (e.g. neural networks and support vector machines) will make it more challenging to deconstruct those drivers.

Before we go on to formulate a model, we first examine the statistical properties of limit order markets than will enable us to explore the validity of our models.

2.3 The Statistical Properties of Limit Order Markets

The empirical literature on LOBs is very large and several non-trivial regularities, so-called "stylised facts", have been observed across different asset classes, exchanges, levels of liquidity and markets. These stylised facts are particularly useful as indicators of the validity of a model (Buchanan, 2012). For example, Lo and MacKinlay (2001) show the persistence of volatility clustering across markets and asset classes, which disappears with a simple random walk model for the evolution of price time series, as clustered volatility suggests that large variations in price are more likely to follow other large variations.

2.3.1 Fat-tailed Distribution of Returns

Across all timescales, distributions of price returns have been found to have positive kurtosis, that is to say they are "fat-tailed" (Officer, 1972; Nelson, 1991). An understanding of positively kurtotic distributions is paramount for trading and risk management as large price movements are more likely than in commonly assumed normal distributions.

Fat tails have been observed in the returns distributions of many markets including: the American Stock Exchange by Plerou and Stanley (2008a), Euronext by Chakraborti et al. (2011), the LSE by Plerou and Stanley (2008b), NASDAQ by Gopikrishnan et al. (1998), the NYSE by Gopikrishnan et al. (1998), the Paris Bourse by Plerou and Stanley (2008a), the S&P 500 index by Cont (2001) and the Shenzhen Stock Exchange (Gu et al., 2008) but the precise form of the distribution varies with the timescale used. Gu et al. (2008) found that across various markets, the tails of the distribution at very short timescales are well-approximated by a power law with exponent $\alpha \approx 3$.

Drozdz et al. (2007) found tails to be less heavy ($\alpha > 3$) in high-frequency data for various indices from 2004 to 2006, suggesting that the specific form of the stylised facts may have evolved over time with trading behaviours and technology. Both Gopikrishnan et al. (1998) and Cont (2001) have found that at longer timescales, returns distributions become increasingly similar to the standard normal distribution.

2.3.2 Volatility Clustering

Volatility clustering refers to the long memory of absolute or square mid-price returns and means that large changes in price tend to follow other large price changes. Liu et al. (1997), Cont (2001), and Stanley et al. (2008) found this long memory phenomenon to exist on timescales of weeks and months while its existence has been documented across the NYSE by Cont (2005), Paris Bourse by Chakraborti et al. (2011), S&P 500 index

futures by Cont (2001), the Shenzhen Stock Exchange by Gu and Zhou (2009) and the USD/JPY currency pair by Cont et al. (1997).

Lillo and Farmer (2004) formalise the concept as follows. Let $\mathbf{X} = X(t_1), X(t_2), \ldots, X(t_k)$ denote a real-valued, wide-sense stationary time series. Then, we can characterise long memory using the diffusion properties of the integrated series $\mathbf{Y}$:

$$Y(l) = \sum_{i=1}^{l} X(t_i) \tag{2.8}$$

Furthermore, let

$$V(l) = Var(Y(t_i + 1), Y(t_i + 2), \ldots, Y(t_i + l)) \tag{2.9}$$

for some $i \in \{0, 1, \ldots, l\}$. Given this, in the limit $l \to \infty$, if $\mathbf{X}$ is a short-memory process, then $V(l)$ scales as $\mathcal{O}(l)$, whereas if $\mathbf{X}$ is a long-memory process, then $V(l)$ scales as $\mathcal{O}(l^{2H})$, for some $H \in (0.5, 1)$. The exponent H is known as the Hurst exponent.

In the empirical research studies outlined above, the values of the Hurst exponent varied from $H \approx 0.58$ on the Shenzhen Stock Exchange to $H \approx 0.815$ for the USD/JPY currency pair. There are a number of potential explanations for volatility clustering and Bouchaud et al. (2009) suggest the arrival of news and the splicing of large orders by traders.

2.3.3 Autocorrelation of Returns

Stanley et al. (2008) and Chakraborti et al. (2011) observed that, across a number of markets, returns series lacked significant autocorrelation, except for slightly negative autocorrelation on very short timescales. This includes: Euronext (Chakraborti et al., 2011), FX markets (Cont et al., 1997), the NYSE (Cont, 2005; At-Sahalia et al., 2011) and the S&P500 index (Gopikrishnan et al., 1999). Cont (2001) explains the absence of strong autocorrelations by proposing that, if returns were correlated, traders would use simple strategies to exploit the autocorrelation and generate profit. Such actions would, in turn, reduce the autocorrelation such that the autocorrelation would no longer remain.

Evidence suggests that the mild negative correlation found on short time-scales has disappeared more quickly in recent years, perhaps an artefact of the new financial ecosystem. Bouchaud and Potters (2003) report that from 1991 to 1995, negative autocorrelation persisted on timescales of up to 20-30 minutes but no longer for the GBP/USD currency pair. For Euronext, Chakraborti et al. (2011) found autocorrelation not to

persist in timescales over 1 minutes during 2007-2008. Moreover, Cont et al. (2013), discovered no significant autocorrelation for timescales of over 20 seconds in the NYSE during 2010.

2.3.4 Long Memory in Order Flow

The probability of observing a given type of order in the future is positively correlated with its empirical frequency in the past. In fact, analysis of the time series generated by assigning the value +1 to incoming buy orders and -1 to incoming sell orders has been shown to display long memory on the NYSE (Lillo and Farmer, 2004), the Paris Bourse (Biais et al., 1995; Bouchaud et al., 2004) and the Shenzhen Exchange (Gu and Zhou, 2009). Study of the LSE has been particularly active, with Lillo and Farmer (2004), Mike and Farmer (2008), and Bouchaud et al. (2009) reporting similar results for limit order arrivals, market order arrivals and order cancellations, while Axioglou and Skouras (2011) suggest that the long memory report by Lillo and Farmer (2004) was simply an artefact caused by market participants changing trading strategies each day.

2.3.5 Long Memory in Returns

The long memory in order flow discussed above has lead some to expect long memory in return series, yet has not been found to be the case. Studies on the Deutsche Bourse (Carbone et al., 2004), the LSE (Lillo and Farmer, 2004) and on the Paris Bourse (Bouchaud et al., 2004) have all reported Hurst exponents of around 0.5, i.e. no long memory. Bouchaud et al. (2004) have suggested that this may be due to the long memory of market orders being negatively correlated with the long memory of price changes caused by the long memory of limit order arrival and cancelation.

2.3.6 Extreme Price Events

Though the fat-tailed distribution of returns and the high probability of large price movements has been observed across financial markets for many years (as documented in Section 2.3.1), our brave new marketplace has introduced a particularly extreme kind of price event.

Since the introduction of automated and algorithmic trading, recurring periods of high volatility and extreme stock price behaviour have plagued the markets. Johnson et al. (2013) define these so called *flash crashes* as an occurrence of a stock price ticking down [up] at least ten times before ticking up [down] and with a price change exceeding 0.8% of the initial price. Remarkably, they found $18,520$ crashes and spikes with durations less than 1500ms to have occurred between January 3rd 2006 and February 3rd 2011 in

various stocks. One of the many aims of recent regulation such as MiFID II and the DoddFrank Wall Street Reform and Consumer Protection Act is to curtail such extreme price events.

2.4 Market Microstructure: Understanding Order Book Dynamics

The discipline of market microstructure is devoted to shedding light on the processes by which the interactions of market players leads to price formation. In particular, quantitative models of the trading process are proposed in order to improve the efficiency of the allocation of goods, through financial markets, in the economy.

2.4.1 Price Impact

Understanding price impact presents one of the most dominant questions of market microstructure analysis, i.e. how trading activity leads to price changes. The early market microstructure literature describes this concept with a focus on specialist markets. In such markets, prices are quoted by a centralised market maker who receives orders from brokers and updates her quoted prices according to the incoming order flow that she witnesses. From the viewpoint of the broker, the price impact of his orders is a cost paid to the market maker for her continued obligation to accept his orders (Demsetz, 1968), i.e. a cost for immediacy. From the viewpoint of the market-maker, some information about the future prices of assets is inferred from the order flow of the brokers. This information is then captured in the market maker's quotes in a process reflected by the *permanent price impact* (Kyle, 1985). The difference between the price that an order obtains and the best prevailing quote is termed the *immediate price impact* and is an increasing function of order size. The *temporary price impact* is then defined as the difference between the immediate and permanent impact of an order. A visualisation of the two kinds of price impact is given in Figure 2.6.

Although specialist markets, with their centralised market maker, have mainly been replaced with electronic LOBs, the same price impact terminology is still used. However, due to the decentralised nature of LOBs, it is much harder to disentangle temporary and permanent price impact.

In an LOB, we may also consider the impact of an event on the entire state of the LOB. This type of impact is termed *market impact*. Often, the terms 'price impact' and 'market impact' are used interchangeably to refer changes in the best prices, though Hautsch and Huang (2012) have shed light on how the actions of traders can affect the depths available at many price levels, suggesting that the two notions should be separated. Bouchaud et al. (2009) provide a great review of studies of both price and market impact.

A Great deal of research has investigated the impact or individual orders, and has conclusively found that impact follows a concave function of volume. That is, the impact increases more quickly with changes at small volumes and less quickly at larger volumes.

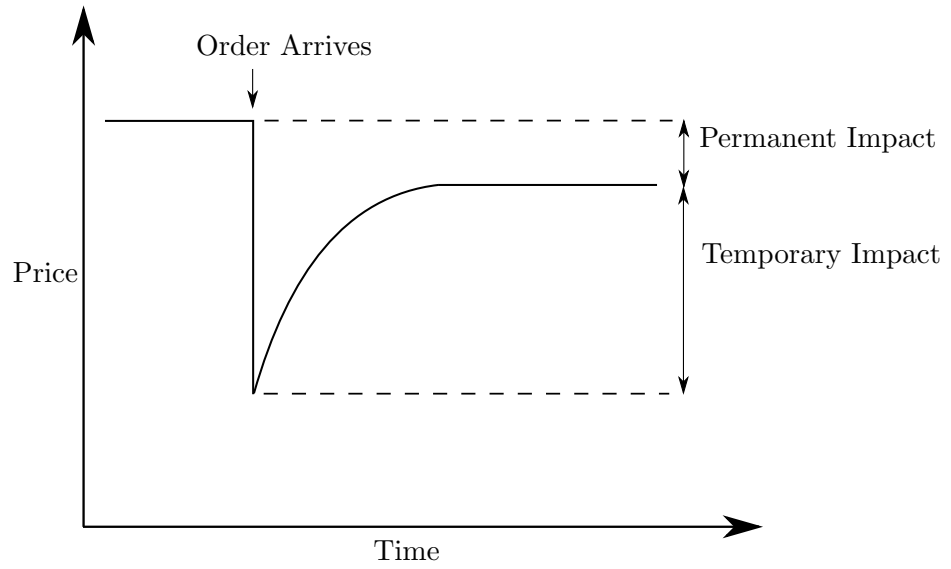


FIGURE 2.6: An illustration of the price impact through time. We can see that impact is instantaneous upon the arrival of an order before a partial recovery of the temporary impact.

However, the detailed functional form has been contensted and varies across markets and market protocols (order priority, tick size, etc.).

Some of the earliest literature found strongly concave functions though did not attempt to identify a functional form (Hasbrouck, 1991; **?**). In a study of the NYSE, Lillo et al. (2003) analysed the stocks of 1000 companies and divided them into groups according to their market capitalisation. Fitting a price impact curve to each group, they found that the curves could be collapsed into a single function that followed a power law distribution of the following form:

$$\Delta p = \frac{\eta v^{\beta}}{\lambda} \tag{2.10}$$

where Δp is the change in the mid-price caused by a trader's action, v is the volume of the trade, η takes the value -1 in the event of a sell and $+1$ in the event of a buy and λ allows for adjustment for market capitalisation. They found the exponent β to be approximately 0.5 for small volumes and 0.2 for large volumes. After normalising for daily volumes, λ was found to vary significantly across stocks with a clear dependence on market capitalisation M approximated by $M \sim \lambda^{\delta}$, with δ in the region of 0.4.

Good approximations of the value for the exponent β have also been found by Farmer and Lillo (2004) on the London Stock Exchange and Hopman (2007) on the Paris Bourse to fall in the range $0.3-0.4$. Consequently, all explorations have identified strongly concave impact functions for individual orders but find slight variations in functional form owing to differences in market protocols.

The square-root impact law, on the other hand, concerns the impact of a parent (or *meta*) orders. Thus, this definition is particularly pertinent as most trades are far too

large to be executed with a single order and, instead, need to be fragmented into many small orders that are executed gradually. This is the domain of *optimal trade execution* and is discussed in the next section. The square impact law of parent orders has been shown to be very universal having been reported by a number of groups across a variety of markets, timescales, trading strategies, microstructures and execution styles (Tóth et al., 2011).

2.4.2 Optimal Trade Execution

One very important implementation of models of price impact lies in optimal trade execution. This is the problem of finding a strategy for buying or selling a large quantity of assets while managing price uncertainty within a limited timeframe. The size of the total trade ('parent order') tends to be far larger than the volume of limit orders available at the best price of the LOB at any given time. Thus, the parent order is sliced into a sequence of smaller 'child orders' that are submitted to the exchange over time. The optimisation of order execution strategies is interesting from both a practical and regulatory standpoint. In the majority of modern markets, regulations require broker-dealers to provide their clients with 'best execution'. There is no concrete definition of best execution [7] and only recently has there been developments in the quantification and and evaluation of trade execution strategies.

There are many models of the optimal execution problem that each differ in their assumptions on price impact, price dynamics and trading mechanism. The seminal paper of this field is often considered to that by Bertsimas and Lo (1998) where a mean execution cost is minimised using dynamic programming. Later this formulation was supplemented by Almgren and Chriss (2001) by including a penalty for the risk inherent in price fluctuations over the course of the execution period. This was the first nod towards the traders' dilemma; where an aversion to price uncertainty presses a trader to trade faster while the cost of immediacy drives her to trade slower. Thus, an optimal execution strategy must balance the risk of future price uncertainty with the price impact cost. This is an interesting problem that has motivated a cast number of studies.

In order to prevent the possibility of arbitrage, price manipulation and erratic strategy behaviours, many restriction have been placed on price impact models. Such restrictions include linearity of the permanent price impact function (Huberman and Stanzl, 2004) and the rate of temporary impact decay (Gatheral, 2010).

The practical applicability of trade execution models have lead to a range of studies attempting to link the models to order placement decisions. However, many of these studies introduce restrictive assumption on strategies and market dynamics. One of the

[7] The Financial Services Authority (FSA) describe this as "...taking all reasonable steps to obtain the 'best possible result' for their clients.". See http://www.fsa.gov.uk/about/what/international/mifid/background/key-topics/best-execution

more popular approaches restricts strategies to using only marketable orders and explicitly computes the cost of these orders by integrating a density function that represents the orders at the top of the LOB (Obizhaeva and Wang, 2013; Predoiu et al., 2011). In such models, the cost is determined by the initial impact as well as the speed of the recovery. This approach is an improvement compared to previous models, though its restriction to only marketable orders is limiting.

Recent research has proposed optimal execution strategies that allow passive limit orders (Bayraktar and Ludkovski, 2011). In this methodology, limit order executions are modelled using a point process where the density is a function of the distance between the limit order price and the current best prices. As this distance is under the control of a trader, the execution problem, under this formulation, may be solved using stochastic control methodologies. There are three major problems with this formulation. Firstly, it is necessary to specify a joint process of price and limit order book dynamics. Second, even with very strong assumptions the resulting formulations are often intractable. Lastly, only very basic details of order placement are included in the formulation. This ignores important information regarding the general state of the LOB that are known to be important for order placement decisions.

In summary, the problem of optimal order placement in limit order book markets remains a complex optimisation problem where today's state-of-the art relies on intractable systems of equations and/or unrealistic assumptions about market dynamics. As such, there is room for for improved models of order book dynamics that incorporate the complete state of the LOB and provide a platform for optimising trade execution.

2.5 Modelling Limit Order Books

The financial community has expressed an active interest in developing models of LOB markets that are realistic, practical and tractable (Avellaneda and Stoikov, 2008; Obizhaeva and Wang, 2013; Predoiu et al., 2011). The literature on this topic is divided into four main streams: theoretical equilibrium models from financial economics, statistical order book models from econophysics, stochastic models from the mathematical finance community, and ABMs from complexity science.

Financial economics models tend to be built upon the idea of liquidity being consumed during a trade and then replenished as liquidity providers try to benefit. **?** and **?**, for example, describe theoretical models of LOB markets finite levels of resilience in equilibrium depending mainly on the characteristics of the market participants. In these models, the level of resilience reflects the volume of hidden liquidity. Many models are partial equilibrium in nature. taking the dynamics of the limit order book as given. For example, **?** provide a framework that allows discrete orders and more general dynamics, while **?** implement general but continous limit order books. In order to operate in a full equilibrium setting, models have to heavily limit the se of possible order-placement strategies. **?**, for example, allows only orders of a given size, while **?** only explore single-shot strategies. Though these simplifications enable the models to more precisely describe the tradeoffs presented by market participants, it comes at the cost of unrealistic assumptions and simplified settings. It is rarely possible to estimate the parameters of these models from real data and their practical applicability is limited. Statistical models, on the other hand, tend to fit the data well but often lack economic rigour and typically involve the tuning of a number of free parameters. Consequently, their practicability is questioned. Given the lack of practical applicability of purely theoretical and statistical models, this report will focus on stochastic and ABMs of LOBs.

2.5.1 Stochastic Order Book Models

Stochastic order book models attempt to balance descriptive power and analytical tractability. Such models are distinguished by their representation of aggregate order flows by a random process, commonly a Poisson process (Cont et al., 2010; Farmer et al., 2005). Given this representation, order books are viewed as a stochastic queuing system in which orders randomly arrive and are cancelled. This modelling approach attempts to circumvent the issue of making detailed assumptions about traders' utilities and strategies by instead making assumptions on the (more easily measured) statistical properties of aggregate order flows. Such a method makes it easier to calibrate against LOB data, is more flexible than economic equilibrium models, and possesses more structure than descriptive statistical models.

The stochastic modelling of LOBs is founded on a rich literature of queuing theory and

as such has proved useful in practical applications. Notably, it has been found that it is effective to submit aggressive orders when the probability of an adverse price change is high (Stoikov and Waeber, 2012). Such probabilities are easily calculated in these models using Laplace transforms (Huang and Kercheval, 2012) and analytical results can also be derived from scaling approximations. In another example, a queuing LOB model is used in a heavy-traffic regime to estimate the distribution of durations between price changes (Cont and De Larrard, 2011). The study found that micro-level price changes in a queueing LOB model result in Brownian motion at a larger scale, and that volatility is a function of the average order queue size and the order arrival rate.

Further empirical studies of level 2 market data with trader identifiers (Kirilenko et al., 2014; Baron et al., 2012) have highlighted the presence of complex interactions and feedback effects between different kinds of market participants. Although the queueing models highlighted above describe the behaviour of order flows from all participants in a limit order book, stochastic models are not able to cope with interactions and dependancies between participants.

One branch of the stochastic modelling literature - the multi-class queuing literature - has made a step towards addressing interaction by introducing systems with heterogenous participants. In these models, differences between agents are expressed by assigning each class of participant a distinct distribution (Jennings and Reed, 2012). Although multi-class queuing models can capture some of the differences between market participants, they do not fully capture the interactions between participants, nor do they account for agents that act on different timescales - a features common to modern financial markets (Hasbrouck and Saar, 2013).

Thus, the high level statistical description of participant behaviour inherent in stochastic order book models ignores important complex interactions between market participants and fails to explain many phenomena that arise. As such, a richer bottom-up modelling approach is needed to enable the further exploration and understanding of limit order markets.

2.5.2 Agent Based Models

Agent-based modelling (ABM) is a modelling tool used to simulate the interaction of elements in complex systems. Complex systems are those composed of a number of elements that react and adapt to the environment that they create. Examples of such elements include cells in an immune system, neurones in a brain, or ions in a spin glass, and they react to chemical signals, neighbouring cell activations, or local magnetic moments. The evolution of the system through time plays an important role as the elements react, systemic changes emerge and elements react afresh.

ABMs are ideally suited to modelling complex systems as the agents are represented by

autonomous computer software that are embedded in a virtual environment that they can sense and act upon. Agents often maintain internal beliefs and/or desires, reason about the state of their environment, and may bargain or negotiate (Jennings et al., 1998). This movement is said to have evolved from early AI research. The agent-based mind-set is rooted in the school of decentralised AI, which focuses on collaborative solutions to problems with distributed group of entities (Demazeau and Müller, 1990). Fundamentally, AI concerns itself with building intelligent artefacts that sense and interact in an environment and thus can be considered to be dealing with agents (Russell and Norvig, 1995).

Though there are clear ties with early AI research, ABM research can be traced further back. One of the earliest and most poignant applications of the ABM was Schelling's (1971) model of segregation. The model takes the from of a two-dimensional grid of cells that represents an urban area, upon which households are placed at random. Each cell on the grid may contain only one household at a time and a large number of the cells are empty at any given time. There are two distinct kinds of agents, let's say "red" and "blue". At each timestep, the following occurs:

- Each household observes the eight cells surrounding it, it's neighbours, and identifies the fraction of households that are of the other colour.
- If the observed fraction is greater than some tolerance threshold that house declares itself 'unhappy' and relocating to a a random vacant cell on the grid.

In the next timestep, this newly relocated household may tip the balance of colour in the area causing some of its neighbours to become happy. This can cause cascades of relocation across the grid. Schelling found that a randomly initialised set of households would segregate int batched of red and blue for any tolerance threshold of 0.3 or above. He interpreted this as indicating that even very slight racial prejudices could lead to the types go ghettoisation observes in many U.S. cities in the nineteen seventeen.

In the early eighties, Axelrod and Hamilton (1981) peaked interested in the prisoner's dilemma with his book that described a tournament he had organised in which academics could enter strategies into an N step prisoner's dilemma game with memory. In the prisoners dilemma, you and an accomplice have committed a crime and been apprehended. You are both being detained in separate cells and are unable to communicate with each other. Both of you are offered the same deal by the police:

- If your accomplice confesses and you deny the crime, you go to prison for ten years and your accomplice walks.
- If you confess and you accomplice denies the crime the you walk and your accomplice gets ten years.

- If you both confess then you will both server six years.
- If you both deny then youll both receive six months.

In the iterated version of the game, the two players play more than once with a memory of their accomplice's previous action and the ability to change their strategy. Axelrod found that, in large iterated games, altruist strategies tended to success and pure greedy strategies did not fair so well. He proposed this as an addendum to natural selection to explain the evolution of altruistic behaviour from initially selfish mechanisms. His work on the prisoner's dilemma made waves in a number of fields including economics, politics and evolutionary biology.

Towards the end of the 1980s, Reynolds (1987) developed some of the first agent-based models for biology to contain social characteristics. His work explored the an approach to simulating flocking behaviour as an elaborate particle system. Each bird was simulated as a individual agent that makes directional decisions based on its interaction with its environment. The emergent motion of the flock was found to be the result of tightly coupled interactions between the simple behaviours of the simulated bird. Through experimentation, Reynolds found that a significant property of accurate flocking behaviour was it's unpredictability over anything more than medium timescales. At very short timescale, say one second, it is trivial to predict the motion of a bird - approximately the same speed and direction. However, prediction at greater timescale is akin to that of chaotic systems. This property is typical of complex systems and falls somewhere between ordered behaviour, which is easily predictable, and chaotic behaviour, which is not predictable of short of long timescales.

As well as the pioneering papers discussed above, ABMs have broken ground in a number of field including from population dynamics (Caplat et al., 2008), ecological modelling (Grimm et al., 2005), and tissue formation (Tang et al., 2011) through to traffic management (Erol et al., 2000), wireless sensor networks (Niazi and Hussain, 2011), and organisational management (Hughes et al., 2012).

2.5.2.1 Applications in Finance and Economics

Naturally, complex systems arise across economics and finance. Agents such as banks, funds, brokers, or traders, constantly adjust portfolios, trading strategies and forecasts to the landscape that these actions create. However, unlike immune cells of the body, which always react in a simple way to their local chemical environment, economic agents react with strategic behaviour, considering outcomes that might result as a consequence of their behaviour. This results in a layer of complexity not found in the physical and natural sciences.

Perhaps in owing to the financial crisis, the popularity of ABMs as tools for economic modelling has increased rapidly. The power of ABMs lies in their freedom from assumptions of equilibrium and representative agents. Instead, they consist of heterogeneous populations of agents with independent behaviours that may even learn over time. Their bottom up approach allows room for emergent phenomena such as bubbles and crashes to bubble out of the local interactions between agents (Tesfatsion and Judd, 2006). A number of highly respected academics have voiced their opinions that ABMs hold the key to representing a complex economy while at the same time building a models based on microfoundations (Farmer and Foley, 2009).

While interest may have peaked of late, a handful of economists in the early 1990s found a use for autonomous agents as a model for human behaviour in complex, real-world systems - so called agent-based modelling. Rust et al. (1992) document a series of CDA tournaments in which autonomous computer agents play the roles of buyers and sellers in a CDA. They found that even with the decentralised nature of the CDA and the self interest of the agents, price trajectories typically converged to competitive equilibrium. Also, they showed that the best performing agent was among the most simple of strategies with a simple rule-of-thumb outperforming many complex agents that incorporated machine learning and statistics prediction models of future prices. That simple strategy was devised by Todd Kaplan and, as such, has since become widely known as "Kaplan's Sniper".

A few years later, in a particularly famous paper, Gode and Sunder (1993) used a combination of autonomous software agents and human subjects to replicate and further a series of experiments first conducted by Smith (1962) in a field later dubbed experimental economics. Gode and Sunder's aim was to investigate how much of the efficient properties of the CDA could be attributed to the structure of the auction itself, rather than the behaviour of the humans that participated in it. Their experiments lead them to believe that it was the structure and rule set of the CDA alone and not the behaviour of agents that lead to the high allocative efficiency even citing Adam Smith's "invisible hand". This was later debunked in a rigorous mathematical treatment by Cliff and Bruten (1997).

Following his confuting of Gode and Sunders hypothesis, Cliff (1997) set out to find the simplest possible agent that could explain human bargaining behaviours. This experimentation gave birth to the Zero-Intelligence-Plus (ZIP) algorithm for trading in periodic CDAs which used a simple machine learning algorithm, Widrow-Hoff with momentum, to adapt the agents margin to prevailing market conditions. Aside from demonstrating realistic market dynamics, the ZIP algorithm, along with a modified GD algorithm (Gjerstad and Dickhaut, 1998), was later shown to consistently outperform human traders (Das et al., 2001).

The excitement surrounding an algorithm that could beat human traders lead to a

flurry of research into algorithms for trading in real global financial markets. In the academic sphere, this manifested itself in the development of a number of simulation environments for algorithmic trading strategies including the Penn-Lehman Automated Trading (PLAT) (Kearns and Ortiz, 2003) and the UCL Algorithmic Trading competition though both lacked any real model of market impact.

The agent-based economics explored by Gode, Sunder, and Cliff were the first studies to demonstrate the effectiveness of ABMs for investigating complex financial systems and the power of relaxing some of the unrealistic assumptions required for theoretical models. This seminal literature has lead to a number of prominent ABMs that have proven themselves particularly useful for understanding, e.g. the interactions between trading algorithms and human traders (De Luca et al., 2011), empirical regularities in the inter-bank foreign exchange market Chakrabarti (2000), and the complexities of systemic risk in the wider economy (Geanakoplos et al., 2012).

In a particularly interesting piece of work, Thurner et al. (2012) build an ABM of leveraged asset purchases with margin calls. They find that downward price fluctuations during boom times (where firms are highly leveraged) can cause a chain reaction of margin calls leading to clustered volatility, power-law tails and trend-following. They contrast this with previous explanations of fat tails and clustered volatility which are dependant on irrational behaviour from agents. They note that a common risk control policy among banks (whereby leverage limits are based on volatility) exacerbates liquidity problems causing extreme price fluctuation.

A huge win for the field came from the work of Darley and Outkin (2007) who worked with Nasdaq to help understand the effects of decimalising its tick size. Many expressed opinions that decimalisation would enable buyers and sellers to communicate in more precise terms, improving spreads, while others were concerned about inefficiencies or loopholes that may be introduced as result of the change. The team from Santa Fe described a sizeable model that involved the detailed replication of the behaviours of a number of market participants including: casual investors, day traders, institutional investors and market makers. Their model suggested that decimalisation would likely lead to impaired price discover and wider spreads. As Nasdaq decimalised,Darley and Outkin's (2007) predictions rang true but enabled Nasdaq to better prepare for the effects of a reduced tick size.

The effectiveness of ABMs has also been demonstrated with LOBs. The first ABMs of LOBs assume the sequential arrival of agents and the emptying of the LOB after each time step (see e.g. Foucault (1999)). Unfortunately, Smith et al. (2003) notes that approaches such as this fail to appreciate the function of the LOB to store liquidity for future consumption. More recently, ABMs have begun to closely mimic true order books and successfully reproduce a number of the statistical features described in Section 2.3.

To this end, Cont and Bouchaud (2000) demonstrate that in a simplified market where

trading agents imitate each other, the resultant returns series fits a fat-tailed distribution and clustered volatility. Furthermore, Chiarella and Iori (2002) describe a model in which agents share a common valuation for the asset traded in a LOB. They find that the volatility produced by in their model is far lower than is found in the real world and there is no volatility clustering. They thus suggest that significant heterogeneity is required for the properties of volatility to emerge.

Farmer and Joshi (2002) use a method of price formation based on a simulated market maker to explore the price dynamics generated by a number of commonly applied trading strategies. They find that certain combinations of strategies give rise to amplified noise and excess volatility. In a later model, Challet and Stinchcombe (2003) note that most LOB models assume that trader parameters remain constant through time and explore how varying such parameters through time affected the price time series. They find that time dependance results in the emergence of autocorrelated mid-price returns, volatility clustering and the fat-tailed distribution of mid-price changes and they suggest that many empirical regularities might be a result of traders modifying their actions through time.

Correspondingly, Preis et al. (2006) reproduced the main findings of the state-of-the-art stochastic models using an ABM rather than and independent Poisson process, while Preis et al. (2007) digs deeper and explores the effects of individual agents in the model. They found that the Hurst exponent of the mid-price return series depends strongly on the relative numbers of agent types in the model.

At a slightly lower level, Mastromatteo et al. (2014) use a dynamical-systems / agent-based approach to understand the non-additive, square-root dependence of the impact of meta-orders in financial markets. Their model finds that this function is independent of epoch, microstructure and execution style. Although their study lends strong support to the idea that the square-root impact function is both highly generic and robust, Johnson et al. (2013) notes that it is somewhat specialised and lacks some of the important agent-agent interactions that give rise to spikes and crashes in price that have been seen to regularly occur in LOB markets.

Similarly, Oesch (2014) describes an ABM that highlights the importance of the long memory of order flow and the selective liquidity behaviour of agents in replicating the concave price impact function of order sizes. Although the model is able to replicate the existence of temporary and permanent price impact, its use as an environment for developing and testing trade execution strategies is limited. In its current form, the model lacks agents whose strategic behaviours depend on other market participants.

2.6 Summary

This chapter has outlined why reliable models of price and order book dynamics are so important. It has also provided a detailed overview of the current state-of-the-art methodologies for predicting inter-day price movements, forecasting price impact in LOBs, and modelling LOB dynamics.

Specifically, the studies of ensemble learning techniques described in Section 2.2.4 demonstrate the ability to avoid overfitting compared to single expert systems. Against this background, we develop an automated system for predicting the price return during the seasonal regularities in stock prices that are described in the Section 3.2. The system is composed of an ensemble of performance-weighted random forests where new experts are trained on the previously unseen data. These experts are then added to the ensemble in an online fashion to allow the algorithm to adapt to changing market regimes. A risk management layer is implemented to prevent the large drawdowns commonly seen in many systems. This novel technique of online generation of random forests and combining predictions in a recent-performance weighted average draws on the proven capabilities of random forests and ensemble approaches to avoid overfitting.

Correspondingly, the ability of modern machine learning approaches to capture complex interaction within data lend themselves perfectly to isolating patterns in high-frequency limit order book (LOB) data and overcoming many of the simplifications and assumptions that are inherent in the models discussed in Section 2.4. Thus, in Chapter 4, we propose an adaptation to the system specified in Chapter 3 for the application of a performance weighted ensemble of random forests to predicting price impact. Specifically, we address the potential overfitting problem common to financial data by using random forests and tackle the non-stationary element of data by generating random forests in an online fashion.

With regards to modelling LOBs, though each of the models described in Section 2.5.2 are able to replicate or explain one or two of the stylised facts reported in Section2.3, no one model exists that demonstrates all empirically observed regularities — a clear requirement of a model intended for real-world validation. Also, no paper has yet presented agents that operate on varying timescales. Against this background, we propose a novel modelling environment that includes a number of agents with strategic behaviours that act on differing timescales as it is these features, we believe, that are essential in dictating the more complex patterns seen in high-frequency order-driven markets.

Chapter 3

Explorations in Forecasting Price Returns - Daily Equities Data

In this chapter, we introduce an expert system for trading stocks that uses performance-weighted ensembles of random forests to predict the price return during well documented seasonality events. Although these seasonal regularities are consistent enough to be used as a stand-alone strategy[1], rates of return are highly volatile (Hong and Yu, 2009). To address this problem, we propose an expert system that captures the current state of the market in its inputs and uses this information to both predict the profitability of a seasonality trade and act upon this information. In doing so, the system performs risk management while opening and closing trading positions in an attempt to improve the profit and reduce the volatility over a basic seasonality trading strategy.

Specifically, we develop an autonomous system for the trading of stocks over seasonality events. To that end, a recency-biased performance-weighted ensemble of random forests is used to predict the expected profit of a seasonality trade given the prevailing market conditions. The random forests are trained and added to the ensemble over time, in an online fashion, so as to capture various phases of the market. A prediction is generated as a weighted average of the predictions of all random forests and is passed through a risk management layer before trades are initiated. Again, let us note, we are not aiming to build the world's most profitable, accurate and stable prediction system. Instead we hope to build the simplest possible model that generates consistant profitability so that we can explore drivers of market dynamics.

Before describing such a system, we first investigate the Deutsche Borse Ag German Stock Index (DAX), Financial Times Stock Exchange 100 index (FTSE 100) and Standard & Poor's 500 index (S&P 500) for the following regularities: upward biases at

[1] A simple seasonality strategy involves buying (selling) an asset before seasonal events that have historically been seen to produce temporary upward (downward) price trends. As we will demonstrate shortly, such events include weekends, turn-of-the-month and exchange holidays.

the turn-of-the-month and over exchange holidays, as well as a downward bias over the weekend.

This chapter is structured as follows. In Section 3.1 we analyse the structure of the data while in Section 3.2 we explore the persistence and reliability of the aforementioned seasonality effects before proposing a number of features and a feature selection method. Section 3.3 describes the trading algorithm itself. In Section 3.4 the results are presented while Section 3.5 provides a summary.

3.1 The Data

The dataset used in this chapter involves daily open, close, high, low and volume data of stocks from the DAX, FTSE 100 and S&P 500 stock indices over the period of 01/01/2000 - 01/01/2013. This particular choice of markets is insignificant due to the similarity of daily time series data across markets.

Before conducting experiments, the data set is split into three non-overlapping sets:

Training - 2000-2008 This data is used for the exploratory analysis in Sections 3.1 and 3.2 and also for the training phase of our ensemble model. During this phase, no positions are taken in the market. In fact, no predictions are made by the ensemble as a whole, this phase is purely use for the staggered training of new random forest and their addition to the ensemble.

Validation - 2008-2010 During the validation phase, the complete ensemble model produces predictions that go on to influence trading decisions. Grid searches of the parameter space are performed on this dataset to enable the optimisation of the model's free parameters.

Test - 2010-2012 - No parameters are tuned using this set of data. One ensemble model, with the set of parameters found to perform best during the validation phase is used to make predictions and influence trading decisions using the test set to produce a true out-of-sample test of performance.

Non-normality

As well as its widely demonstrated ability to generalise, our choice of the non-parametric RF algorithm as the basis of our prediction system has been driven by the non-Gaussian nature of financial data. Figure 3.1 shows a commonly used gauge of normality (Wilk and Gnanadesikan, 1968), the Q-Q plot of the returns series of a typical actively traded stock.

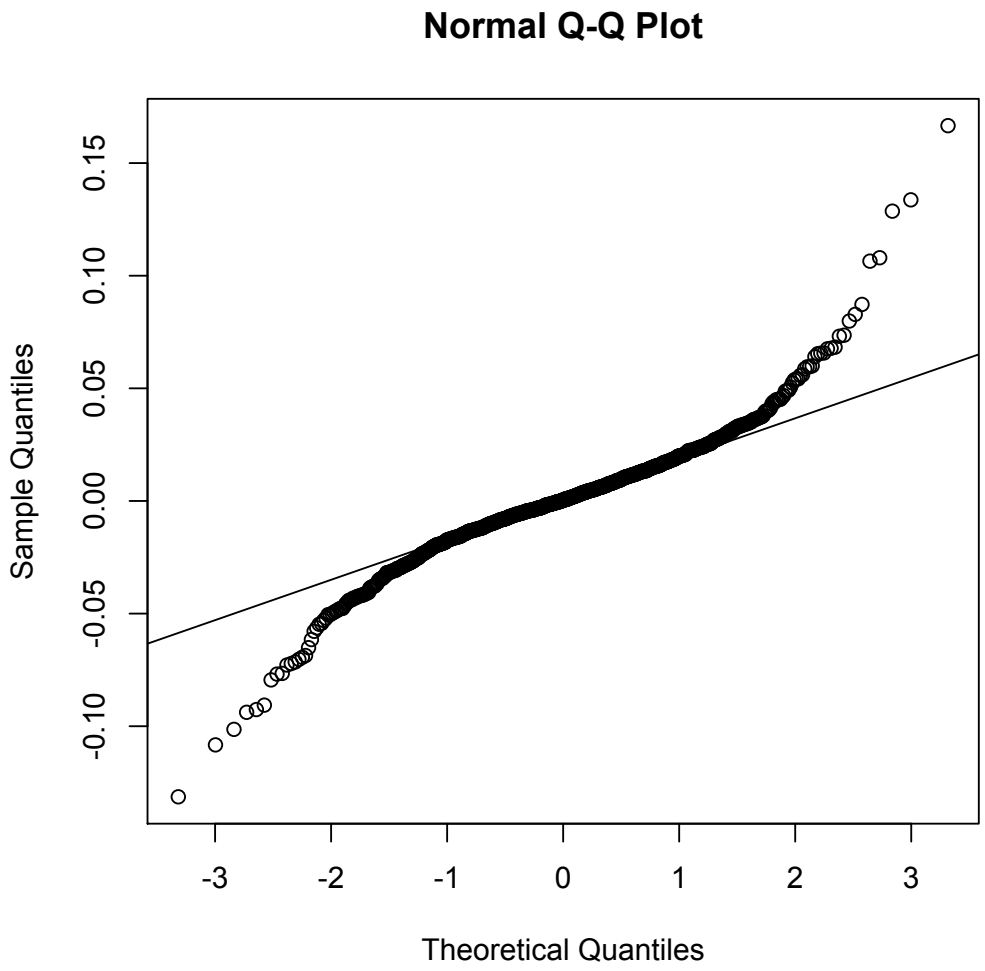


FIGURE 3.1: QQ-plot of the distribution of returns of the GOOG stock from 01/01/2000 to 01/01/2009. The downward curvature on the left hand side and the upward curvature on the right are indicative of the kind of fat-tailed distribution often found in financial data.

We can see from Figure 3.1 that the distribution of stock returns deviates substantially from Gaussian. The left end of the pattern falling far below the line $y = x$ and the right end laying far above indicates a substantially long tailed distribution that is typical of financial data. We can confirm the departure from normality with a Shapiro-Wilk test (Shapiro and Wilk, 1965), which produces a test statistic of $W = 0.93$ with a p-value $< 2 \times 10^{-16}$. This allows us to reject the null hypothesis of normally distributed data.

Non-stationarity

Financial time-series are notoriously non-stationary and have means, variances and covariances that change over time. We can demonstrate this by performing a Kwiatkowski, Phillips, Schmidt and Shin (KPSS) test on the price time series of stocks (Shapiro and

Wilk, 1965). With the KPSS test, we assume there is no trend and form a null hypothesis that the data is stationary and an alternative hypothesis that that the data generating process is stochastic with unit root.

Taking the same example as above, we perform the KPSS test on the price time series of the GOOG stock from 01/01/2000 to 01/01/2009. This yields a KPSS Level of 1.005, a truncation lag parameter of 7 and a p-value < 0.01. As our null was that the data follows a straight line time trend with stationary errors, we may reject it at the usual 5% level.

Many traditional prediction and machine learning models rely on stationary data to make predictions. Thus, to enable effective prediction, we propose the use of an online learning layer to enable our system to adapt and continue to produce viable predictions after the training of the initial models. The online learning layer is described fully in Section 3.3.

3.2 Seasonality Effects and Feature Selection

Many seasonalities and empirical regularities of financial markets have been documented in the literature with events including turn-of-the month, weekends effects and exchange holidays. In the following we discuss these events in full.

In more detail, turn-of-the-month effects were first documented by Ariel (1987) who found that the return for the latter half of the average month is negative and that positive returns occur in the first part of the month. This result was further explored by Lakonishok and Smidt (1988). Using a 90-year series for the Dow Jones Industrial Average (DJIA), they showed that the returns for the four days around the turn of the month, starting with the last day of the prior month, is 0.473 percent with the average return for any four-day period being 0.0612 percent.

The first study of weekend effects in security markets appeared in 1931. In this study, Fields (1931) examined the pattern of the DJIA for the period 1915-1930 to see if the unwillingness of traders to carry their holdings over weekends lead to a liquidation of long accounts and a consequent decline of security prices. He compared the closing price of the DJIA for Saturday with the closing prices on the adjacent Friday and Monday. He found that, in fact, prices tended to increase on Saturdays. Since then, numerous studies have reinforced Fields' findings, confirming that asset prices tend to be lower on Mondays than the preceding Fridays (Cross, 1973; French, 1980; Rogalski, 1984).

In French's investigation of weekend effects he also looked at the price behaviour after exchange holidays, finding no empirical regularities. However, in another very early study, Fields (1934) found that the DJIA showed a high proportion of positive returns one day prior to holidays. These results were confirmed by Ariel (1990) who looked

Event	**Percentage of times trend was observed (%)**		
	DAX	FTSE 100	S&P 500
Turn-of-month	62.6	61.1	54.0
Holiday	73.2	57.2	58.1
Weekend	67.5	56.7	51.9
Percentage of total days in which upward price movement occurred	52.4	52.3	51.1

TABLE 3.1: Table showing the percentage of times that each of the following seasonality effects was observed in the DAX, FTSE 100 and S&P 500 for the period of 2000-2010: turn-of-the-month, exchange holiday, weekend effect. These figures should be compared to the percentage of upward market movements for all days.

at the returns on the days that surrounded exchange holidays during 1963-1982. It was found that the mean return on the preholidays was significantly greater than other days; a results that has been further supported by others (Cadsby and Ratner, 1992; Lakonishok and Smidt, 1988).

In the following section, we explore the persistence and reliability of the aforementioned seasonality effects.

3.2.1 Explorations in Seasonality

We analyse data from the DAX, FTSE 100 and S&P 500 over the years 2000 to 2010 to explore the existence and reliability of the regularities described above. We test the hypothesis that these events lead to an average positive or negative return for the index. We perform these investigations to ensure that the seasonality effects have not been arbitraged away following the publishing of the studies above. To do this we measure returns over a particular event, reporting the percentage of times the expected trend was observed from: the third week of a month to the first week of the next month; the day before exchange holiday until the day after; Friday to the following Monday. Results are given in Table 3.1. We can see that all of the aforementioned regularities are apparent in the DAX data with holiday effect being particularly consistent. In both the FTSE 100 and the S&P 500 the seasonality effects are far weaker and thus, from here on in, we will focus on the DAX.

In order to make a seasonality trading strategy more robust, it seems appropriate to investigate whether these regularities are consistent through time or whether they are more pronounced at particular times. For this we have calculated the percentage of time that the index has followed the seasonality effect during each month over the entire dataset the result is shown in Figure 3.2. The percentage for each month includes all events (weekend, turn-of-month or exchange holiday) that occurred within that month

while events that span two months are included in the month during which the event began.

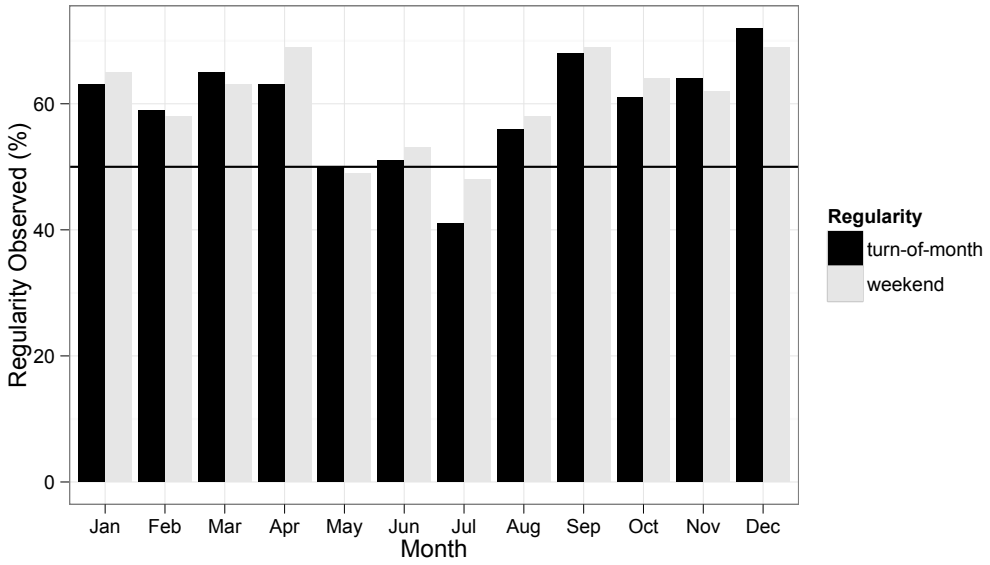


FIGURE 3.2: The difference in the consistency of the weekend effect and the turn-of-month effect between months over the period of 2000-2010

As Figure 3.2 demonstrates, there are significant differences in the consistency of the regularities between months. It is evident that the regularities hold strong during both the first and last few months of each year. However, following a simple seasonality strategy would be highly unprofitable during the middle third of the year.[2] Thus, in order to generate a consistently profitable trading strategy based on seasonality trading, it is important to take into account the month of the year and the type of event in question. To this end, we explore the relative importance of various criteria on predicting the price change over a seasonality event.

3.2.2 Features

We propose a number of features (see table A.1 in Appendix A) to capture the state of the market, as best we can, at a given time. First, the three seasonality events discussed above are included: the exchange holidays, turn-of-the-month, and the weekend. Additionally, the open, high low, and close values of the day before the event are explored, as well as a number of technical indicators, with various parameterisations, which are described in Table A.2 on page 119.

As well as the aforementioned technical indicators we include the current month, the

[2]When referring to a "simple seasonality strategy" we mean always adhering to the following: buy a stock in the third week of the month and sell in the first week of the next, sell stock on Friday and buy it back the proceeding Monday and buy stock the day before an exchange holiday selling it back the day after.

close price of the index to which the stock belongs, and our own risk-based indicator. For the latter, we take a concept from the value-at-risk (VaR) literature to engineer a feature that signals whether it is likely to be a particularly risky time to trade. The idea of this is to equip the prediction algorithm with a feature that acts in a self regulatory way, indicating when the market is acting in an unusual way. To do this, we calculate a parametric VaR of a buy-and-hold strategy on the asset we are trading. Specifically, we assume that the daily price changes of an asset are drawn from a Gaussian distribution, the variance of which we estimate with an exponentially weighted moving average:

$$\sigma_t^2 = \Lambda\sigma_{t-1}^2 + (1-\Lambda)r_{g,t}^2 \tag{3.1}$$

where σ_t^2 is the variance at time t, $r_{g,t}$ is the return of stock g, and Λ is the decay factor. We set Λ to 0.94. Once we have an estimate of the variance, a standard 95 percent VaR is estimated as 1.96 times the square root of the variance. This value can be interpreted as follows: on any given day there is a 5% chance that the buy-and-hold strategy will lose more that the VaR estimate. However, the standard VaR does not cope well with the heteroscedasticity inherent in financial returns data. To overcome this, we introduce a VaR scaling factor. Each day that a VaR break occurs (a drawdown exceeding our 95% VaR) we add one to the VaR scaling factor while each day that a VaR break does not occur we multiply the VaR scaling factor by Λ. Thus, the VaR is calculated as follows:

$$\text{VaR}_t = 1.96\nu_t\sqrt{\sigma_t^2} \tag{3.2}$$

where the *VaR scaling factor*, ν_t, is calculated as:

$$\nu_t = \begin{cases} \nu_{t-1} + 1 & \text{if a VaR break occurred at } t-1 \\ \Lambda\nu_{t-1} & \text{otherwise} \end{cases} \tag{3.3}$$

This violent reaction of the VaR scaling factor to any unusual event and the gradual settling down afterwards is what allows our measure to cope with heteroscedasticity.

3.2.3 Feature Selection

In order to select only the most effective features, we eliminate those features that have little or no impact on the performance of the predictive model. To this end, we use a method of feature importance ranking, first suggested by Breiman (2001).

In particular, to rank the features, a single random forest (as described on page 31 of Section 2.2.4) is trained using the training data. In doing so, the root-mean-square error (RMSE) on the out-of-bag portion of the data[3] is recorded for each tree. This is repeated after randomly permuting the values of each of the features. The difference in

[3]Given that each tree in a random forest is constructed using a unique bootstrap sample of about 60% of the data, the rest of the data can be used to generate unbiased error estimates.

error between the permuted and non permuted trees are then averaged and normalised by the standard deviation of the differences. To this end, the importance of feature j, $\mathcal{VI}_j$, is calculated as:

$$\mathcal{VI}_j = \frac{\sum_{\theta=1}^{\Theta} (e_{\theta,j} - e_{\theta,\pi j})}{\Theta \cdot \hat{\sigma}} \tag{3.4}$$

where Θ is the total number of trees in the forest, $e_{\theta,j}$ is the RMSE of tree θ before permuting j, $e_{\theta,\pi j}$ is the RMSE of that tree after permutation of j, and $\hat{\sigma}$ is the standard deviation of the differences between $e_{\theta,j}$ and $e_{\theta,\pi j}$. A feature that produces a large $\mathcal{VI}$ value is considered more important than a feature with a lower $\mathcal{VI}$.

To select the features to be used in our expert system, we propose a backwards elimination method. We do so as it has been shown that such methods provide a stronger feature subset than similar alternatives (Guyon and Elisseeff, 2003). The feature selection method that we use is described fully in Algorithm 5.

Algorithm 5 Our feature elimination algorithm based on the feature importance ranking method first proposed by Breiman (2001).

1: Train a random forest using the training data with all J features
2: Compute the average RMSE of the model on validation data
3: Rank the features according to performance as described by equation 3.4
4: **for** each subset of variables $J_i = J-1, J-2, \ldots 1$ **do**
5: Train a new forest using J_i features with the highest $\mathcal{VI}$
6: Compute the average RMSE of model on validation set
7: Rerank the features
8: **end for**
9: Determine which J_i yielded the smallest RMSE

The dataset used in this chapter involves 30 stocks from the DAX stock index over the period of 01/01/2000 - 01/01/2013. Before conducting experiments, the data set is divided into training, cross validation (CV) and test sets as follows: 2000-2008 training, 2008-2010 validation and 2010-2012 test. Algorithm 5 is applied to the training data and a plot of the RMSE at each stage of the elimination is given in Figure 3.3. It is evident that there is a initial increase in performance (decrease in RMSE) as the unnecessary features are dropped. Following this, there is a rapid decline in performance as features that are essential for prediction are eliminated. The feature selection algorithm yields the optimal set of 23 features shaded grey in table listed in Table A.1 in Appendix A. Note that all continuous valued inputs are taken as logs and normalised with the stock's closing price.

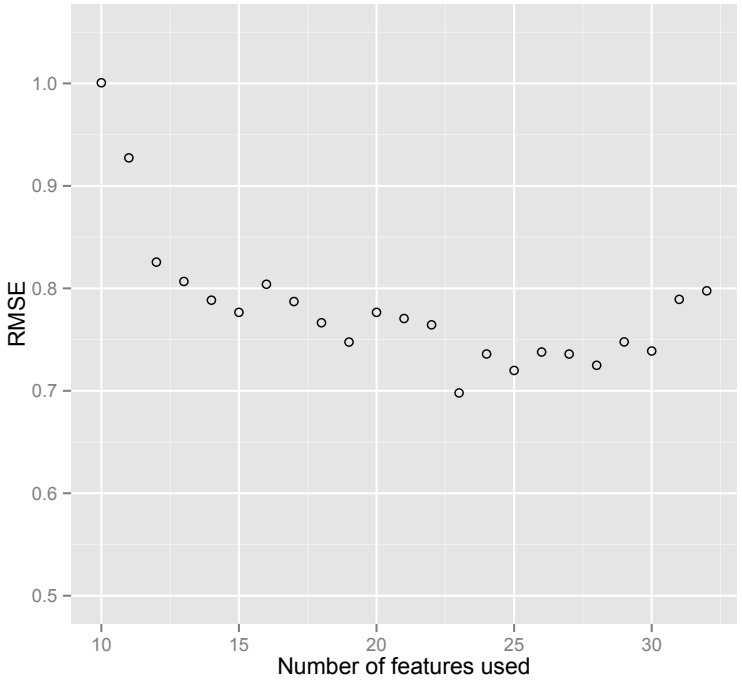


FIGURE 3.3: Plot of RMSE for each round of the feature elimination algorithm. It can be seen that, as features are removed, there is an initial, slight improvement in performance before a rapid decline.

3.3 Trading Model

The trading system discussed in this chapter is designed to trade stocks and consists of three layers: random forest prediction, expert weighting and risk management. While each of these layers is discussed in greater detail below, an overview of the system is given in Figure 3.4.

3.3.1 Layer 1: The Random Forest Prediction Algorithm

The role of the prediction layer is to generate a random forest (RF) every d days to form an ensemble of RFs that predicts the amount of profit that would be realised from buying a stock the day before a seasonality event and selling it two days after (a seasonality trade).

Specifically, we build random forests from an ensemble of regression trees as described on page 31 in Section 2.2.4. Each RF, i, at time t produces the prediction $S_{i,t}$ using the features given in Table A.1 as inputs. In our study, the goal of the prediction layer is to generate RFs in an online fashion that each represent a function of the inputs for predicting the dollar profit from a seasonality trade. The outputs of all RFs are combined using a performance weighted system described in the next section.

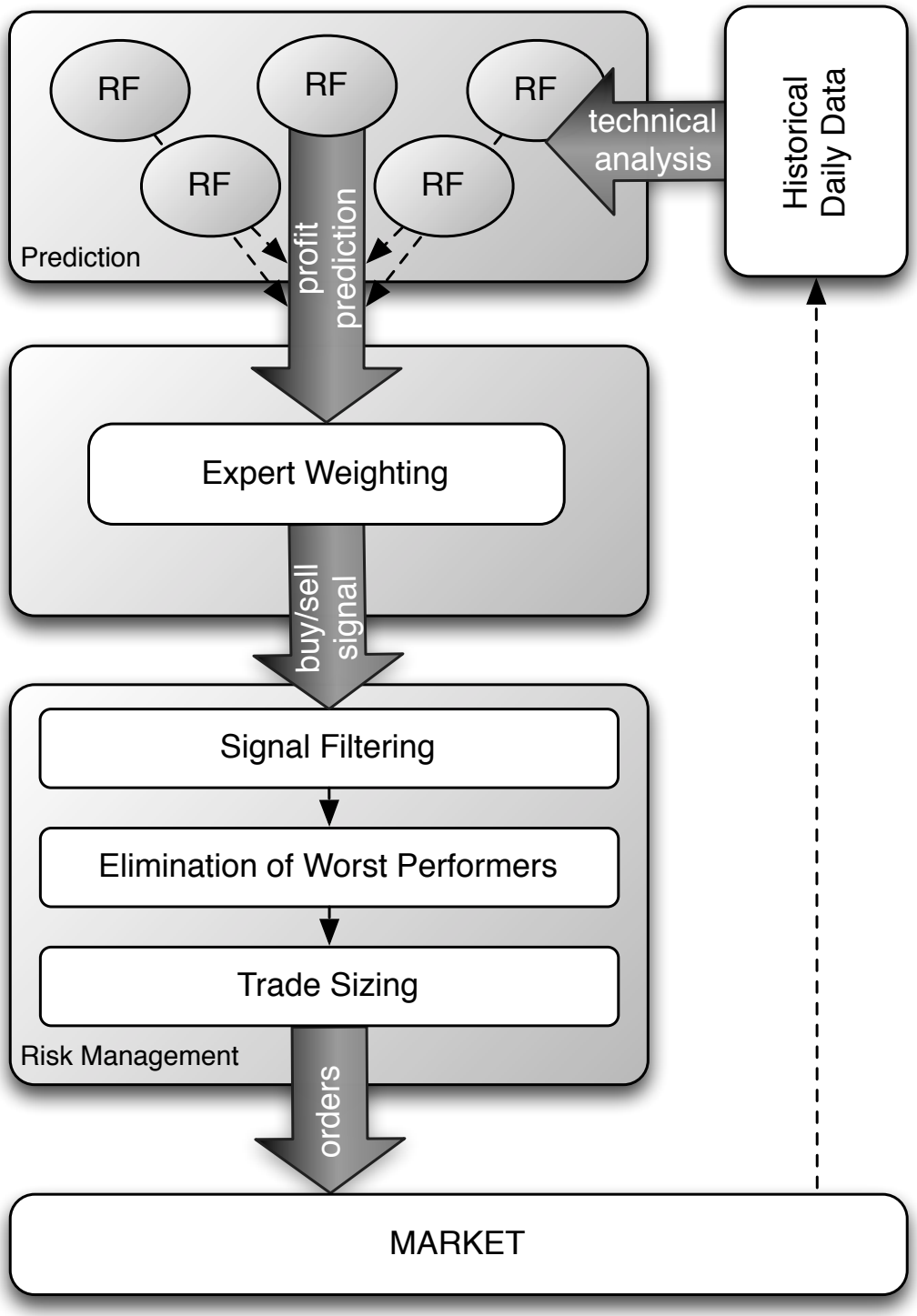


FIGURE 3.4: Diagrammatic representation of the layered workings of the expert trading system. First the inputs described in Table A.1 on page 117 are generated and used as inputs to an ensemble of random forests. The predictions of each of the forests are combined using the performance weighting method described in Section 3.3.2. Next, risk management s performed to eliminate weak signals before a trade is initiated.

3.3.2 Layer 2: Expert Weighting - An Ensemble of Ensembles

The previous section describes the process of generating and predicting with a single RF. We now improve the performance of the prediction system by creating multiple RFs and adding an online learning feature. Here, a new expert RF is trained every d days on a moving window of training data to form an ensemble of experts (RFs). The outputs of all of the RFs are then combined using expert weighting to produce a buy, sell or hold signal before passing the signal through to the risk management layer described in Section 3.3.3.

The expert weighting algorithm described in this chapter is derived from a previous algorithm suggested by Creamer and Freund (2010) and described on page 31 of Section 2.2.4. We explore two variants of an expert weighting system, the one proposed by Creamer and Freund (2010) and one introduced in this thesis, that both centre around an exponential weighting of performance.

In more detail, the historical performance of RF i at time t is denoted by $k_{i,t}$ with more weight given to experts with higher values of $k_{i,t}$. Two methods for measuring this performance are investigated:

1. The original cumulative performance measure:

$$k_{i,t} = \sum_{s=t_{i+1}}^{t} r_{i,s} \tag{3.5}$$

   where $t_1 = 0$, t_i is the time step at which expert i was added to the ensemble and $r_{i,s}$ is the abnormal return on investment for expert i at time s.

2. Our novel exponentially-weighted performance measure:

$$k_{i,t} = \lambda r_{i,t-1} + (1 - \lambda) k_{i,t-1} \tag{3.6}$$

   where $r_{i,t}$ is the abnormal return for expert i at time t and λ is a smoothing parameter that allows us to control the recency weighting of the performance measure.

As mentioned earlier, more weight is given to experts that have a higher $k_{i,t}$ as, in both cases, this represents better historical performance. To this end, we use the exponential weighting algorithm described by Creamer and Freund (2010) to generate the weights for each expert. Specifically, the weight of the first expert is given by:

$$w_{1,t} = \exp\left(\frac{k_{1,t-1}}{\sqrt{t}}\right) \tag{3.7}$$

The weight of all future experts at time t is:

$$w_{i,t} = I_i \cdot ramp(t - t_i) \cdot \exp\left(\frac{k_{i,t-1}}{\sqrt{t - t_i}}\right) \tag{3.8}$$

where I_i is the initial weight assigned to a new expert and is the mean of all of the current experts weights; $ramp(t-t_i) = \min\left(\frac{t-t_i}{t_{i+1}-t_i}, 1\right)$ brings the new expert in gradually; and t_{i+1} is the time the next expert is added.

The combined output of the ensemble of experts at time t is then the weighted average of the predictions of each expert at time t:

$$P_{g,t} = \frac{\sum_{i=1}^{\mathfrak{I}} S_{i,t} w_{i,t}}{\sum_{i=1}^{\mathfrak{I}} w_{i,t}} \tag{3.9}$$

where $S_{i,t}$ is the prediction of expert i at time t, $\mathfrak{I}$ is the total number of experts and $P_{g,t}$ is a prediction of the profit that would result from buying stock g the day before a seasonality event and selling it two days after.

3.3.3 Layer 3: Signal Filtering and Risk Management

Once a potential profit has been predicted, appropriate risk management is of vital importance to prevent large drawdowns. To this end, our algorithm evaluates the prediction, considering market factors, before making an investment decision.

To rule out weak signals we also calculate the difference between the fraction of experts that suggested buying the asset and those that suggested selling as follows:

$$\mathcal{D}_t = \text{Buy}_t - \text{Sell}_t \tag{3.10}$$

where:

$$\text{Buy}_t = \frac{\sum_{i:S_t^i>0} w_{i,t}}{\sum_i w_{i,t}} \tag{3.11}$$

and $\text{Sell}_t = 1 - \text{Buy}_t$. As the value of $\mathcal{D}_t$ is normalised between fully short (-1) and long (+1), weak signals may be eliminated by ruling out predictions where $|\mathcal{D}_t|$ falls below a threshold α_0.

Also, it cannot be assumed that predictions will be accurate for all assets at all times. Thus, we follow Creamer and Freund (2010) and monitor the performance of the algorithm on each stock g using the maximum drawdown:

$$d_{g,t} = \max(r_{g,t_x} - r_{g,t_y} | t_0 \leq t_x \leq t_y \leq t) \tag{3.12}$$

where r_{g,t_x} and r_{g,t_y} are the return of stock g from time t_0 to t_x and t_y respectively. The

maximum drawdown represents the maximum possible loss in a certain period of time and the vector of maximum drawdowns for all stocks is represented as $\mathbf{d}_t$. If $d_{g,t} < \lambda_1$ for thirty days the system hold its current position and if $d_{g,t-1} = \min(\mathbf{d}_{t-1})$, the system liquidates the position in that stock.

The final output of this layer, $\Phi_{g,t}$, is a buy or sell order whose size is determined by the magnitude of the profit prediction, $P_{g,t}$. Specifically, more is invested in stocks where a large price movement is forecast:

$$\Phi_{g,t} = \begin{cases} -1 \cdot q_{g,t} & \text{if } (P_{g,t} > 0 \wedge q_{g,t} < 0) \vee \\ & \quad (P_{g,t} < 0 \wedge q_{g,t} > 0) \\ C_t * \frac{P_{g,t}}{\sum_g |P_{g,t}|} & \text{otherwise} \end{cases} \tag{3.13}$$

where $q_{g,t}$ is the size of the current position in stock g and C_t is the total dollar wealth at time t. Thus, if the order size, $\Phi_{g,t}$ is positive, this refers to a buy order; if this is negative, it refers to a sell order.

3.4 Experiments & Empirical Results

A single experimental run proceeds as follows:

- 15 stocks are randomly selected from the DAX with equal probability.
- Features are generated from the data using the the extraction techniques outlined in section.
- The model iterates through the training set, generating three new random forest experts every 50 days each with moving windows of size 50, 100 and 200 days.
- The prediction performance of each expert is monitored and the performance weights are updated after every event according to Equation 3.8.
- When the model reaches the validation data it generates a performance weighted ensemble prediction for each event using Equation 3.9. If this prediction passes through the risk management layer then the model takes a long, short or hold position.
- At the end of the validation period the performance of the model is reported as cumulative abnormal return and Sharpe Ratio. Simulated transaction costs are incorporated in to each trade at a value of We tested our results with transaction costs of $0.003 per stock. This value is in line with the current NASDAQ pricing policy.

We use the experimental procedure outlined above to find explore the performance of a number of well studied prediction algorithms and to identify the optimal parameterisation of the models, using only the training and validation data.

3.4.1 The Base Learning Algorithm

Before exploring the performance of our performance-weighted ensemble, we first justify our choice of base regression algorithm. To this end, we compare the predictive performance of a number of well studied regression algorithms in predicting the return of stocks over a seasonality event given the inputs in Table A.1. The following predictive techniques are explored:

Linear Regression Simple ordinary least squares linear regression that attempts to generate a hyperplane that best minimizes the residual sum of squares between observed target, and the responses predicted by the model.

Regression Tree Simple CART regressions tress proposed by **?** where at each node the feature is chosen that best minimises the sum or squares error.

Multi-layer Perceptron Simple feed-forward neural network with back propagation.

Support Vector Regression (SVR) Standard epsilon-Support Vector Regression.

Random Forest Regressor Where each tree is built from a bootstrap. When splitting a node during the construction of the tree, the split that is chosen is the best split among a random subset of the features.

In order to ensure that we maximisate the generalisation ability of each model we perform a exhaustive grid-search of the hyperparameters for each model. We define the hyperparameters as any parameter that is not directly learned by a model. Specifically, we exhaustively generate all possible parameterization of each model from the hyper-parameter grids listed in Table 3.2.

The model variants are all trained on the training data and validated using the validation data. For each point in parameter space we perform 100 experiments as defined above to generate an average annualised return. The parameterisation of the best performing variant of each model type is given in Table 3.3 and it is this parameterization that is used in all experiments henceforth.

Hyper-parameter	**Description**	**Grid Values**
	Multi-layer Perceptron	
h_n	The number of hidden layers in the network	[1,2,3]
n_n^h	The number of neurones in each hidden layer	[1 .. 100]
α	The Learning rate that controls the magnitude of weight changes during training	[0, 1] intervals of 0.02
m	Momentum: the fraction of the previous weight update added to the current one	[0, 1] intervals of 0.02
	Support Vector Regression	
Kernel	The kernel type	[linear, polynomial, RBF, sigmoid]
C	The penalty parameter of the error term	[0.01, 0.05, 0.1, 0.5, 1.0, .. 1000]
γ	The kernel coefficient for RBF, polynomia and sigmoid kernels	[0.0001, 0.0005, 0.001, 0.005, .. 10]
	Random Forest	
n_{trees}	The number of trees in each forest	[2 .. 500]
m	The size of the random subset of features to consider when splitting a node	[1 .. N]

TABLE 3.2: The hyper-parameters for each models and the various paramterisations explored in the exhaustive grid search.

Model	**Parameterisation**
Multi-layer Perceptron	$h_n = 1$, $n_n^h = 31$, $\alpha = 0.32$, $m = 0.84$
Support Vector Regression	kernel: RBF, $C = 50$, $\gamma = 0.05$
Random Forest	$n_{\text{trees}} = 200$, $m = 18$

TABLE 3.3: Blah

There are only two parameters to consider for the random forests themselves: the number of trees in each forest (n_{trees}) and the size of the random subset of features to consider when splitting a node (m). For the former, we found the larger the better. However, we use $n_{\text{trees}} = 200$ as we find no improvement in performance beyond this value. Generally, lower values of m lead to a reduction of variance in random forest models but also an increase in bias.

As well as finding optimal values for the hyperparameters of the various base-layer prediction models, we use cross validation to explore values of λ in Equation 3.6. Recall that this value controls the recency bias of the performance metric for each of the experts. As above, we perform an exhaustive grid-search of the parameter space.

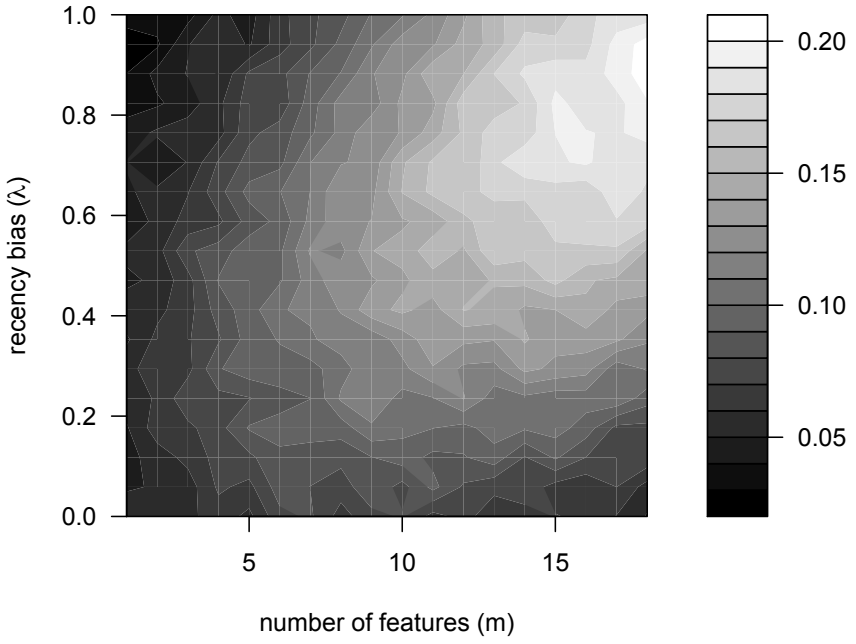


FIGURE 3.5: Heat map showing results of the cross validation gridsearch for the performance-weighted random forest ensemble.

A visual representation of the results of two dimensions of the random forest grid search is provided in Figure 3.5. It can be seen that the best parametrisation for the recency-weighted model lies in the vicinity of $\lambda = 0.90 - 0.95$ and $m = 18$. This value for m is equivalent to using all possible features for each node split, and it has been shown that such a parametrisation is often optimal for random forest regression (Liaw and Wiener, 2002). The optimal value of λ is also interesting as it is a very commonly used value for exponential moving averages and is also used by the financial risk management firm RiskMetrics^{TM} for their moving average estimation of volatility.

With the optimal parametrisation for each model identified, we explore the relative performance of each class of model. Experiments were conducted as described above and results are averaged over 50 runs for each model. We measure the mean absolute percentage error (MAPE) and the root-mean-square error (RMSE). For both the MAPE and RMSE, a lower value means better forecasting accuracy of the model. A description of these measures is given in Table 3.4. Table 3.5 shows the performance of each of the regression algorithms.

From Table 3.5 the ability of support vector regression (SVR) to tightly fit high-dimensional data is clear from its superior error statistics on the training data. However,

Statistical measure	**Description**
Annualised Return	$R_A = (\prod_{i=1}^{n}(1+r_i))^{\frac{1}{n}} - 1$
Sharpe ratio	$S = \frac{E[r_m - r_b]}{\sigma}$
Maximum drawdown	$\text{MD} = \max_{\tau \in (0,T)} \left[\max_{\tau \in (t,\tau)} X_t - X_\tau \right]$
Mean abs. % error	$\text{MAPE} = \frac{1}{n} \sum_{i=0}^{n-1} \lvert \frac{r_s - \hat{r}_s}{r_s} \rvert$
rmse	$\text{rmse} = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} (r_s - \hat{r}_s)^2}$

Where r_i is the return in year i; r_m is the model's return; r_b is return of the DAX (the benchmark); $X(t$ is the return at time t; r_s is the return of stock s over a seasonality event and $\hat{r}_s$ is the predicted return over that event.

TABLE 3.4: Description of the performance measures reported in Table 3.7

Model	**MAPE**	**RMSE**
Training Phase		
Linear regression	0.018*	0.022*
Single regression tree	0.015*	0.017*
MLNN	0.008	0.015*
SVR	0.008	0.013
Random Forest	0.009	0.013
Validation Phase		
Linear regression	0.020*	0.023*
Single regression Tree	0.019*	0.020*
MLNN	0.022*	0.023*
SVR	0.026*	0.037*
Random Forest	0.011	0.015
Test Phase		
Linear regression	0.021*	0.024*
Single regression Tree	0.019*	0.019*
MLNN	0.021*	0.024*
SVR	0.030*	0.041*
Random Forest	0.010	0.013

TABLE 3.5: A comparison of the performance of linear regression, regression trees, multi-layer feed forward neural network (MLNN), support vector regression (SVR) and random forest algorithms in predicting the return of a stock over a seasonality event. Asterisks denote a statistical significance compared to the Random Forest model of $p < 0.02$. For each dataset, the best performing algorithm is highlighted in grey.

Model		MAPE	RMSE
Training Phase			
Single RF	(*offline*)	0.009*	0.013*
Simple RF ensemble	(*offline*)	0.007*	0.009*
Simple RF ensemble	(***online***)	0.007*	0.009*
CW RF ensemble	(***online***)	0.004*	0.007*
RW RF ensemble	(***online***)	0.003	0.005
CV Phase			
Single RF	(*offline*)	0.011*	0.015*
Simple RF ensemble	(*offline*)	0.010*	0.013*
Simple RF ensemble	(***online***)	0.008*	0.010*
CW RF ensemble	(***online***)	0.004	0.007
RW RF ensemble	(***online***)	0.004	0.007
Test Phase			
Single RF	(*offline*)	0.01*	0.013*
Simple RF ensemble	(*offline*)	0.008*	0.011*
Simple RF ensemble	(***online***)	0.006*	0.010*
CW RF ensemble	(***online***)	0.004	0.008*
RW RF ensemble	(***online***)	0.004	0.006

TABLE 3.6: A comparison of the performance of various methods for generating ensemble predictions as well as a single random forest. Asterisks denote a statistical significance compared to our RW RF ensemble model of $p < 0.02$. For each dataset, the best performing algorithm is highlighted in grey. For *offline* models predictors are only created in the training phase, while for ***online*** models new predictors are added throughout all phases in an online fashion. A specific description of each model is given in Table 3.9.

in both the validation and test data the random forest (RF) produce far superior prediction accuracy; testament to its ability to generalise. Thus we confirm the suggestions laid out in Section 2.2.4, that RFs have superior out of sample performance. As such, we proceed to use the RF as the base learner for our prediction system.

3.4.2 Explorations of Ensembles

Next, the merit of ensembles and associated methodologies is explored. To this end, Table 3.6 shows the prediction performance and profitability of a single RF compared with ensembles of the models explored above, trained using the online method described at the beginning of this section. The results represent the average performance of each model over 50 experimental runs. A description of the various models is given in the first section of Table 3.9. Again, for the MAPE and RMSE, a lower output means a better the forecasting accuracy for the model.

We note that our recency-weighted RF ensemble (RW RF ensemble) outperformed all other models in terms of prediction accuracy. A clear advantage of even the simplest of ensembles can been seen when comparing the results of a single RF to the simple RF ensemble (*offline*). Combining the outputs of several random forests, even when the ensemble is only generated on the training data, reduces the out-of-sample RMSE from 0.008 to 0.006 with a statistical significance at $p < 0.02$.

It is also worth noting that moving from an offline system (where the RFs are only generated during the training phase) to an online one (where RFs are continually generated through all phases of the experiment) produces a clear advantage in out of sample data. Online learning systems clearly provide an advantage in non-stationary financial data.

Model	Annualised return	Sharpe ratio	MD
Training Phase			
Buy & Hold	0.023*	0.96*	-22.2*
Basic Seasonality	0.059*	0.67*	-18.5*
CW RF ensemble	0.14*	1.60*	-13.2*
RW RF ensemble	0.17	1.65	-12.8
CV Phase			
Buy & Hold	-0.13*	-0.63*	-30.1*
Basic Seasonality	0.061*	0.68*	-17.6*
CW RF ensemble	0.13*	1.52	-13.9
RW RF ensemble	0.14	1.60	-12.9
Test Phase			
Buy & Hold	-0.012*	-0.82*	-27.9*
Basic Seasonality	0.057*	0.68*	-18.2*
CW RF ensemble	0.12*	1.44*	-14.1
RW RF ensemble	0.15	1.64	-12.3

TABLE 3.7: A comparison of model performance metrics. Asterisks denote a statistical significance compared to the RW RF ensemble model of $p < 0.02$. The best performing model in each phases is shaded grey. A description of each model is given in Table 3.9.

3.4.3 Trading System Performance

We now proceed to test the CW RF ensemble and RW RF ensemble methods as trading systems using the trade sizing and risk management techniques proposed in Section 3.3.3. We remind the reader: this is not an attempt to identify the world's foremost trading algorithm and, as such, we do not compare the our methodology to any cited in the literature. Instead, we wish to identify a simple yet robust prediction model that is transparent enough to allow us to explore it's behaviour. Thus, we compare our method to a simple buy and hold strategy and a basic seasonality strategy (to ensure it is worthwhile) and we explore the most reliable method for weighting experts predictions.

A detailed description of the strategies used is given in Table 3.9. We measure MD, for which a lower output is preferable, a well as annualised return and Sharpe ratio, for which a higher output is better. A description of all statistical measures is given in Table 3.4. From Table 3.7, we can see that our recency-weighted random forest ensemble (RW RF ensemble) outperformed all other models in terms of both profitability and risk-adjusted return. It produces significantly higher annualised return and Sharpe ratio than the ensemble that is weighted using cumulative returns (CW RF ensemble). In out-of-sample data, our RW RF ensemble model produced an average annualised return of 15% compared with the 12% achieved by the CW RF ensemble model.

3.4.4 The Effect of Seasonality

As a matter of interest, we investigate the effect of focussing only on seasonality events by comparing the performance of the RW RF ensemble model over seasonality events only (as in the experiments reported in Table 3.7) with its performance when predicting and trading every day that the markets are open. The results of the comparison are shown in Table 3.8 where it can be seen that the RW RF ensemble model is more profitable, has greater prediction accuracy and produces small drawdowns when used only during seasonality events.

Model	**Annualised return**	**Sharpe ratio**	**MD**	**MAPE**	**RMSE**
Training Phase					
Daily	0.13	1.42	-14.5	0.005	0.008
Seasonal	0.16	1.66	-12.8	0.003	0.006
CV Phase					
Daily	0.14	1.47	-18.9	0.006	0.009
Seasonal	0.14	1.59	-13.0	0.004	0.008
Test Phase					
Daily	0.11	1.36	-23.0	0.006	0.010
Seasonal	0.15	1.63	-12.7	0.003	0.007

TABLE 3.8: A comparison of RW RF ensemble performance when trading only over seasonal events vs. trading every day. Asterisks denote a statistical significance of $p < 0.02$

3.5 Summary

In this chapter we apply performance weighted ensembles of random forests to the prediction of price returns during known seasonality events with stocks from the DAX. This approach consists of three layers: expert generation, expert weighting, and risk management. In more detail, in the first layer, random forests are continually generated and make predictions about the magnitude of stock price changes. Then, the expert weighting layer generates an overall prediction by averaging the prediction of all forests weighted based on their recent performance. Finally, the risk management layer rules out weak signals and liquidates positions in stocks that are proving difficult to predict. This approach is then benchmarked against constant-weight random forests, a solo random forest, a naïve seasonality strategy and a buy-and-hold strategy. The models are trained during a period from 2000-2008, cross-validated from 2008-2010 and tested out-of-sample from 2010-2012. We also explore and compare two variants for expert weighting: cumulative performance of an expert and recency weighted performance.

In out-of-sample trading simulations, both forms of performance-weighted random forest ensembles outperform all other models in terms of both prediction accuracy and profitability. At the same time, we found that, using a recency weighting of experts' performance, as opposed to the cumulative performance, resulted in over a 10% improvement in risk adjusted return, as well as decreased drawdowns and prediction error. This demonstrates that the ability of the online-generated ensemble to capture different phases of the market allow both approaches of random forest ensemble to excel in forecasting compared with static ensemble approaches. This is because using a moving average of performance to compare experts, allows the prediction algorithm to adjust more swiftly to changing market conditions and hence improve prediction accuracy and profitability.

In addition to the increased performance, the constant training of new experts for the performance weighted random forest ensemble approach has a number of other advantages. First of all, new experts can be added rapidly with minimal training time, a very desirable quality for a quantitative trading application. Secondly, each random forest, and thus the entire ensemble, exists as a human readable decision tree. Thus, our strategy is not a 'black box' and, as such, analysis and simulation can give managers and regulators insight into the risks involved in such a trading strategy. Specifically, the human readable nature of the random forest ensemble ensures that, at any given time, a risk manager is able to ascertain exactly what the system will do and what decisions it will make. That said, the complexity of an RF ensemble of this size would certainly affect its opacity.

Model	**Description**
Buy & Hold	A long position in the stocks is taken at the beginning of the test period and closed out at the end.
Basic Seasonality	Positions are taken in line with the empirical regularities discussed in section 3.2, e.g., always take a short position in a stock over a weekend.
Single RF	A single random forest with 200 trees and $m = 17$ is used to predict the return of a stock over a seasonality event.
Simple RF ensemble (*offline*)	An ensemble of random forests is generated by training a new random forest every 50 days *throughout the training phase only*. A prediction for the return of a stock is generated by taking an average of the predictions of all of the random forests in the ensemble.
Simple RF ensemble (***online***)	An ensemble of random forests is generated in an online fashion by training a new random forest every 50 days *throughout all phases of the experiment*. A prediction for the return of a stock is generated by taking an average of the predictions of all of the random forests in the ensemble.
PW RF ensemble	Performance-Weighted Random Forest ensemble. As above except that the performance of each random forest in the ensemble is monitored as its cumulative return, as described in equation 3.7, such that the final prediction of the ensemble is a performance-weighted average of the experts.
RW RF ensemble	Recency-Weighted Random Forest ensemble. As above, except that, instead of the cumulative return, an exponential moving average of returns, as in equation 3.8, for each expert is used for the performance weighted-prediction.

TABLE 3.9: A description of the models whose results are described in Table 3.7

Chapter 4

Predicting Equity Market Price Impact Using Market Depth Data

The previous chapter explored the use of online ensemble learning methods for predicting daily financial returns. In this chapter, we focus on our target domain, the order book, by applying methods honed in the previous chapter to high-frequency limit order book (LOB) data. Now, although both daily and tick-level data series are known for their non-normality and non-stationarity, high-frequency data is known to have even fatter tailed distributions, be highly non-stationary and exhibit clustered volatility. Each of these traits poses a serious challenge for making predictions in this environment.

As discussed in Section 2.4.1, there is a growing need for an understanding of the impact of trading in limit order books. To address this challenge, this chapter presents an empirical model for predicting the short term price impact in LOBs of events that alter the best available prices in the book. Such events include any market orders or limit orders at the current best prices, as well as cancelations that remove all volume at the best quoted price. Henceforth, we refer to such situations simply as "events". Specifically, we develop a model, based on the performance-weighted methodology proposed in Chapter 3, to forecast the relative change in price 1, 5, 10, 60 and 600 seconds after an event. The model is trained and tested on 100 days of full depth of book data from the BATS Chi-X exchange. The model is later used as a proxy for assessing the importance of various parameters in predicting price impact.

While other studies have investigated the predictive power of more tradition regression techniques, currently, no study has explored the use of performance-weighted ensemble to predict the short term price impact of order book events (e.g. see Section 2.4). We demonstrate that an ensemble of recency-biased, performance-weighted random forests is able to predict the price impact of events more consistently and with greater accuracy than linear regression, neural networks, support vector regression as single algorithms or combined as ensembles. More interestingly, we propose a novel method for identifying

the importance of features in our model and find that increases in the volume at the best price and the number of recent order arrivals is found to cause a decrease in price impact, while increases in the average price increment, spread and event size cause an increase.

Against the background of the literature described in Section 2.4, we propose an adaptation to the system specified in Chapter 3 for the application of a performance-weighted ensemble of random forests to predicting price impact. Specifically, we address the potential overfitting problem common to financial data by using random forests and tackle the non-stationary element of data by generating random forest in an online fashion.

The chapter is structured as follows. Section 4.1 describes the data used in the chapter, Section 4.2 introduces the features and the feature selection algorithm used to refine the inputs to the model while Section 4.3 describes the prediction algorithm itself. In Sections 4.4 and 4.5 the experiments are described and results summarised while Section 4.6 gives concluding remarks.

4.1 The Data

The analysis in this chapter is based on historical depth-of-book data from the BATS Chi-X exchange using the 25 most actively traded stocks over 100 days of trading from 12$^{\text{th}}$ February 2013 to 3$^{\text{rd}}$ July 2013. This choice of market is due to the difficulty and expense of obtaining level 2 data. However, we believe that this particular choice of market is insignificant and, given the similarity of financial time series across markets and asset classes, we believe that our results could be verified in other markets. For a more detailed breakdown of the market share of BATS and Chi-X see Section B.1 of the Appendix.

Although 100 days of data may, at first, not seem significant, the high-frequency nature of today's limit order markets must not be overlooked. For the top 5 most liquid stocks, it is not uncommon to see over $1,000,000$ order book events in a given day. Thus, for all stocks considered in this study, we have tens of millions of data points available for training, validation and testing.

Prior to conducting experiments, the data is split into training, validation and test sets as follows: 60 days training, 20 days validation and 20 days testing. It should be noted that no parameters are tuned using the test set. This set is used only to report the performance of the model found to perform best on the validation data.

We consider only the regular trading time between 9:30 - 16:00, and all other periods are discarded. This raw data contains details on all actions in the book, that is: order arrivals, executions, modifications and cancellations, with each item timestamped to the millisecond. We use the term "event" for any action the modifies the best bid or

Symbol	Event Description
LO^0	A limit order at the current best price
LO'	A limit order within the best prices
MO^0	A market order whose volume $<$ outstanding volume at the best price
MO'	A market order whose volume $\geq$ outstanding volume at the best price
CA^0	A cancellation within the best price queue that only affects the volume available
CA'	A cancelation at the best price that removes all available volume and changes the best price
MN	A modification of an existing order that alters the volume available at the best price

TABLE 4.1: Summary of the seven possible types of event along with the corresponding symbol.

Statistic	$P(LO^0)$	$P(LO')$	$P(MO^0)$	$P(MO')$	$P(CA^0)$	$P(CA')$	P(MN)
Mean	0.451	0.082	0.046	0.037	0.344	0.042	0.009
SD	0.102	0.087	0.006	0.033	0.069	0.051	0.007

TABLE 4.2: Summary statistics for the stocks, showing the probability of each of the events described in Table 4.1. The first row shows the probability of each event happening averaged across all 25 stocks. The second row shows the standard deviation of these probabilities.

ask prices or the volumes quoted at these prices. We do not attempt to describe any potential impact of actions that occur deeper in the LOB. Moreover, we are mindful that liquidity is fragmented across many exchanges and that the stocks in this study are traded on many other platforms. This trading activity will also affect the impact of the events we observe.

Specifically we define seven events that cause changes at the best price in a LOB and a definition of the these events along with their notation is provided in Table 4.1.

Table 4.2 provides a summary of the probability of the different events and some basic statistics. The standard deviation values show that there is a fairly large deviation in the probabilities across different stocks. In fact, we find that for stocks whose spread is almost always equal to one tick there is a relatively low probability of events that change the best price (approximately 4%). With stocks whose spread is typically greater than one tick, we find that this probability is much greater (up to 25%). Thus, the probability of theses events varies greatly between stocks and we are conscious of this throughout the remainder of our study, training a separate model for each stock.

4.2 Feature Selection

In order to make meaningful predictions we wish to capture the state of the LOB at a given time with a number of features. These features then form the input to the prediction system. There is a far richer variety of information available in LOBs than with daily stock price data, including data on: order sizes, arrival times, cancelations and modifications. Given this abundance of information, we propose the exploration of a great number of features that can be broadly divided into three categories:

Incoming event This group of features describes the attributes of incoming events: its type (P(LO^0),P(MO^0)), etc.) and its size, side and price.

Price These features provide information about the log-normalised best bid/ask price series. This includes various technical analysis indicators such as: exponential moving averages (EMAs) of the last n prices before a trade, price oscillators and the relative strength index (RSI) over the last N price changes.

Spread These features aim to provide information about relative changes in the bid/ask spread. Again, many technical analysis indicators are used here, with time series data normalised by the minimum allowable price increment for a particular stock.

Liquidity These features contain information about the apparent liquidity of the book. E.g. relative depth of each side of the book, order arrival and cancelation rates, and modal order price relative to best bid/ask price.

Clearly, not all features will have the same predictive power. Thus, in the interests of dimensionality reduction and computation, we wish to eliminate those features that have little or no impact on the performance of our random forest based prediction model. One simple method for selecting features is to choose the subset of features that is most highly correlated to the target variable. However, this is likely to generate a highly collinear feature space and impair the performance of the learning algorithm. As a result, we use a method of feature importance ranking, first suggested by Breiman (2001), to eliminate irrelevant variables.

Specifically, we use the same feature elimination method described in Algorithm 5 in the previous chapter. Recall from Section 3.2.3 that a single random forest is trained on the training portion of the data so as to calculate the relative importance of each feature j, $\mathcal{VI}_j$, as follows:

$$\mathcal{VI}_j = \frac{\sum_{\theta=1}^{\Theta} (e_{\theta,j} - e_{\theta,\pi j})}{\Theta \cdot \hat{\sigma}} \tag{4.1}$$

where Θ is the number of trees in the forest, $e_{\theta,j}$ is the RMSE of tree θ without permuting j, $e_{\theta,\pi j}$ is the RMSE of that tree after permutation of j, and $\hat{\sigma}$ is the standard deviation

of the differences between $e_{\theta,j}$ and $e_{\theta,\pi j}$. Features that produce larger $\mathcal{VI}$ values are considered more important than features that produce smaller values.

Thus, for feature selection, we use the backwards elimination method described in Algorithm 5. The algorithm was applied to the training portion of the data and a plot of the RMSE at each stage of the elimination is shown in Figure 4.1. As this figure shows, there is a slight initial increase in performance (decrease in RMSE) as features are eliminated. Following this, there is a swift decline in performance as features that are essential for prediction are eliminated (from right to left). The algorithm yields the optimal set of 80 features listed in Table B.3 in Appendix B.

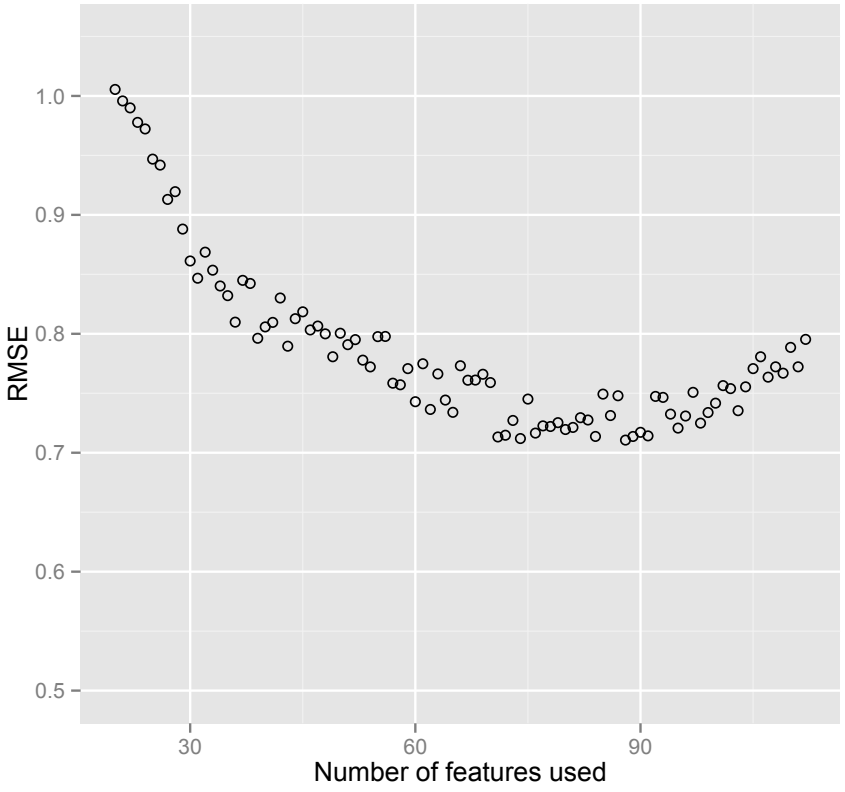


FIGURE 4.1: Plot of RMSE for each round of the feature elimination algorithm. It can be seen that, as features are removed, there is an initial, slight improvement in performance before a rapid decline.

4.3 The Model

This section describes an adaptation of the predictive system proposed in Chapter 3 for predicting daily changes in stock prices. Our modified system consists of an ensemble of models, that is used to predict the price impact of events that alter the best bid/ask prices. The price impact is measured by taking the price changes at 1, 5, 10, 60 and 600 seconds after the event normalised by the pre-event price. A separate ensemble is used

for the prediction of each time interval. An overview of the model structure is given in Figure 4.2.

To that end, the prediction system described in this section is based on an ensemble of random forest regressors that are, as before, referred to as experts. Specifically, every fifteen minutes, three new experts are trained on the previous 1, 2 and 3 hours of data and added to the ensemble in an online fashion. That is, new experts are trained and introduced into the ensemble continuously, even during out of sample tests. The ensemble is capped to a maximum number of experts, $\mathfrak{I}_{max}$, and once the maximum number is reached the poorest performing expert is dropped upon each new addition.

4.3.1 The Base Learner

Each expert in the ensemble is represented by a random forest regression algorithm. In this chapter, a random forest is an ensemble of many regression trees designed to produce accurate predictions without overfitting the training data (Breiman, 2001). We use the same method laid out in Section 3.3.1. In the rest of this chapter, this prediction is denoted by $S_{i,t}$, where i is the random forest, and t is the time of the observation.

The features given in Table B.3 form the input to the random forest prediction algorithm. In our study, the goal of the random forest is to generate a function of the features that predicts the relative price change at particular intervals after an order book event that alters the best bid/ask price.

4.3.2 An Ensemble of Ensembles

The previous section describes the process of training and predicting with a single (random forest) expert. In order to improve performance of the prediction system, we explore the method of online ensemble generation proposed in the previous chapter. Here, three experts are trained every fifteen minutes on a moving window of 1, 2 and 3 hours of training data to generate an ensemble of experts. The outputs of all of the experts are then combined using an expert weighting algorithm to generate a prediction about the price change after an event.

The expert weighting method used in this chapter is an adaptation of that laid out in Chapter 3. Specifically, instead of defining the historical performance $k_{i,t}$ of expert i at time t as an exponential moving average of returns ($r_{i,t}$), we instead define it as an exponential moving average of the reciprocal of that experts RMSE ($\frac{1}{\eta_{i,t}}$). This maintains the ability to assign more weight to experts with larger values of $k_{i,t}$. We now define $k_{i,t}$ as:

$$k_{i,t} = \lambda/\eta_{i,t-1} + (1-\lambda)k_{i,t-1} \tag{4.2}$$

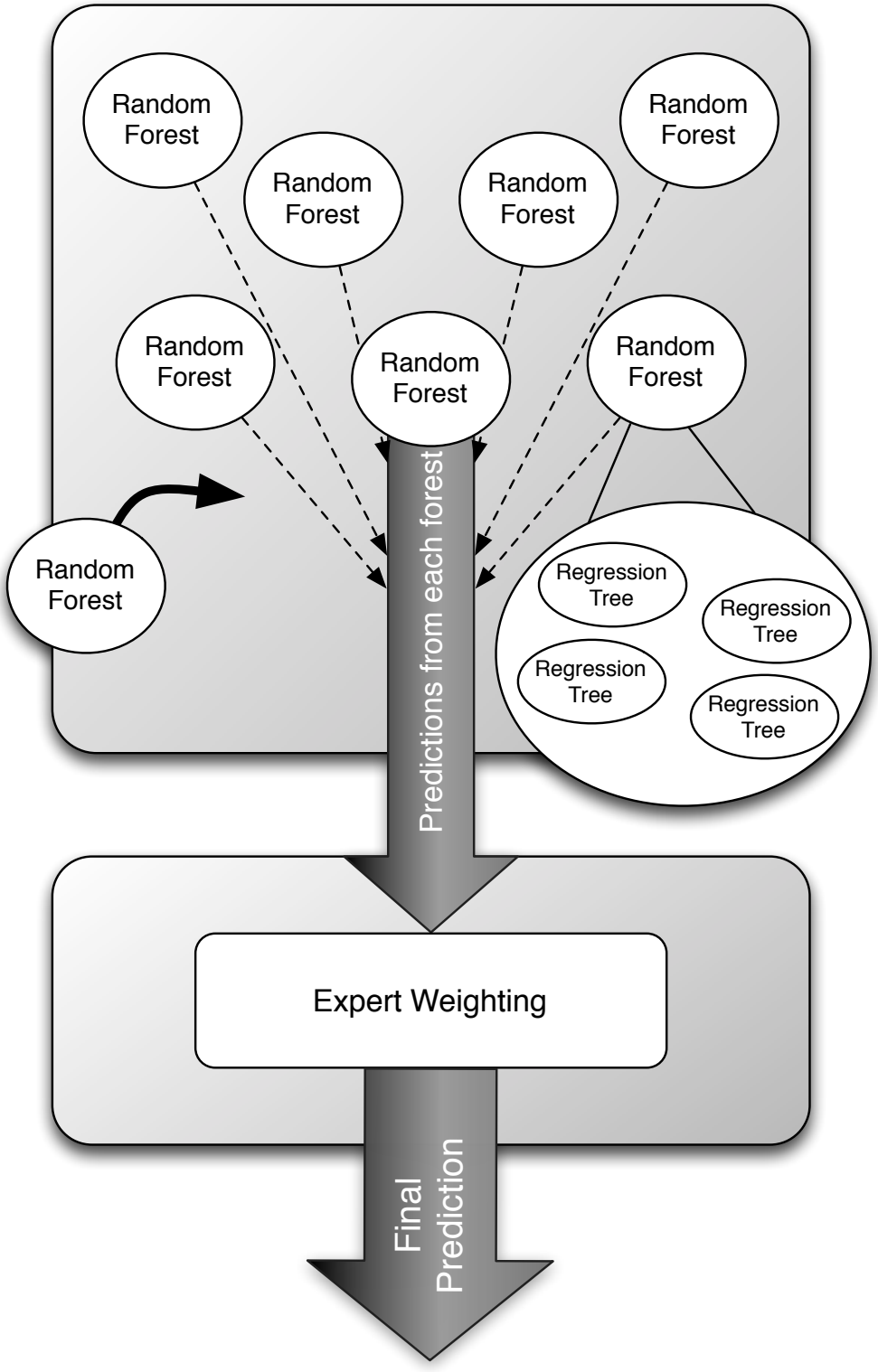


FIGURE 4.2: An overview of the prediction system. The inputs given in Table B.3 are used to train three new random forests (experts) every fifteen minutes. Each expert makes a prediction about the price change after an event and the expert weighting layer aggregates these predictions using the method described in Section 4.3.2.

where $\eta_{i,t-1}$ is the RMSE of the last prediction made by expert i at time $t-1$ and λ is a smoothing parameter that allows us to control the recency weighting of the performance measure.

As mentioned previously, a greater weight is given to experts that have a higher $k_{i,t}$ as this represents better historical performance. As such, we use the exponential weighting algorithm described in Chapter 3 for generating weights for each expert. In detail, the weight of the expert trained first is given by:

$$w_{1,t} = \exp\left(\frac{k_{1,t-1}}{\sqrt{t}}\right) \tag{4.3}$$

The weight of all following experts is then defined as:

$$w_{i,t} = I_i \cdot ramp(t - t_i) \cdot \exp\left(\frac{k_{i,t-1}}{\sqrt{t - t_i}}\right) \tag{4.4}$$

where I_i is the weight assigned to a newly generated expert and is the mean of all current experts' weights; $ramp(t - t_i) = \min\left(\frac{t-t_i}{t_{i+1}-t_i}, 1\right)$ allows new experts to be brought in cautiously; and t_{i+1} is the time at which the next expert will be added.

The combined output of the ensemble of random forests at time t is then a weighted average of the predictions of all expert at time t:

$$P_{g,t} = \frac{\sum_{i=1}^{\mathfrak{I}} S_{i,t} w_{i,t}}{\sum_{i=1}^{\mathfrak{I}} w_{i,t}} \tag{4.5}$$

where $S_{i,t}$ is the prediction of expert i at time t, $\mathfrak{I}$ is the total number of experts at time t and $P_{g,t}$ is a prediction of the relative price change.

4.4 Experiments

The experimental procedure outlined in Algorithm 6 is used to determine the optimal parameterisation of our model, as well as the benchmarks, using only the training and validation data sets. Once the optimal parameterisations are found, the performance on test set is reported. For the RFs themselves, there are only two parameters of interest: the number of trees grown in each forest (n_{trees}) and the number of features randomly chosen when splitting a node (m). It is generally thought that lower values of m lead to a reduction of variance but also an increase in bias. We thus use cross validation to find optimal values for this parameter.

In addition to finding optimal values for m, cross validation is used to explore values of the recency bias, λ, for the performance metric described in Equation 4.2. To asses the

Algorithm 6 The experimental procedure

```
for all parameter settings do
    for all stocks do
        Generate features from data as described in Section 3.1.
        Iterate through training set generating three new experts every 15 minutes
            with a moving windows of data sized 1, 2 and 3 hours.
        Monitor the prediction performance of each expert, updating the weights after
            every event according to Equation (4.4).
        Upon reaching the CV data, an ensemble prediction is made for each
            event using Equation 4.5.
        At the end of the CV period, store the average RMSE of prediction across
            the CV set for current stock.
    end for
    CV performance is reported as the average RMSE of the model across all
            25 stocks.
end for
The model with the best CV performance for each stock is kept and run on the
corresponding stock over the test set.
```

performance of these variables we perform an exhaustive grid-search of the parameter space. The experimental procedure above is performed for each point in the parameter space to generate an average RMSE across all stocks for each parameterisation.

As the benefits of this method of expert weighting have been demonstrated in Chapter 3 we explore the effectiveness of random forests (RFs) as the basis of the prediction system. To this end, we compare the predictive performance of the system described above with a multitude of base learners including: random forests, ordinary least squares regression, multi-layer feed-forward neural networks with backpropagation and support vector regression (SVR). The parameters of each of the base learning algorithms optimised using multi-dimensional grid search on the validation data sets. Results are reported as the performance on the previously unused test set.

4.5 Results

In this section we begin by tuning the parameters of the model proposed in Section 4.3 for both the BATS and the Chi-X exchange before exploring the performance of the RF base learner and the advantages of ensemble methods. We go on to assess the performance of our trading system compared to buy-and-hold, basic seasonality and the current state-of-the art. Finally we explore which features the model finds to be most important in predicting price impact. Unless otherwise stated, reported results are averaged across the two exchanges.

Model	Parameterisation
Multi-layer Perceptron	$h_n = 1$, $n_n^h = 25$, $\alpha = 0.39$, $m = 0.90$
Support Vector Regression	kernel: Gaussian, $C = 50$, $\gamma = 0.1$
Random Forest	$n_{\text{trees}} = 250$, $m = 78$

TABLE 4.3: Blah

4.5.1 Parameter Grid Search

As in Section 3.4, we perform an exhaustive grid-search of the parameter sets listed in Table 3.2. The parameter grid-search yielded the same values for both the BATS and Chi-X data and the results are shown in Table 4.3.

Figure 4.3. It can be seen that the best parametrisation for the recency-weighted model lies in the vicinity of $\lambda = 0.85$ and $m = 78$. This value for m is equivalent to using all possible features for each node split, and it has been shown that such a parametrisation is often optimal for random forest regression (Liaw and Wiener, 2002). A value of $\lambda = 0.88$ is also interesting as it is a very commonly used value for exponential moving averages.

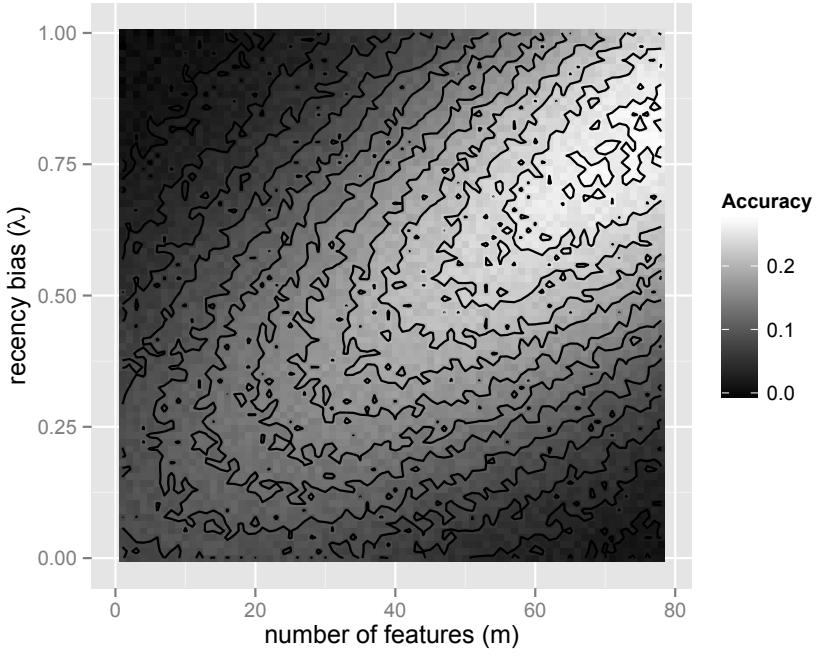


FIGURE 4.3: Heat map showing results performance of our ensemble model over the gridsearch space. Higher values represent better performance.

Model	Chi-X RMSE at $t =$					BATS RMSE at $t =$				
	1	5	10	60	600	1	5	10	60	600
Training Phase										
Linear regression	0.24*	0.30*	0.32*	0.48*	0.49	0.26*	0.31*	0.33*	0.49*	0.50
Neural Network	0.19*	0.23*	0.28	0.28*	0.47	0.20*	0.24*	0.29	0.32	0.48
SVR	0.19*	0.24*	0.26*	0.32*	0.45	0.20*	0.26*	0.27*	0.32	0.46
Single RF	0.21	0.27	0.30	0.33	0.42	0.23	0.28	0.30	0.32	0.42
CV Phase										
Linear regression	0.30*	0.36*	0.38*	0.47*	0.51*	0.30*	0.37*	0.40*	0.48*	0.50*
Neural Network	0.26*	0.35*	0.37*	0.49*	0.51*	0.27*	0.38*	0.37*	0.48*	0.53*
SVR	0.29*	0.34*	0.38*	0.46*	0.48	0.30*	0.34*	0.42*	0.47*	0.50
Single RF	0.23	0.27	0.32	0.35	0.43	0.24	0.28	0.32	0.34	0.44
Test Phase										
Linear regression	0.30*	0.42*	0.40*	0.50*	0.51*	0.29*	0.41*	0.42*	0.51*	0.49*
Neural Network	0.31*	0.43*	0.44*	0.48*	0.50*	0.32*	0.41*	0.47*	0.50*	0.53*
SVR	0.28*	0.37*	0.45*	0.45*	0.50*	0.31*	0.38*	0.45*	0.43*	0.52*
Single RF	0.18	0.31	0.32	0.34	0.44	0.19	0.30	0.30	0.35	0.47

TABLE 4.4: A comparison of the performance of various regression algorithms in predicting the relative change in asset price $t = 1, 5, 10, 60$ and 600 after an event. Asterisks signify a statistical significance compared to the RF algorithm.

4.5.2 The Base Learner

As in Chapter 3, before exploring the performance of our proposed ensemble predictor, we first investigate the validity of our choice of the RF as a base learning algorithm. Accordingly, Table 4.4 displays the performance of a number of regression algorithms in predicting the price of stocks at various time points after the best-price-altering events listed in Table 4.1 using the inputs in Table B.3. Experiments were conducted as described above and results report the root-mean-square error (RMSE) averaged across the 25 stocks.

It is clear from Table 4.4 that, in out-of-sample test data, the random forest (RF) algorithm outperforms the alternative regression models in terms of minimising root-mean-square error (RMSE). Upon closer inspection, it is evident the ability of the neural network and SVR algorithms to accurately describe that training data leads to superior performance on the training set. However, there is evidence of overfitting in both of these algorithms due to their poor performance in both the cross validation (CV) and test sets. Also, we note that the performance of all regression algorithms is very similar for both the BATS and Chi-X data. Thus, from this point forward we will present results averaged across the two datasets.

Model	RMSE at $t =$				
	1	5	10	60	600
Training Phase					
Single RF	0.22*	0.28*	0.29	0.33	0.43
Linear regression ensemble	0.18*	0.22*	0.26	0.34*	0.49*
Neural Network ensemble	0.13*	0.19*	0.23*	0.31*	0.41*
SVR ensemble	0.13*	0.18*	0.20*	0.31*	0.40*
RF ensemble	0.15	0.20	0.26	0.32	0.40
CV Phase					
Single RF	0.23*	0.28*	0.31*	0.35	0.44
Linear regression ensemble	0.21*	0.26*	0.32*	0.38*	0.50*
Neural network ensemble	0.22*	0.29*	0.31*	0.40*	0.45*
SVR ensemble	0.24*	0.29*	0.35*	0.36*	0.42*
RF ensemble	0.15	0.22	0.25	0.33	0.40
Test Phase					
Single RF	0.19*	0.30*	0.31*	0.35	0.46*
Linear regression ensemble	0.24*	0.27*	0.31*	0.42*	0.59*
Neural network ensemble	0.24*	0.28*	0.31*	0.37*	0.50*
SVR ensemble	0.40*	0.39*	0.39*	0.45*	0.47*
RF ensemble	0.15	0.23	0.24	0.34	0.40

TABLE 4.5: A comparison of the performance of various ensembles of regression models and a single random forests algorithm in predicting the relative change in asset price $t = 1, 5, 10, 60$ and 600 after an event. Asterisks signify a statistical significance compared to our model of $p < 0.05$. Outputs of all regression algorithms are combines as described in section 4.3.2

4.5.3 Ensembles

The previous chapter demonstrated the gain in predictive power achieved by moving from single RFs to ensembles as well as the advantage of online model generation. Thus, in this section we explore the power of ensembles of models versus single models and the predictive performance of ensembles of RFs versus ensembles of linear regression models, neural networks and support vector machines.

Table 4.5 shows the results of the comparative performance of the various base learners outlined above in predicting the price of stocks at various time points after best-price-altering events. The inputs for all algorithms are those listed in Table B.3. Experiments were performed as described above with results averaged across all stocks.

It can be seen that our recency-weighted random forest (RF) ensemble outperforms all other models on out-of-sample data. While the random forest based ensemble produces higher error values on the training data, the proven ability of random forests to avoid

overfitting allows the ensemble of random forests to produce significantly superior out of sample results to all other models on 4 out of 5 time intervals. A distinct advantage of online ensemble generation can be seen when comparing the results of a single random forest to the ensemble of random forest. Introducing online training of experts improves the performance of the model across all 5 time periods, with 4 of 5 results proving significant at $p = 0.05$

4.5.4 Comparative influence of features

The power of random forests lies not only in their ability to make predictions with little overfitting but also in their relative transparency. Thus, this section seeks to extract knowledge about the comparative influence of the considered features. Specifically, we explore the relative importance and direction of influence of each of the features in order to gain an increased understanding of the price formation process.

4.5.4.1 Feature importance

Given the effectiveness of our model as demonstrated in the previous section, we now explore which features are found to be most important in predicting the price impact of events. To this end, we apply Algorithm 5, our feature selection algorithm, described in Section 3.2.3 of Chapter 3 to our ensemble learning model to gain insight into the which features are most influential in forming a prediction.

In detail, each time a new RF is trained (every 15 mins in our experiments) Algorithm 5 (see section 4.2) is applied when training the RF to generate a relative importance ranking for the features of that RF. At the end of the test period, we calculate the importance of each feature by taking a weighted arithmetic mean of its importance ranking in each of the forests in the ensemble using the same weights used for making predictions and described in Equations (4.3) and (4.4). The importance of feature j for the entire ensemble model, $\mathbf{Vi}_j$, is thus calculated as follows:

$$\mathbf{Vi}_j = \frac{\sum_{i=1}^{\mathfrak{I}} w_{i,t} \mathcal{VI}_{i,j}}{\sum_{i=1}^{\mathfrak{I}} w_{i,t}} \tag{4.6}$$

Where $\mathcal{VI}_{i,j}$ is the importance of feature j for RF i and is calculated using Equation 4.1 and $w_{i,t}$ is the current weight assigned to RF i, as calculated by Equations (4.3) and (4.4).

We use Equation 4.6 to rank the relative importance of the features at the end of the test period and the top five highest ranked features are given in Table 4.6

To explore how the relative importance of the features changes through time, we calculate the importance of the variables, as above, at the end of each day of the training,

Feature	Rank
Event size	1
EMA of spread	2
EMA of volume at best price	3
Mean price increment between order prices	4
Number of quotes arrived in the last n observations	5

TABLE 4.6: The five most important features at the end of the test period as ranked by Equation (4.6).

Feature	Mean rank	SD rank
Event size	2.3	1.9
EMA of volume at best price	4.5	3.2
Mean price increment between order prices	4.8	3.6
EMA of spread	5.6	4.4
Event type	7.5	5.8

TABLE 4.7: The five most important features averaged across the entire training, validation and test period. The importances are calculated at the end of each day. Both the mean and standard deviation are provided for comparison.

validation and test periods. The results of this investigation are presented in Table 4.7 where we can see the mean and standard deviation of the relative importance of the five highest ranked features.

4.5.4.2 Direction of feature influence

As well as exploring the relative importance of the features, we aim to identify the directional relation to the response variable. For this purpose we propose a method which we shall refer to as feature partition response differencing (FPRD).

The basic idea of FPRD is that the direction of influence of a feature can be determined by comparing the differences between the response variable (price impact in our case) values for each of the two subspaces created by at the nodes of the regression trees. That is, for each node that contains feature j in all trees of all forests in our model, we separately calculate the mean price impact for all data that falls into the low j partition (where a data-point's value for j is less than and the threshold determined by a particular node)and for the high j partition. We then calculate the percentage difference of these means relative to the low j mean before averaging all of these differences across all nodes in our model that pertain to feature j. A single value is then obtained for each feature that describes the relative directional relationship with the price impact. Formally, we

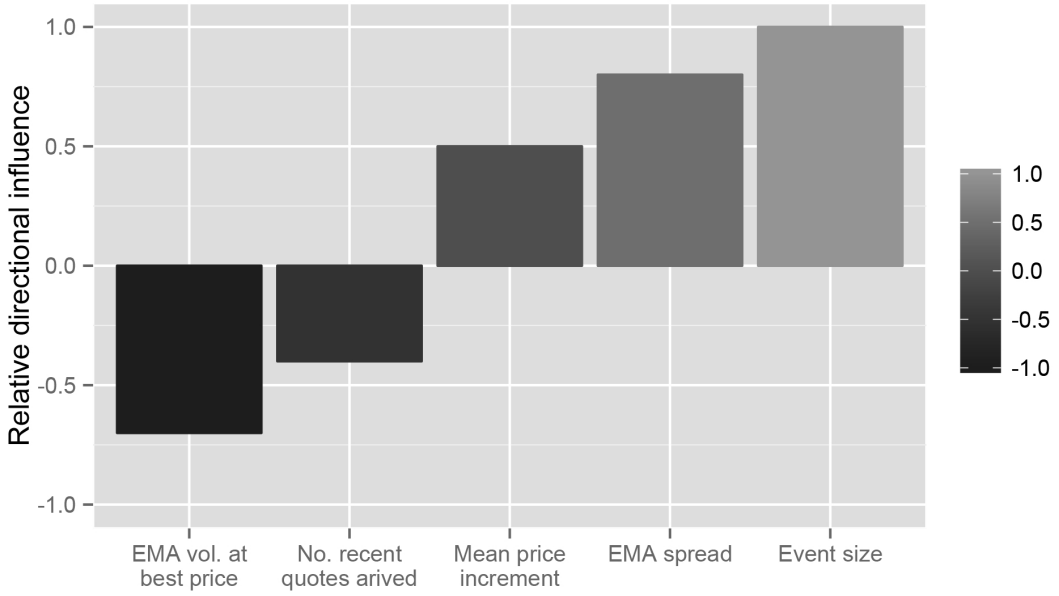


FIGURE 4.4: The relative directional influence (FPRD) of the five most important features for price impact calculated using Algorithm 7.

calculate the FPRD using Algorithm 7

Algorithm 7 The feature partition response differencing algorithm.

1: **for all** Random Forests **do**
2: **for all** Trees in Forest **do**
3: Calculate:
4:

$$i_{r,t} = \frac{\frac{1}{n}\sum_{P^k:P^{k+}}}{\frac{1}{n}\sum_{P^k:P^{k-}}} - 1$$

where P^{k+} is the set of all price impacts in the higher partition
5: and P^{k-} those that fall into the lower partition.
6: **end for**
7: **end for**
8: Calculate the FPRD as the average of the difference over all trees in all forests.

To compare the directional influence of the five features found to be most important in Table 4.6, we perform FPRD for each feature before normalising across each of the percentage differences. The output is presented in Figure 4.4. The output from the FPRD and the bar chart in Figure 4.4 the magnitude of the relative influence aligns with our findings in Section 4.5.4.1. We also find that increases in the volume at the best price and the number of recent order arrivals is found to cause a decrease in price impact, while increases in the average price increment, spread and event size cause an increase [1].

[1] An obvious question concerns whether indications of relative directional importance provide similar information as the true relationships between variables. This has been confirmed using artificial data.

4.6 Summary

In this chapter we apply performance weighted ensembles of random forests to the prediction of the price impact of order book events. In more detail, random forest regressors are continually generated, at fixed time intervals, and added to an ensemble of such experts. Upon the occurrence of an event that alters the best prices of the LOB (limit order at the best price, market order, cancellation or modification), each expert makes a prediction and an expert weighting system averages the predictions of all experts weighted by their recent performance. A separate ensemble is used to predict the price change at $t = 1, 5, 10,$ and 60 seconds after an event. The performance of random forest ensembles is then benchmarked against ensembles of linear regression, neural networks and support vector models as well as single random forests. The models are trained using 60 days of data, cross-validated using 20 and tested using 20. Reported results are an average over the 25 most liquid stocks from the BATS Chi-X exchange.

In out-of-sample simulations, the performance-weighted random forest ensemble outperforms all other models across all time intervals. Specifically, our model outperforms all other models by around 20% when predicting the price change at 1, 5, and 10 seconds. Above this, performance of all models fall substantially. This demonstrates the ability of the online-generated ensemble to make predictions in the highly non-stationary order book environment.

In addition, we identified the primary drivers of market impact which conform to and supplement the existing literature in the field. Firstly, we find the most dominant feature to be the size of an incoming event and this is well documented throughout the microstructure literature (Lillo et al., 2003; Eisler et al., 2012; Potters and Bouchaud, 2003; Plerou et al., 2002). Secondly, we see both the volume at the best prices and the mean price increment to ply an important role in the determination of market impact. Specifically, we see that higher volumes of orders at the best price on one side of the book significantly dampens market impact and that smaller average increments between passive limit orders have the same effect - this supports more recent empirical and simulations Farmer et al. (2013). Additionally, we found that the rate of arrival or new orders to the book had a significant impact on price impact though we are unable to identify any literature that supports this.

For those in the trading industry, these results should be of particular interest. Volumes at the best prices, quote arrival rate, and mean price increments of orders should be closely monitored for abnormalities or large swings that may lead to liquidity vacuums or, alternatively, buffer the book and provide a cost effective period for unloading orders and minimizing price impact.

Chapter 5

An Agent-Based Order Book Model for Automated Trading Algorithms

Previous chapters have explored the use of machine learning algorithms in forecasting price movements and impact in limit order books. Such techniques are useful in understanding the codependencies of LOB features and as inputs for optimal trade execution algorithms. However, the regulators require more than this.

As Chapter 1 describes in detail, the all-too-common flash crashes are a dramatic consequence of the growing complexity of modern financial markets and have not gone unnoticed. In November 2011, the European Union (2011) made proposals for a revision of the Markets in Financial Instruments Directive (MiFID). Although this directive only governs the European markets, according to the World Bank (2012) (in terms of market capitalisation), the EU represents a market capitalisation around two thirds of the size of the US. In the face of declining investor confidence and rapidly changing markets, a draft of MiFID II was produced. After nearly three years of debate [1], on the 14$^{\text{th}}$ January 2014, the European Parliament and the Council reached an agreement on the updated rules for MiFID II, with a clear focus on transparency and the regulation of automated trading systems (European Union, 2014b).

Specifically, MiFID II introduces rules on algorithmic trading in financial instruments. Any firm participating in algorithmic trading is required to ensure it has effective controls in place, such as circuit breakers to halt trading if price volatility becomes too high. Also, any algorithms used must be tested and authorised by regulators. We find the last requirement particularly interesting as MiFID II is not specific about how algorithmic

[1] Many of the articles in the original draft of the MiFID II proposal were met with accusations of tokenistic politics from both industry and academia (see http://www.bankingtech.com/81761/mifid-ii-is-a-dogs-dinner-says-former-uk-government-advisor/).

trading strategies are to be tested.

Given the clear need for robust methods for testing these strategies in such a new, relatively ill-explored and data-rich complex system, an agent-oriented approach, with its emphasis on autonomous actions and interactions, is an ideal approach for addressing questions of stability and robustness. As such we hope to build on the work of Rust et al. (1992), Darley and Outkin (2007), and Mastromatteo et al. (2014) outlined in Chapter 2 by exploring and agent based model that includes important agent-agent interactions that have been shown to give rise to spikes and crashes in price that have but have been missing from previous studies.

Thus, in this chapter, we describe an agent-based simulation environment that is realistic and robust enough for the analysis of algorithmic trading strategies. In detail, we describe an agent-based market simulation that centres around a fully functioning LOB and populations of agents that represent common market behaviours and strategies: market makers, fundamental traders, high-frequency momentum traders, high-frequency mean reversion traders and noise traders. We demonstrate that the model accurately reproduces empirically observed values for: the autocorrelation, volatility clustering, kurtosis and variance of price return and order-sign time series; the price impact function and the occurrence of extreme price events as described in Section 2.3.

The model described in this chapter includes agents that operate on different timescales and whose strategic behaviours depend on other market participants. We find the decoupling of actions across timescales combined with dynamic behaviour of agents to be essential in dictating the more complex patterns seen in high-frequency order-driven markets. Consequently, this chapter presents a model that is represents more trading behaviours and is able to replicate more of the empirically observed empirical regularities than any other study. Such abilities provide a crucial step towards a viable platform for the testing of trading algorithms as outlined in MiFID II.

This chapter is structured as follows. Section 5.1 provides a description of the model structure and agent behaviours in detail. In Section 5.2 the results are summarised while Section 5.3 gives concluding remarks.

5.1 The Model

This chapter describes a model that implements a fully functioning limit order book as used in most electronic financial markets (see Section 2.1 for a detailed description). The model does not concern itself with a representation of time with any marked relation to real time steps. Instead, the model is stated in pseudo-continuous time. That said, a simulated *day* is divided into $T = 300,000$ periods (approximately the number of 10^{ths} of a second in an 8.5 hour trading day) and during each period there is a possibility

for each agent to act — a close approximation to reality. This unit free model of time enables us to steer clear of unnecessary (over)calibration of the model in order for it results to be compared to real-word timelines and instead to focus on relative difference of the timescales upon which agents act and the emergent dynamics that occur as a result.

The model comprises of 5 agent types: Market makers, liquidity consumers, mean reversion traders, momentum traders and noise traders that are each presented in detail later in this section. To replicate the mismatch in the timescales upon which market participants can act (as highlighted by Johnson et al. (2013)), during each period every agent is given the opportunity to act based on probability, δ_τ, that is determined by their type, τ, (market maker, trend follower, etc.). In more detail, to represent a high-frequency trader's ability to react more quickly to market events than, say, a long term fundamental investor, we assigned a higher delta providing a higher chance of being chosen to act. Importantly, when chosen, agents are not required to act. This facet allows agents to vary their activity through time and in response the market, as with real-world market participants. A more formal treatment of the simulation logic is presented in Algorithm 8:

Algorithm 8 Simulation logic - *rand()* represents a function that generates a uniformly distributed floating point number in the interval [0,1].

```
for t = 0 to T do
    for all agents do
        if rand() < δ_τ then
            Agent may submit, modify or cancel an order or do nothing.
        end if
        All agents update their internal states.
    end for
end for
```

The probability of a member of each agent group acting is denoted δ_{mm} for market makers, δ_{lc} for liquidity consumers, δ_{mr} for mean reversion traders, δ_{mt} for momentum traders and δ_{nt} for noise traders. If an agent is chosen to act, that agent is asked whether it wishes to perform an action. If it wishes to submit an order, it will communicate an order type, volume and price determined by that agent's internal logic. The order is then submitted to the LOB where it is matched using price-time priority (for a detailed explanation see Section 2.1). Although the model contains a fair number of free parameters, those parameters are determined through experiment (see Section 5.2.1) and found to be relatively insensitive to reasonable variation. Table 5.2 lists the parameters of the model along with their default values. Below we define the 5 agent types.

Market Makers

Market makers represent market participants who attempt to earn the spread by supplying liquidity on both sides of the LOB. In traditional markets, market makers were appointed but in modern electronic exchanges any agent is able to follow such a strategy [2]. These agents simultaneously post an order on each side of the book, maintaining an approximately neutral position throughout the day. They make their income from the difference between their bids and offers. If one or both limit orders is executed, it will be replaced by a new one the next time the market maker is chosen to trade. In this study we implement an intentionally simple market making strategy based on the liquidity provider strategy described by Oesch (2014). Each round, the market maker generates a prediction for the sign of the next period's order using a simple w period rolling-mean estimate. When a market maker predicts that a buy order will arrive next, she will set her sell limit order volume to a uniformly distributed random number between v_{min} and v_{max} and her buy limit order volume to v^-. An algorithm describing the market makers logic is given in Algorithm 9.

Algorithm 9 Market Maker logic.

if rand() $< \delta_{mm}$ **then**
    Cancel any existing orders
    if predicts next order is buy **then**
        Submit sell at best price with volume $= U(v_{min}, v_{max})$
        Submit buy at best price with volume $= v^-$
    else
        Submit buy at best price with volume $= U(v_{min}, v_{max})$
        Submit sell at best price with volume $= v^-$
    end if
end if
Update buy/sell prediction with w-period rolling mean

Liquidity Consumers

Liquidity consumers represent large slower moving funds that make long term trading decisions based on the rebalancing of portfolios. In real world markets, these are likely to be large institutional investors. These agents are either buying or selling a large order of stock over the course of a day for which they hope to minimise price impact and trading costs. Whether these agents are buying or selling is determined with equal probability. The initial volume h_0 of a large order is drawn from a uniform distribution between h_{min} and h_{max}. To execute the large order, a liquidity consumer agent looks at the current volume available at the opposite best price, Φ_t. If the remaining volume of his large

[2] Although, at present, any player in a LOB may follow a market making strategy, MIFiD II is likely to require all participants that wish to operate such a strategy to register as a market maker. This will require them to continually provide liquidity at the best prices no matter what.

order, h_t, is less than Φ_t the agent sets this period's volume to $v_t = h_t$. Otherwise, he takes all available volume at the best price $v_t = \Phi_t$. For simplicity liquidity consumers only utilise market orders. An algorithm describing the Liquidity Consumer's logic is given in Algorithm 10.

Algorithm 10 Liquidity Consumer logic.

if Start of day **then**
  if $rand() < 0.5$ **then**
    Buying
  else
    Selling
  end if
  Initial order volume, $h_0 = U(h_{min}, h_{max})$
end if
if rand() $< \delta_{lc}$ **then**
  if $h_t \leq \Phi_t$ **then**
    Submit market order with volume $v_t = h_t$
  else
    Submit market order with volume $v_t = \Phi_t$
  end if
end if
$h_t = h_t - v_t$

Momentum Traders

This group of agents represents the first of two types of high frequency traders. Also known as *trend followers*, they invest based on the belief that price changes have inertia — a strategy known to be widely used (Keim and Madhavan, 1995). A momentum strategy involves taking a long position when prices have been recently rising, and a short position when they have recently been falling. Specifically, we implement simple momentum trading agents that rely on calculating a *rate of change* (roc) to detect momentum, given by:

$$\text{roc}_t = \frac{p_t - p_{(t-n_r)}}{p_{(t-n_r)}} \tag{5.1}$$

where p_t is the price of an asset at time t. When roc_t is greater than some threshold κ the momentum trader enters buy market orders of a value proportional to the strength of the momentum. That is, the volume of the market order will be:

$$v_t = |\text{roc}_t| * W_{a,t} \tag{5.2}$$

where $W_{a,t}$ is the wealth of agent a at time t. A complete description of the momentum trader's logic is given in Algorithm 11.

Algorithm 11 Momentum Trader logic.

if rand() $< \delta_{mt}$ **then**
  if $roc_t \geq \kappa$ **then**
    Submit buy market order with $v_t = |\text{roc}_t| * W_{a,t}$
  else if $roc_t \leq -\kappa$ **then**
    Submit sell market order with $v_t = |\text{roc}_t| * W_{a,t}$
  end if
end if
Update ROC using Equation 5.1

Mean Reversion Traders

The second group of high-frequency agents are the mean-reversion traders. Again, this is a well documented strategy (Serban, 2010) in which traders believe that asset prices tend to revert towards an historical average (although this may be a very short term average). They attempt to generate profit by taking long positions when the market price is below the historical average price, and short positions when it is above. Specifically, we define agents that, when chosen to trade, compare the current price to an exponential moving average of the asset price, ema_t, at time t calculated as:

$$\text{ema}_t = \text{ema}_{(t-1)} + \alpha(p_t - \text{ema}_{(t-1)}) \tag{5.3}$$

where p_t is the price at time t and α is a discount factor that adjusts the recency bias. If the current price, p_t, is k standard deviations above ema_t the agent enters a sell limit order at a single tick size improvement of the best price offer, and if it's k standard deviations below then he enters a buy. The volume of a mean reversion trader's order is denoted by v_{mr}. An algorithm describing the mean reversion traders' logic is given in Algorithm 12.

Algorithm 12 Mean Reversion Trader logic.

if rand() $< \delta_{mr}$ **then**
  if $p_t - \text{ema}_t \geq k\sigma_t$ **then**
    Submit sell limit just inside best ask with $v_t = v_{mr}$
  else if $\text{ema}_t - p_t \geq k\sigma_t$ **then**
    Submit buy limit just inside best bid with $v_t = v_{mr}$
  end if
end if
Update ema_t using Equation 5.3

Noise Traders

These agents are defined so as to capture all other market activity and are modelled very closely to the agents introduced by Cui and Brabazon (2012). There parameters

are fitted using empirical order probabilities. The noise traders are randomly assigned whether to submit a buy or sell order in each period with equal probability. Once assigned, they then randomly place either a market or limit order or cancel an existing order according to the probabilities λ_m, λ_l and λ_c respectively.

When submitting an order, the size of that order, v_t, is drawn from a log-normal distribution described by:

$$v_t = exp(\mu + \sigma u_v) \tag{5.4}$$

where μ and σ represent the mean and standard deviation of the v_t's natural logarithm and u_v is a uniformly distributed random variable between 0 and 1. If a limit order is required, on the other hand, the noise trader faces 4 further possibilities:

- *With probability* λ_{crs} the agent crosses the spread and places a limit order at the opposing best ensuring immediate (but pottentially partial) order fulfilment. If the order is not completely filled, it will remain in the order book.

- *With probability* λ_{inspr} the agent places a limit order at a price within the bid and ask spread, p_{inspr}, that is uniformly distributed between the best bid and ask.

- *With probability* λ_{spr} the agent places a limit order at the best price available on their side of the book.

- *With probability* λ_{offspr} the agent will place a limit order deeper in the book, at a price, p_{offspr}, distributed with the power law:

$$\text{xmin}_{\text{offspr}} * (1 - u_0)^{-\frac{1}{\beta - 1}} \tag{5.5}$$

  where u_0 is a uniformly distributed random variable between 0 and 1 while $\text{xmin}_{\text{offspr}}$ and β are parameters of the power law that are fitted to empirical data.

The sum of these probabilities must equal one ($\lambda_{\text{crs}} + \lambda_{\text{inspr}} + \lambda_{\text{spr}} + \lambda_{\text{offspr}} = 1$). To prevent spurious price processes, noise traders' market orders are limited in volume such that they cannot consume more than half of the total opposing sides available volume. Another restriction is that noise traders will make sure that no side of the order book is empty and place limit orders appropriately. The full noise trader logic is described in Algorithm 13.

Algorithm 13 Noise Trader logic.

if rand() $< \delta_{nt}$ **then**
  if rand() < 0.5 **then**
    Selling
  else
    Buying
  end if
  Generate $U(0,1)$ to determine action, λ_m, λ_l and λ_c.
  switch action **do**
    case Submit Market Order
      Submit market order with volume calculated by Equation 5.4
    case Submit Limit Order
      Generate $U(0,1)$ for action, λ_{crs}, λ_{inspr}, λ_{spr} & λ_{offspr}.
      switch Limit Order **do**
        case Crossing limit order
          Submit limit order at opposing best price using Equation 5.4
        case Inside spread limit order
          Generate a random value, $p_{insprd} = U(BetBid, BestAsk)$
          Submit limit order at price p_{insprd} using Equation 5.4
        case Spread limit order
          Submit limit order at the best price using Equation 5.4
        case Off-spread limit order
          Generate a random value, $rp_{offsprd}$ using Equation 5.5
          Submit order at a price $rp_{offsprd}$ outside of spread using Equation 5.4.
    case Cancel Limit Order
      Cancel the oldest order previously submitted.
end if

5.2 Results

In this section we begin by performing a global sensitivity analysis to explore the influence of the parameters on market dynamics and ensure the robustness of the model. Subsequently, we explore the existence of the following stylised facts in depth-of-book data from the Chi-X exchange compared with our model: fat tailed distribution of returns, volatility clustering, autocorrelation of returns, long memory in order flow, concave price impact function and the existence of extreme price events.

5.2.1 Sensitivity analysis

In this section, we asses the sensitivity of the agent-based model described above. To do so, we employee an established approach to global sensitivity analysis known as variance-based global sensitivity (**?**).

In variance-based global sensitivity analysis, the inputs to an agent-based model are treated as random variables with probability density functions representing their associated uncertainty. The impact of the set of input variables on a model's output measures may be independent or cooperative and so the output $f(x)$ may be expressed as a finite hierarchical cooperative function expansion using an analysis of variance (ANOVA). Thus, the mapping between input variables $x_1, \ldots, x_n$ and output variables

$f(x) = f(x_1, \ldots, x_n)$ may be expressed in the following functional form:

$$f(x) = f_0 + \sum_i f_i(x_i) + \sum_{i<j} f_{i,j}(x_i, x_j) + \ldots + f_{1,2,\ldots,n}(x_1, x_2, \ldots, x_n) \tag{5.6}$$

where f_0 is the zeroth order mean effect, $f_i(x_i)$ is a first order term that describes the effect of variable x_i on the output $f(x)$, and $f_{i,j}(x_i, x_j)$ is a second order term that describes the cooperative impact of variables x_i and x_j on the output. The final term, $f_{1,2,\ldots,n}(x_1, x_2, \ldots, x_n)$ describe the residual n^{th} order cooperative effect of all of the input variables. Consequently, the total variance is calculated as follows:

$$D = \int (f(x) - f_0)^2 \rho(x) \mathrm{d}x \tag{5.7}$$

where $\rho(x)$ is the probability distribution over input variables. Partial variances are then defined as:

$$D_{i_1 \ldots, i_s} = \int f^2_{i_1 \ldots, i_s}(x_{i_1}, \ldots, x_{i_s}) \rho(x) \mathrm{d}x \tag{5.8}$$

Now, the total partial variances D_i^{tot} for each parameter x_i, $i = \overline{1, n}$, is computed as

$$D_i^{tot} = \sum_{\langle i \rangle} D_{i_1, \ldots, i_s}; 1 \leq s \leq n, \tag{5.9}$$

where $\langle i \rangle$ refers to the summations over all D that contains i. Once the above is computed, the total sensitivity indicies can be calculated as:

$$S_i^{tot} = \frac{D_i^{tot}}{D}; 0 \leq S_i^{tot} \leq 1, \tag{5.10}$$

It follows that the total partial variance for each parameter x_i is

$$D_i^{tot} = D - Var(\mathbb{E}(f|x_{-i})) \equiv \mathbb{E}(Var(f|x_{-i})) \tag{5.11}$$

In this study, twenty three input parameters and four output parameters are considered. The input parameters include: The probabilities of each of the five agent groups performing an action (δ_{mm}, δ_{lc}, δ_{mr}, δ_{mt}, δ_{nt}), the market makers' parameters (w, the period length of the rolling mean, and v_{max}, the max order volume for limit order), the upper limit of the distribution from which liquidity consumers' order volume is drawn (h_{max}), the momentum traders' parameters (n_r, the lag parameter of the ROC, and κ, the trade entry point threshold), as well as the following noise trader parameters:

- Probability of submitting a market order, λ_m
- Probability of submitting a limit order, λ_l
- Probability of cancelling a limit order, λ_c
- Probability of a crossing limit order λ_{crs}
- Probability of a inside-spread limit order λ_{inspr}
- Probability of a spread limit order λ_{spr}
- Probability of a off-spread limit order λ_{offspr}
- Market order size distribution parameters, μ_{mo} and σ_{mo}
- Limit order size distribution parameters, μ_{lo} and σ_{lo}
- Off-spread relative price distribution parameters, $\text{xmin}_{\text{offspr}}$ and β_{offspr}

The following output parameters are monitored: the Hurst exponent H of volatility (as calculated using the DFA method described by Peng et al. (1994)), the mean autocorrelation of mid-price returns $R(m)$, the mean first lag autocorrelation term of the order-sign series $R(o)$, and the best fit exponent β of the price impact function as in Equation 2.10.

As our model is stochastic (agents' actions are defined over probability distributions), there is inherent uncertainty in the range of outputs, even for fixed input parameters. In the following, ten thousand samples from within the parameter space were generated with the input parameters distributed uniformly in the ranges displayed in Table 5.1.

or each sample of the parameters space, the model is run for 300000 trading periods to approximately simulate a trading day on a "high-frequency" timescale. The global variance sensitivity, as defined in Equation 5.11 is presented in Figure 5.1.

The global variance sensitivities clearly identify the upper limit of the distribution from which liquidity consumers' order volume is drawn (h_{max}) and the probabilities of each of the agent groups acting (particularly those of the high-frequency traders, δ_{mr} and δ_{mt}) as the most important input parameters for all outputs. The biggest influence of each of these parameters was on the mean first lag autocorrelation term of the order-sign series $R(o)$ followed by the exponent of the price impact function β.

To find the set of parameters that produces outputs most similar to those reported in the literature and to further explore the influence of input parameters we perform a large scale grid search of the input space. This yields the 'optimal' set of parameters displayed in Table 5.2. With this set of parameters we go on tho explore the model's ability to reproduce the various statistical properties outlines in Section 2.3.

Parameter	**Symbol**	**Setting**
Probability of Market Makers acting	δ_{mm}	$[0.05, 0.95]$
Probability of Liquidity Consumers acting	δ_{lc}	$[0.05, 0.95]$
Probability of Momentum Traders acting	δ_{mr} $[0.05, 0.95]$	
Probability of Mean Reverters acting	δ_{mt}	$[0.05, 0.95]$
Probability of Noise Traders acting	δ_{nt}	$[0.05, 0.95]$
Market maker (mm) parameters		
Max order volume	v_{max}	$[10^3, 10^6]$
Rolling mean period	w	$[10, 10^3]$
Liquidity consumer (lc) parameters		
Max order volume	h_{max}	$[10^3, 10^6]$
Momentum trader (mt) parameters		
ROC lag	n_r	$[1, 100]$
Trade entry threshold	κ	$[10^{-6}, 10^{-1}$
Noise trader (nt) parameters		
Market order probability	λ_m	$[0, 1]$
Limit order probability	λ_l	$[0, 1]$
Cancel order probability	λ_c	$[0, 1]$
Market order size	μ_{mo}	$[2, 10]$
Market order size	σ_{mo}	$[0, 1]$
Limit order size	μ_{lo}	$[2, 10]$
Limit order size	σ_{lo}	$[0, 1]$
Off-spread relative price	$\text{xmin}_{\text{offspr}}$	$[0, 1]$
Off-spread relative price	β_{offspr}	$[0, 1]$
Crossing limit order	λ_{crs}	$[0, 1]$
Inside-spread limit order	λ_{inspr}	$[0, 1]$
Spread limit order	λ_{spr}	$[0, 1]$
Off-spread limit order	λ_{offspr}	$[0, 1]$

TABLE 5.1: Parameter ranges for global sensitivity analysis

5.2.2 Fat Tailed Distribution of Returns

Figure 5.2 displays a side-by-side comparison of how the kurtosis of the mid-price return series varies with lag length for our model and an average of the top 5 most actively traded stocks on the Chi-X exchange in a period of 100 days of trading from 12th February 2013 to 3rd July 2013. A value of 1000 on the x-axis mean that the return was taken as $\log(p_{t+1000}) - \log(p_t)$. In our LOB model, only substantial cancelations, orders that fall inside the spread, and large orders that cross the spread are able to alter the mid price. This generates many periods with returns of 0 which significantly reduces the variance estimate and generates a leptokurtic distribution in the short run, as can be seen in Figure 5.2(a).

Kurtosis is found to be relatively high for short timescales but falls to match levels of the normal distribution at longer timescales. This not only closely matches the pattern of decay seen in the empirical data displayed in Figure 5.2(b) but also agrees with the findings of Cont (2001) and Gu et al. (2008).

FIGURE 5.1: Heatmap of the global variance sensitivity.

5.2.3 Volatility Clustering

To test for volatility clustering, we compute the Hurst exponent of volatility using the DFA method described by Peng et al. (1994). Figure 5.3 details the percentage of simulations runs with significant volatility clustering — defined as $0.6 < H < 1$. Once again, in the shortest time lags volatility clustering seems to be present at short timescales in all the simulations but rapidly disappears for longer lags in agreement with Lillo and Farmer (2004).

5.2.4 Autocorrelation of returns

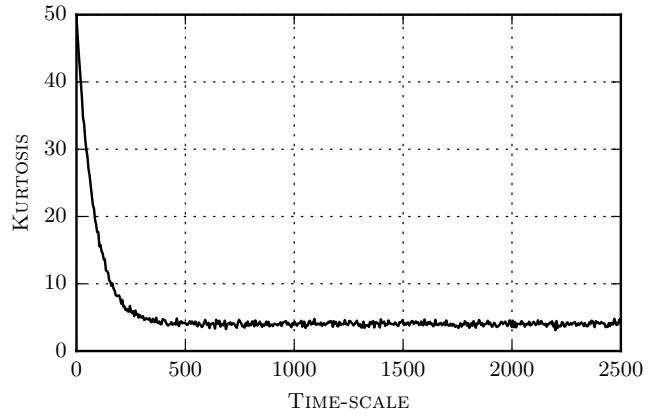
Table 5.3 reports descriptive statistics for the first lag autocorrelation of the returns series for our agent based model and for the Chi-X data. In both instances, there is a very weak but significant autocorrelation in both the mid-price and trade price returns.

Market parameters	**Setting**
Initial Price	100
Initial Spread	0.05
Tick Size	0.01
Agent group	**Action probability**
δ_{mm}	0.10
δ_{lc}	0.10
δ_{mr}	0.40
δ_{mt}	0.40
δ_{nt}	0.75
Market maker (mm) parameters	**Setting**
v_{min}	1
v_{max}	200,000
v	1
w	50
Liquidity consumer (lc) parameters	**Setting**
h_{min}	1
h_{max}	100,000
Mean reversion (mr) parameters	**Setting**
v_{mr}	1
α	0.94
Momentum trader (mt) parameters	**Setting**
n_r	5
κ	0.001
Noise trader (nt) parameters	**Setting**
Buy or sell probability	0.5
Market order probability	$\lambda_m = 0.03$
Limit order probability	$\lambda_l = 0.54$
Cancel order probability	$\lambda_c = 0.43$
market order size	$\mu_{mo} = 7$ $\sigma_{mo} = 0.1$
limit order size	$\mu_{lo} = 8$ $\sigma_{lo} = 0.7$
off-spread relative price	$\text{xmin}_{\text{offspr}} = 0.005$ $\beta_{\text{offspr}} = 2.72$
NT limit order types	Probability
crossing limit order	$\lambda_{crs} = 0.003$
inside-spread limit order	$\lambda_{inspr} = 0.098$
spread limit order	$\lambda_{spr} = 0.173$
off-spread limit order	$\lambda_{offspr} = 0.726$

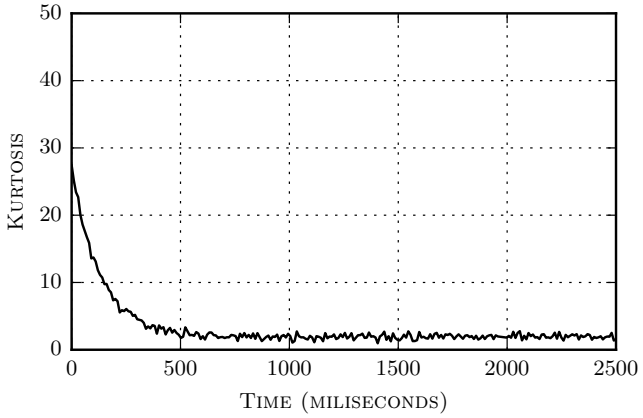
TABLE 5.2: Optimal parameter values

The median autocorrelations of mid-price returns for the agent-based model and the Chi-X data were found to be -0.0034 and -0.0044, respectively. Using a non-parametric test, the distributions of the two groups were not found to differ significantly (MannWhitney $U = 300, P > 0.1$ two-tailed).

This has been empirically observed in other studies (see Section 2.3.3) and is commonly thought to be due to the refilling effect of the order book after a trade that changes the best price. The result is similar for the trade price autocorrelation but as a trade price will always occur at the best bid or ask price a slight oscillation is to be expected and is observed.



(a) Model Kurtosis



(b) Empirical Kurtosis

Figure 5.2: Kurtosis by timescale for our model and for the empirical data (in milliseconds).

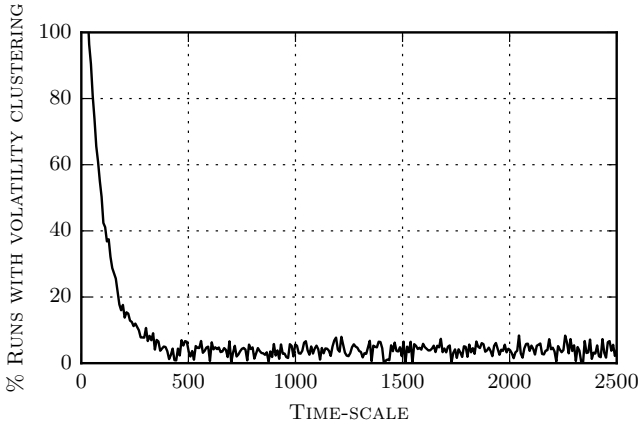


Figure 5.3: Volatility clustering by timescale

Stats	Min.	1st Q.	Median	3rd Q.	Max.
Agent-based model					
AC mid-price returns	-0.0208	-0.0071	-0.0034	0.0090	0.0366
AC trade price returns	0.5734	0.6029	0.6200	0.6209	0.6476
Empirical data					
AC mid-price returns	-0.0233	-0.0169	-0.0044	0.0081	0.0436
AC trade price returns	0.5287	0.5353	0.5678	0.6000	0.6346

TABLE 5.3: Return autocorrelation statistics

Stats	Min.	1st Q.	Mean	3rd Q.	Max.
Agent-based model					
AC order signs	0.1983	0.2059	0.2079	0.2104	0.2172
H order signs	0.6734	0.7029	0.7115	0.7209	0.7476
Empirical data					
AC order signs	0.2000	0.2544	0.2629	0.2701	0.3013
H order signs	0.7681	0.8053	0.8444	0.8780	0.8849

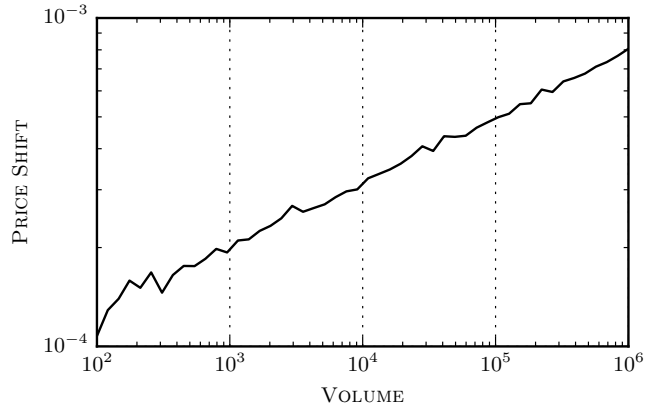
TABLE 5.4: Order sign statistics

5.2.5 Long Memory in Order Flow

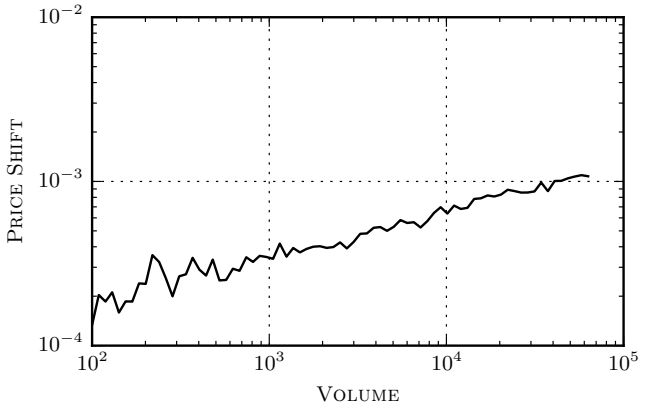
As presented in Table 5.4, we find the mean first lag autocorrelation term of the order-sign series for our model to be 0.2079 which is close to that calculated for the empirical data and those reported in the literature. Most studies find the order sign autocorrelation to be between 0.2-0.3 (see Lillo and Farmer (2004) for example). In Table 5.4, *H Order signs* shows a mean Hurst exponent of the order signs time series for our model of ≈ 0.7 which indicates a long-memory process and corresponds with the findings of previous studies and with our own empirical results (see Lillo and Farmer (2004) and Mike and Farmer (2008)).

5.2.6 Concave Price Impact

Figure 5.4(a) illustrates the price impact in the model as a function of order size on a log-log scale. The concavity of the function is clear. The shape of this curve is very similar to that of the empirical data from Chi-X shown in Figure 5.4(b). The price impact is for the model is found to be best fit by the relation $\Delta p \propto v^{0.28}$, while the empirically measured impact was best fit by $\Delta p \propto v^{0.35}$. Both of these estimates of the exponent of the impact function agree with the findings of Lillo et al. (2003), Farmer and Lillo (2004) and Hopman (2007) but the model is sensitive to the volume provided by the market makers. When the market makers order volume is reduced, the volume at the opposing best price reduces compared to the rest of the book. This allows smaller trades to eat further into the liquidity stretching the right-most side of the curve.



(a) Model Price Impact



(b) Empirical Price Impact

FIGURE 5.4: Log-log price impact function for our model and for the Chi-X data.

Figure 5.5 demonstrates the effects of varying the liquidity consumers' volume parameter h_{max} on the price impact curve. This parameter appears to have very little influence on the shape of the price impact function. However, it does appear to have an effect on the size of the impact. Although h_{max} is relatively insensitive to minor changes, when the volume traded by the liquidity consumers is reduced dramatically, the relative amount of available liquidity in the market increases to the point where price impact is reduced. Very similar results are seen as the market makers' order size (v_{max}) is increased.

Figure 5.6 shows the effects on the price impact function of adjusting the relative probabilities of events from the high frequency traders. It is clear that strong concavity is retained across all parameter combinations but some subtle artefacts can be seen. Firstly, increasing the probability of both types of high frequency traders equally seems to have very little effect on the shape of the impact function. This is likely due to the strategies of the high frequency traders restraining one another. Although the momentum traders are more active — jumping on price movements and consuming liquidity at the top of the book — they are counterbalanced by the increased activity of the mean reversion traders who replenish top-of-book liquidity when substantial price movements

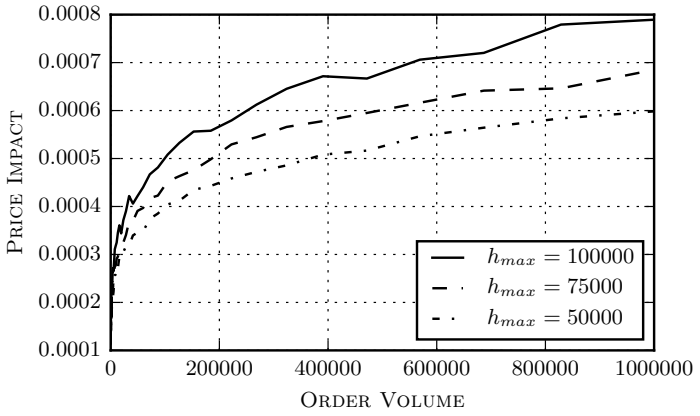


FIGURE 5.5: The price impact function with different liquidity consumer parameterisations. Each line represents a different setting for h_{max}

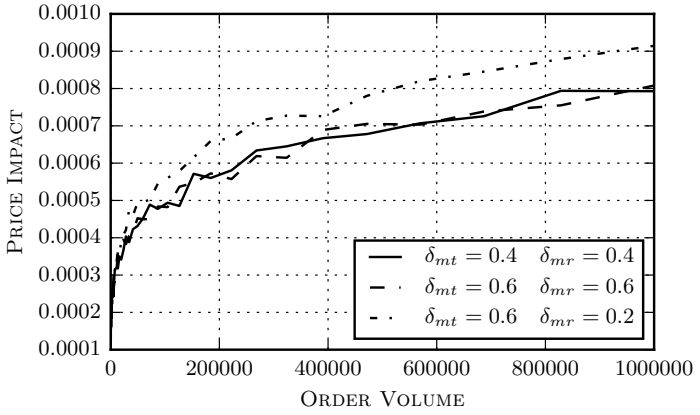


FIGURE 5.6: Price impact for various values for the probability of the high frequency traders acting.

occur. In the regime where the probability of momentum traders acting is high but the probability for mean reversion traders is low (the dotted line) we see an increase in price impact across the entire range of order sizes. In this scenario, when large price movements occur, the activity of the liquidity consuming trend followers outweighs that of the liquidity providing mean reverters, leading to less volume being available in the book and thus a greater impact for incoming orders.

5.2.7 Extreme Price Events

We follow the definition of Johnson et al. (2013) and define an extreme price event as an occurrence of a stock price ticking down [up] at least ten times before ticking up [down] and with a price change exceeding 0.8% of the initial price. Figure 5.7 shows a plot the mid-price time-series provides with an illustrative example of a flash crash occurring in

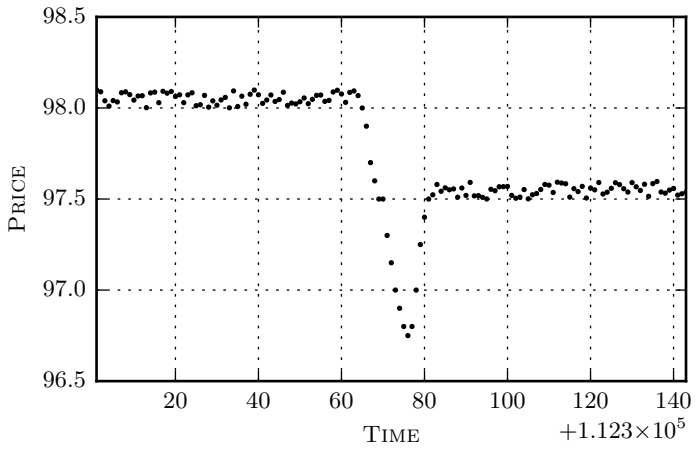


Figure 5.7: Flash crash example

Stats	Min.	Median	Mean	Max.
Events per day in ABM	0	1	0.8286	3
Events per day in Chi-X	0	1	1.0066	6

Table 5.5: Flash crash statistics

the simulation. During this event, the number of sequential down ticks is 11, the price change is 1.3%, and the event lasts for 12 simulation steps.

Table 5.5 shows statistics for the number of events for each day in the Chi-X data and per simulated day in our ABM. On average, in our model, there are 0.8286 events per day very close to the average number observed in empirical data.

Upon inspection, we can see that such events occur when an agent makes a particularly large order that eats through the best price (and sometimes further price levels). This causes the momentum traders to submit particularly large orders on the same side, setting off a positive feedback chain that pushes the price further in the same direction. The price begins to revert when the momentum traders begin to run out of cash while the mean reversion traders become increasingly active.

Figure 5.8 illustrates the relative numbers of extreme price events as a function of their duration. The event duration is the time difference (in simulation time) between the first and last tick in the sequence of jumps in a particular direction. It is clear that these extreme price events are more likely to occur quickly than over a longer timescale. This is due to the higher probability of momentum traders acting during such events. It is very rare to see an event last longer than 35 time steps.

Figure 5.9 shows the relative number of crash and spike events as a function of their duration for different schemes of high frequency activity. The solid line shows the result

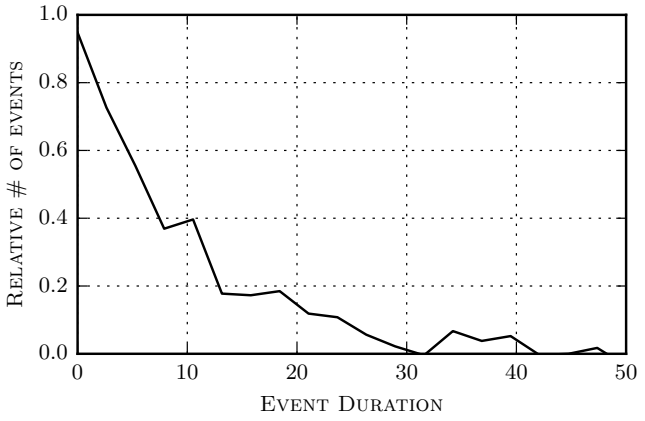


FIGURE 5.8: Relative numbers of crash/spike events as a function of their duration

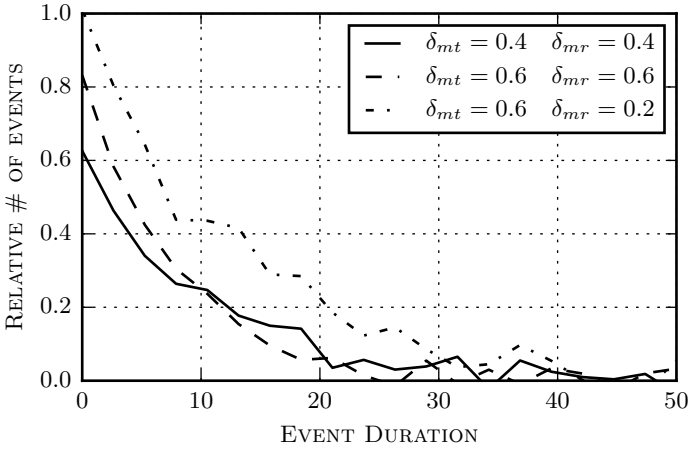


FIGURE 5.9: Flash crash occurrence with various values for the probability of the high frequency traders acting.

with the standard parameter setting from Table 5.2. The dashed line shows results from a scheme with an increased probability of both types of high frequency trader acting. Here, we see that there is an increased incidence of short duration flash events. It seems that the increased activity of the trend follows causes price jumps to be more common while the increased activity of the mean reverts ensures that the jump is short lived. In the scenario where the activity of the momentum followers is high but that of the mean reverts is low (the dotted line) we see an increase in the number of events cross all time scales. This follows from our previous analogy.

5.3 Summary

The agent-based simulation proposed in this chapter is designed to allow practitioners and academics to better understand the behaviour of automated algorithmic trading strategies. It is able to replicate a number of well-known statistical characteristics of financial markets including: clustered volatility, autocorrelation of returns, long memory in order flow, concave price impact and the presence of extreme price events. This supports prevailing empirical findings from microstructure research.

Perhaps the most intersting observation is the emergence of flash crash-like behaviour without explicit modelling of trading strategies to generate such behaviour. Though the well-known trading strategies outlined above have been modelled in previous studies, the addition of the ability for agents to act on differing timescales gives rise to previously undocumented flash crash-like behaviour. Note, then, that we find extreme price events to occur not due to malicious behaviours that seek to disrupt the market and benefit from the aftermath but instead to emerge as a happenstance of the interactions of otherwise benign trading strategies. As such, policy makers should take note that focussing efforts on the prevention of malicious behaviours and guiding regulation in this direction may prove not to be fruitful. Instead, regulators should focus on understanding how the interactions of market participants can lead to unexpected systemic behaviours as demonstrated here.

In this vein, a number of interesting facets are explored. Firstly, we find that increasing the total number of high frequency participants has no discernible effect on the shape of the price impact function while increased numbers do lead to an increase in flash crash events. We also find that the balance of trading strategies is important in determining the shape of the price impact function. Specifically, excess activity from aggressive liquidity-consuming strategies leads to a market that yields increased price impact.

Chapter 6

Conclusions

This thesis has focused on predicting and modelling financial systems. Specifically, a novel random forest (RF) based online ensemble learning system was proposed for predicting both daily price returns and intra-day price impact. In Chapter 3 we described the specific methodologies of this system and showed that RFs were able to produce superior out-of-sample prediction accuracies when forecasting daily price returns. We also showed that introducing our novel method for online generation and combination of multiple RFs outperformed other state-of-the-art methodologies. Finally, it was demonstrated that taking advantage of season regularities in the data produced both better prediction accuracy and greater risk adjusted returns than simply trading every day.

Thus we achieved our first aim of developing a completely autonomous trading system that captures the state of the market in its inputs, makes predictions about price movements, manages risk and generates profit in the face of realistic transaction costs. The academic development and study of such systems is a step closer to understanding the new, highly computerised financial ecosystem.

In Chapter 4, we apply the algorithms mentioned above to the prediction of price impact in limit order books (LOBs). In more detail, upon the occurrence of an event that alters the best prices of the LOB (limit order at the best price, market order, cancellation or modification), each RF in an ensemble makes a prediction before an expert weighting system averages the predictions of all RFs weighted by their recent performance. A separate ensemble is used to predict the price change at $t = 1, 5, 10,$ and 60 seconds after an event. This performance of the RF ensemble is then explored using 100 days of market depth data on the 25 most liquid stocks on the BATS Chi-X exchange. Our system is benchmarked against ensembles of linear regression, neural networks and support vector models as well as single random forests and show to produce superior performance than all other models across all timeframes. We also explore the importance of the features used to predict price impact, finding that the drivers of impact vary through time.

As outlined in Section 2.4.1 models of price impact are particularly important for in-

stitution traders as an integral part of an optimal order execution model. More than that, models such as that outlined in Chapter 4 provide us with an insight into important issues like how changes in the order book affect price impact and what drives price movement across various timescales.

In light of the requirements of the forthcoming MiFID II regulations, Chapter 5 describes an interactive simulation environment for analysing trading algorithms. Not only does such a model allow regulators to understand the effects of algorithms on the market dynamics it also allows trading firms to optimise proprietary algorithms.

The strategic interaction of the agents and the differing time-scales on which they act are, at present, unique to this model and crucial in dictating the complexities of high-frequency order-driven markets. As a result, this thesis presents the first model capable of replicating all of the aforementioned stylised facts of limit order books, an important step towards an environment for testing automated trading algorithms. Such an environment not only fulfils a requirement of MiFID II, more than that, it makes an important step towards increased transparency and improved resilience of the complex socio-technical system that is our brave new marketplace.

6.1 Implications and Limitations

This thesis has used machine learning and agent-based modelling techniques to explore the drivers of market price dynamics. Chapter 4 suggested that the most influential features that affect price movements in limit order books are the volume at the best price and the number of recent order arrivals, which are found to inversely relate to price movements, and average price increment, spread and event size, which are directly related. Chapter 5 takes a different, bottom-up, approach, finding that an increase in the total number of high frequency strategies has no discernible effect on the shape of the price impact function though do lead to an increase in flash crash events. It was also found that the balance of trading strategies is important in determining the shape of the price impact function. Specifically, excess activity from aggressive liquidity-consuming strategies lead to a markets with increased price impact.

Although there is a need for improved understanding and stability in electronic limit order book markets, such markets represent complex adaptive systems where small micro-level changes can give rise to large (often unforeseen) consequences. Thus, regulators may wish to tread carefully. Our research suggests the following:

- Standard models of LOB dynamics are not adequate for capturing the complex interactions of agents and time varying nature of the features that affect the markets.

- Agent-based models are able to provide a simulation environment where market dynamics are emergent phenomena that arise from the interactions of behaviours that occur across differing timescales.
- Regulation of algorithmic trading is not a simple matter of yes or no. Stable markets can arise from environments dominated by high-frequency algorithmic strategies — it is the interactions of these strategies that is particularly important.

However, the extent to which specific policy recommendations might be made is restricted by the limitations of the model. Below, we present a discussion of these limitations and their importance.

Conceptualising Agents

By definition, ABMs consider systems at a decentralised level. Such a level of detail involves the description of a number of agent attributes and behaviours and their interaction with the environment. While we have explored agents that learn and demonstrate strategic behaviours, the agents used in the our work are a dramatic simplification of reality. Real-world market participants have been seen to display irrational behaviour, subjective choices, and complex psychology. However, our agents were still able to generate realistic macro-level stylised facts. Constructing such agents requires a very different kind of data from traditional top-down, mathematic models. We need to know how market participants behave, interact, form relationships and make decisions.

Market Interdependencies

The models in Chapters 4 and 5 are based around a single exchange. In reality, there are significant and important interdependencies in price and liquidity across exchange platforms. Particularly, many algorithmic trading strategies seek to arbitrage between venues and seek liquidity across the now highly fragmented marketplace. However, our model was kept intentionally simple so as to isolate important features and assess the effect of agents acting across differing timescales. Although this was successfully demonstrated, future work into multiple competing exchanges would provide an interesting line of inquiry.

6.2 Future Work

Research into the understanding and prediction of agent behaviour and price dynamics in complex electronic marketplaces is increasingly gathering pace. Furthermore, as trading technology and the regulatory landscape in which it operates is ever changing,

additional research is need in order to improve current modelling techniques and better inform regulatory decisions. Thus, while this thesis makes significant steps towards an understanding of the determinants of price formation in limit order markets, there are a number of areas that require further work. Below we highlight some of the most important of them:

- In the first part of this thesis, we develop a model for forecasting highly non-stationary time-series. Because predicting with online generation of models proved to be so much more effective than traditional static methods, we it would be of particular interest to look in more detail at how various phases of the market can be identified and leveraged for greater prediction accuracy.
- An interesting direction for future research would be the integration of an ensemble based price impact prediction model into an algorithmic trading strategy. Furthermore, the analysis of such a strategy could be performed in the simulation environment developed in the latter part of this thesis.
- While our simulation environment has been shown to accurately produce a number of order book dynamics, the intra-day volume profile has not been examined. It is proposed that the exploration of the relative volumes traded throughout a simulated day and extensions made so as to replicate the well known U-shaped volume profiles (see Jain and Joh (1988); McInish and Wood (1992)).
- It is proposed as future work that the use of intelligent, time-adaptive agents are investigated. Given that real-world firms change strategies over time, it would be interesting to see the effect that adding such an ability has on the behaviour of the model. For example, we might expect that firms that are able to adapt their strategies over time live longer on average or are more profitable, since they may be better equipped to respond to a changing environment.

With the growing popularity and need for improved regulation of electronic order-driven markets, we envisage much future work in this area. To this end, we believe this thesis highlights the pertinent areas of research and makes advances the state of the art for limit order book modelling.

Appendix A

Addition details for Chapter 3

A.1 Inputs

Features	Variants
Month	
Closing price at $t-1$	
Highest price on day $t-1$	
Lowest price on day $t-1$	
Volume traded on day $t-1$	
Index closing price at $t-1$	
Simple moving average of closing prices	
Exponential moving average of closing prices	$n = 10, 16$ and 22
Bollinger bands	
Momentum	
Acceleration	$n = 24$
Rate of change	$n = 10$ and 16
MACD	$s = 24$ and 30
MACD signal	$s = 24$
Relative strength index	$n = 14$ and 20
FAST%K	$n = 24$
FAST%D	
SLOW%K	
SLOW%D	
Chaikin volatility	
Accumulation distribution line	
On balance volume	
Chaikin Oscillator	
VaR-breaks measure (see equation 3.2)	

TABLE A.1: A list of all features considered for input to the prediction model. Those shaded grey are used in the experiments reported in Section 3.4.

A.2 Technical Indicators

Indicator	Description	Calculation
$\text{EMA}_t^c(n)$	Exponential moving average of the last n observations of series p^c	$\text{EMA}(p^c, n) = \lambda \sum_{s=0}^{\infty}(1-\lambda)^s p_{t-s}^c$ $\lambda = \frac{2}{n+1}$ where $n = 10, 16$ and 22
$\text{SMA}_t^c(n)$	Simple moving average of the last n observations of p^c	$\text{SMA}_t(p^c, n) = \frac{1}{n}\sum_{s=0}^{n-1} p_t^c - s$ where $n = 10, 16$ and 22
$\text{Boll}_t^u(n,s)$ and $\text{Boll}_t^d(n,s)$	The Bollinger bands are calculated as a function of s standard deviations from the reference $\text{SMA}_t^c(n)$. When the price crosses above the upper Bollinger band, it is a sign that the market is overbought and vice versa.	$\text{Boll}_t^u(n,s) = \text{SMA}_t^c(n) + s\sigma_t^2(n)$ $\text{Boll}_t^d(n,s) = \text{SMA}_t^c(n) - s\sigma_t^2(n)$ where $s = 2, n = 20, 26$ and 32
	Momentum Indicators	
$\text{MOM}_t(n)$	Momentum represents the price change over the last n periods. If it is above (below) zero, it indicates the the market is trending up (down).	$p_t^c - p_{t-n}^c$ where $n = 12, 18$ and 24
$\text{ACC}_t(n)$	Acceleration represents the change in momentum.	$\text{MOM}_t(n) - \text{MOM}_{t-1}(n)$ where $n = 12, 18$ and 24
$\text{ROC}_t(n)$	The rate of change of a time series p_t^c.	$100.\frac{p_t^c - p_{t-n}^c}{p_{t-n}^c}$ where $n = 10, 16$ and 22
$\text{MACD}_t(s,f)$	The moving average convergence divergence is the difference between two moving average of varying periods (s, f).	$\text{EMA}_t(p^c, s) - \text{EMA}_t(p^c, f)$ where $f = 12$ and $s = 18, 24$ and 30
$\text{MACDS}_t(s,f,n)$	The MACD signal is the moving average of the $\text{MACD}_t(s,f)$ of the last n periods. Usually a buy signal is generated when the $\text{MACD}_t(s,f)$ crosses the signal line	$\text{EMA}_t(\text{MACD}_t(s,f), n)$ where $f = 12$, $n = 9$ and $s = 18, 24$ and 30
$\text{RSI}_t(n)$	The relative strength index compares the days that a price finishes up with those that it finishes down. Normally a buy signal occurs when $\text{RSI}_t(n)$ crosses below a lower band of 30 and a sell signal occurs when it crosses an upper band of 70	$100 - 100/\left(1 + \frac{\text{SMA}_t(p_n^{\text{up}}, n_1)}{\text{SMA}_t(p_n^{\text{dn}}, n_1)}\right)$ where $n = 8, 14$ and 20
$\text{FAST\%K}_t(n)$	The fast stochastic K represents a percent measure of the last close price in relation to the highest high and the lowest low of the last n periods	$\frac{p_t^{\text{uc}} - \min(p_n^l)}{\max(p_n^h) - \min(p_n^l)}$ where $n = 12, 18$ and 24
$\text{FAST\%D}_t(n)$	Moving average of the $\text{FAST\%K}_t(n)$.	$\text{SMA}_t(\text{FAST\%K}_t(n), 3)$
$\text{SLOW\%K}_t(n)$	Moving average of the $\text{FAST\%D}_t(n)$.	$\text{SMA}_t(\text{FAST\%D}_t(n), 3)$
$\text{SLOW\%D}_t(n)$	Moving average of the $\text{SLOW\%K}_t(n)$.	$\text{SMA}_t(\text{SLOW\%K}_t(n), 3)$
	Volatility Indicators	

$\text{CHV}_t(n, n_1)$	The Chaikin volatility evaluates the breadth of the range between high and low prices. It also calculates the ROC of the moving average of the difference between the high and low prices. Often a very fast increase (decrease) of this indicator is a signal that the bottom (top) of the market is near.	$\frac{\text{EMA}_t(p^h - p^l, n)}{\text{EMA}_{t-n_1}(p^h - p^l, n)} - 1$ where $n_1 = 10$
	Volume Indicators	
ADL_t	The accumulation/distribution line was also developed by Chaikin. It is calculated using the close location value (CLV). This indicator compares the close with the range of prices from the same period. A positive value indicates a buying pressure and a negative the opposite.	$\sum_{t=1}^{n} \text{CLV}_t.\text{VOL}_t$ where $CLV_t = \frac{2p^u c_t - p_t^l - p_t^h}{p_t^h - p_t^l}$
OBV_t	The on balance volume evaluates the impact of positive and negative volume flows. It adds the volume when the close has increased and subtracts it when the close has decreased.	if $p_t^c > p_{t-1}^c$: then $\text{OBV}_t = \text{OBV}_{t-1} + \text{VOL}_t$ if $p_t^c < p_{t-1}^c$: then $\text{OBV}_t = \text{OBV}_{t-1} - \text{VOL}_t$
CHO_t	The Chaikin oscillator is the MACD of the ADL. It represents the difference between a short and long $\text{EMA}_t(\text{ADL}, n)$. This indicator is to be interpreted similarly to the MACD.	$\text{EMA}_t(\text{ADL}, n_1) - \text{EMA}_t(\text{ADL}, n_2)$ where $n_1 = 3$, and $n_2 = 10$

Table A.2: A description of the technical indicators with parameters shown in parentheses.

Appendix B

Addition details for Chapter 4

B.1 Stock Descriptives

	Notional Value			Volume		
Index	BXE Book	CXE Book	Combined	BXE Book	CXE Book	Combined
CHERI Pan Europe	8.20%	20.34%	28.55%	9.30%	19.40%	28.70%
CHERI Eurozone	3.28%	15.08%	18.37%	9.29%	17.80%	27.08%
CHERI Pan Europe 60	3.33%	15.50%	18.83%	8.14%	19.75%	27.89%
STOXX Europe 50	3.57%	15.76%	19.33%	8.44%	21.38%	29.83%
EURO STOXX 50	3.21%	14.58%	17.79%	8.53%	18.25%	26.78%
CHERI Euro 40	3.51%	14.63%	18.14%	9.13%	17.79%	26.92%
FTSE 100	3.76%	16.51%	20.27%	8.54%	22.72%	31.26%
CAC 40	3.05%	16.56%	19.61%	8.01%	22.66%	30.67%
DAX	4.40%	17.49%	21.89%	9.00%	23.54%	32.55%
FTSE MIB	3.44%	9.62%	13.06%	8.31%	14.78%	23.09%
IBEX 35	1.07%	7.09%	8.16%	7.20%	14.35%	21.55%
AEX	3.98%	15.12%	19.10%	9.29%	20.33%	29.63%
CHERI Nordic	4.33%	14.94%	19.27%	10.86%	20.19%	31.05%
OMXS30	4.16%	14.01%	18.17%	10.11%	19.81%	29.92%
FTSE 250	3.47%	11.28%	14.75%	8.09%	17.87%	25.96%
BEL20	2.69%	16.73%	19.42%	8.01%	23.20%	31.21%
MDAX	4.37%	12.92%	17.29%	10.06%	23.38%	33.44%
FTSERIOB	1.10%	8.29%	9.39%	6.03%	10.83%	16.86%
CAC Next20	2.98%	13.56%	16.54%	8.22%	20.91%	29.14%
OBX	2.31%	14.94%	17.26%	6.54%	18.01%	24.54%
OMXH25	3.90%	14.18%	18.08%	10.05%	20.34%	30.39%
OMXC20	3.09%	12.69%	15.78%	8.49%	17.72%	26.21%
PSI20	1.95%	6.55%	8.50%	5.90%	7.99%	13.89%
AMX	1.91%	11.85%	13.76%	6.68%	14.84%	21.52%
ATX	1.66%	8.22%	9.89%	7.19%	13.40%	20.59%
ISEQ 20	3.60%	3.17%	6.76%	8.64%	7.27%	15.91%

TABLE B.1: Table percentage of market share of BATS (BXE) and Chi-X (CXE) across various indices.

	Notional Value			Volume		
Market	BXE Book	CXE Book	Combined	BXE Book	CXE Book	Combined
London	8.69%	22.54%	31.23%	8.12%	19.82%	27.94%
Frankfurt	9.26%	22.72%	31.98%	9.58%	21.34%	30.92%
Paris	7.48%	22.98%	30.46%	7.52%	21.95%	29.48%
Milan	8.54%	14.79%	23.33%	8.43%	13.69%	22.11%
Madrid	6.91%	14.93%	21.85%	6.51%	13.15%	19.67%
Amsterdam	8.46%	19.49%	27.95%	7.72%	17.72%	25.44%
Stockholm	10.07%	20.52%	30.58%	10.16%	20.42%	30.58%
Brussels	7.35%	23.14%	30.49%	7.41%	19.86%	27.27%
Oslo	8.45%	19.76%	28.22%	7.18%	16.46%	23.64%
Helsinki	9.15%	21.51%	30.66%	9.56%	18.30%	27.86%
Copenhagen	8.71%	17.78%	26.49%	8.08%	14.14%	22.22%
Lisbon	6.27%	12.06%	18.33%	6.22%	8.57%	14.79%
Vienna	6.39%	13.81%	20.19%	7.51%	13.70%	21.21%
Dublin	6.89%	8.20%	15.09%	8.93%	7.18%	16.11%
Total	8.46%	20.91%	29.37%	7.78%	16.23%	24.01%

TABLE B.2: Table percentage of market share of BATS (BXE) and Chi-X (CXE) across various markets.

B.2 Inputs

Indicator	Parameters
Incoming event features	
Event type	-
Event side (the bid book or the offer book)	-
Event size	-
Event price	-
Price features	
Exponential moving average of the last n observations of best prices	$n = 16$
Bollinger bands of the last n observations of best prices	$n = 32$
Momentum of the best prices over the last n observations	$n = 12,$ and 24
Acceleration of the best prices over the last n observations	$n = 18$
The rate of change of the best prices over the last n observations	$n = 22$
The MACD of the best prices	$f = 12$, $s = 24$
The relative strength index of the best prices over the last n observations	$n = 20$ and 32
The fast stochastic K of the best prices over the last n observations	$n = 12$ and 18
The Chaikin volatility of the best prices over the last n observations	$n = 10$
The accumulation/distribution line	-
The Chaikin oscillator	$n_1 = 3$, and $n_2 = 10$
Spread features	
Exponential moving average of the last n observations of spread	$n = 10$
Momentum of the spread over the last n observations	18
The rate of change of the spread over the last n observations	$n = 10, 16$ and 22
The MACD of the spread	$f = 12$, $s = 30$
The relative strength index of the spread over the last n observations	$n = 14$
The fast stochastic K of the spread over the last n observations	$n = 12$ and 24
Liquidity features	
Exponential moving average of bid/ask book volume over the last n observations	$n = 22$
Exponential moving average of volume at best bid/ask price over the last n observations	$n = 12$ and 36
*Momentum*of bid/ask book volume over the last n observations	$n = 12, 24$ and 36
Number of price improvements in the last n observations	$n = 25,$ and 50
Number of trades in the last n observations	$n = 50$
Number of bid/ask quotes arrived in the last n observations	$n = 50$
Number of bid/ask cancellations in the last n observations	$n = 50$
Current modal bid/ask price relative to best bid/ask price	-
Current mean price increment between order prices	-

TABLE B.3: Table describing features used for the model ensemble of random forests model. Price features are all normalised by the price preceding an event, while spread features are normalised by the minimum price increment allowable in the book.

Bibliography

M. Abdullah and V. Ganapathy. Neural network ensemble for financial Trend Prediction. In *TENCON Proceedings. IEEE*, volume 2, pages 157–161. Ieee, 2000. ISBN 0-7803-6355-8.

a. F. Agrawal. Sexual selection and the maintenance of sexual reproduction. *Nature*, 411(6838):692–695, 2001.

E. Alba and B. Dorronsoro. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *Evolutionary Computation, IEEE Transactions on*, 9(2):126–142, 2005.

S. S. Alexander. Price movements in speculative markets: Trends or random walks. *Industrial Management Review*, 2(2):7–26, 1961.

F. Allen and R. Karjalainen. Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, 51(2):245–271, 1999.

R. Almgren and N. Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–40, 2001.

H. Andersson and T. Britton. Stochastic epidemic models and their statistical analysis. *Springer Lecture Notes in Statistics*, 2000.

J. J. Angel, L. E. Harris, and C. S. Spatt. Equity trading in the 21st century. *Quarterly Journal of Finance*, 1(1):1–53, 2011.

R. A. Ariel. A monthly effect in stock returns. *Journal of Financial Economics*, 18(1): 161–174, 1987.

R. A. Ariel. High stock returns before holidays: Existence and evidence on possible causes. *Journal of Finance*, 45(5):1611–1626, 1990.

S. Asadi, E. Hadavandi, F. Mehmanpazir, and M. M. Nakhostin. Hybridization of evolutionary Levenberg-Marquardt neural networks and data pre-processing for stock market prediction. *Knowledge-Based Systems*, 35:245–258, 2012.

Y. At-Sahalia, P. A. Mykland, and L. Zhang. Ultra high frequency volatility estimation with dependent microstructure noise. *Journal of Econometrics*, 160(1):160–175, 2011.

M. Avellaneda and S. Stoikov. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224, 2008.

R. Axelrod and W. D. Hamilton. The evolution of cooperation. *Science*, 211(4489): 1390–1396, 1981.

C. Axioglou and S. Skouras. Markets change every day: Evidence from the memory of trade direction. *Journal of Empirical Finance*, 18(3):423–446, 2011.

M. Baron, J. Brogaard, and A. Kirilenko. The trading profits of high frequency traders. Technical report, Working paper, 2012.

L. Barone and L. While. An adaptive learning model for simplified poker using evolutionary algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, pages 153–160, 1999. ISBN 0-7803-5536-9.

F. M. Bass. A new product growth for model consumer durables. *Management Science*, 15(5):215–227, 1969.

E. Bayraktar and M. Ludkovski. Optimal trade execution in illiquid markets. *Mathematical Finance*, 21(4):681–701, 2011.

Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Y. Bengio. Practical recommendations for gradient-based training of deep architectures. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTU:437–478, 2012.

D. Bertsimas and A. W. Lo. Optimal control of execution costs. *Journal of Financial Markets*, 1(1):1–50, apr 1998.

B. Biais, P. Hillion, and C. Spatt. An empirical analysis of the limit order book and the order flow in the Paris Bourse. *Journal of Finance*, 50(5):1655–1689, 1995.

F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

A. Blumer, a. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

D. J. Bodas-Sagi, P. Fernandez-Blanco, J. I. Hidalgo, and F. J. Soltero-Domingo. A parallel evolutionary algorithm for technical market indicators optimization. *Natural Computing*, 12(2):195–207, 2013.

A. Booth, E. Gerding, and F. McGroarty. Automated trading with performance weighted random forests and seasonality. *Expert Systems with Applications*, 41(8):3651–3661, 2014a.

A. Booth, E. Gerding, and F. McGroarty. Performance-weighted ensembles of random forests for predicting price impact. *Quantitative Fiance*, (ahead-of-print):1–13, 2014b.

A. Booth, E. Gerding, and F. McGroarty. Predicting equity market impact with performance weighted ensembles of random forests. In *IEEE Symposium on Computational Intelligence for Financial Engineering and Economics*, pages 1–8. IEEE, 2014c.

A. Booth, E. Gerding, and F. McGroarty. A brave new model for a brave new market. *European Journal of Finance*, (Under-Review), 2015.

A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *IEEE 11th International Conference on Computer Vision*, pages 1–8. Ieee, 2007.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992. ISBN 089791497X.

J.-P. Bouchaud, J. D. Farmer, and F. Lillo. How markets slowly digest changes in supply and demand. In *Handbook of Financial Markets: Dynamics and Evolution*, pages 57–160. 2009.

J.-P. Bouchaud, Y. Gefen, M. Potters, and M. Wyart. Fluctuations and response in financial markets: The subtle nature of 'random' price changes. *Quantitative Finance*, 4(2):176–190, 2004.

J.-P. Bouchaud and M. Potters. *Theory of financial risk and derivative pricing: From statistical physics to risk management*. Cambridge university press, 2003.

H. Braun. On solving travelling salesman problems by genetic algorithms. In *Parallel Problem Solving from Nature*, pages 129–133. Springer, 1991.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

A. E. Bryson, Y.-C. Ho, and G. M. Siouris. Applied optimal control: Optimization, estimation, and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(9): 366–367, 1979.

M. Buchanan. It's a (stylized) fact! *Nature Physics*, 8(1):3, 2012.

C. B. Cadsby and M. Ratner. Turn-of-month and pre-holiday effects on stock returns: Some international evidence. *Journal of Banking & Finance*, 16(3):497–509, 1992.

P. Caplat, M. Anand, and C. Bauch. Symmetric competition causes population oscillations in an individual-based model of forest dynamics. *Ecological Modelling*, 211(3-4): 491–500, 2008.

A. Carbone, G. Castelli, and H. E. Stanley. Time-dependent Hurst exponent in financial time series. *Physica A: Statistical Mechanics and its Applications*, 344(1):267–271, 2004.

R. Chakrabarti. Just another day in the inter-bank foreign exchange market. *Journal of Financial Economics*, 56:2–32, 2000.

A. Chakraborti, I. M. Toke, M. Patriarca, and F. Abergel. Econophysics review: I. Empirical facts. *Quantitative Finance*, 11(7):991–1012, 2011.

D. Challet and R. Stinchcombe. Non-constant rates and over-diffusive prices in a simple model of limit order markets. *Quantitative finance*, 3(3):155–162, 2003.

B. Chen, B. M. Marlin, and J.-a. Ting. Deep Learning of Invariant Spatio-Temporal Features from Video. *NIPS Workshop*, (August):1–9, 2010.

W.-h. Chen and J.-y. Shih. Comparison of support vector machines and back propagation neural networks in forecasting the six major asian stock markets. *European Journal Of Operational Research*, 1(1):49–67, 2006.

Y. Chen, B. Yang, and A. Abraham. Flexible neural trees ensemble for stock index modeling. *Neurocomputing*, 70(4-6):697–703, jan 2007.

R. Cheng, M. Gen, and Y. Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithmsI. Representation. *Computers & industrial engineering*, 30(4): 983–997, 1996.

T. Chenoweth, Z. Obradovic, and S. Lee. Technical trading rules as prior knowledge to a neural network prediction system for the S&P 500 Index. In *IEEE Technical Applications Conference and Workshops*, pages 111–116, 1995.

C. Chiarella and G. Iori. A simulation analysis of the microstructure of double auction markets. *Quantitative Finance*, 2(5):346–353, 2002.

T. Chordia, R. Roll, and A. Subrahmanyam. Liquidity and market efficiency. *Journal of Financial Economics*, 87(2):249–268, 2008.

W. G. Christie and P. H. Schultz. Why do NASDAQ market makers avoid odd-eighth quotes? *The Journal of Finance*, 49(5):1813–1840, dec 1994.

S.-H. Chun and Y.-J. Park. Dynamic adaptive ensemble case-based reasoning: application to stock market prediction. *Expert Systems with Applications*, 28(3):435–443, apr 2005.

D. C. Cirean, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220, 2010.

D. Cliff. Minimal-intelligence agents for bargaining behaviors in market-based environments. Technical report, HP Laboratories Bristol, 1997.

D. Cliff and J. Bruten. More than zero intelligence needed for continuous double-auction trading. Technical report, Hewlett Packard, 1997.

M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, adaBoost and Bregman distances. *Engineering*, 48(3):158–169, 2000.

R. Cont. Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236, 2001.

R. Cont. Long range dependence in financial markets. In *Fractals in Engineering*, pages 159–179. Springer, 2005.

R. Cont and J.-P. Bouchaud. Herd behavior and aggregate fluctuations in financial markets. *Macroeconomic Dynamics*, 4(2):170–196, 2000.

R. Cont and A. De Larrard. Order book dynamics in liquid markets: Limit theorems and diffusion approximations. *Available at SSRN 1757861*, 2011.

R. Cont, A. Kukanov, and S. Stoikov. The price impact of order book events. *Journal of Financial Econometrics*, 12(1):47–88, 2013.

R. Cont, M. Potters, and J.-P. Bouchaud. Scaling in stock market data: Stable laws and beyond. In *Scale invariance and beyond*, pages 75–85. Springer, Berlin Heidelberg, 1997.

R. Cont, S. Stoikov, and R. Talreja. A stochastic model for order book dynamics. *Operations research*, 58(3):549–563, 2010.

G. Creamer and Y. Freund. Predicting performance and quantifying corporate governance risk for latin american adrs and banks. *Financial Engineering and Applications, MIT, Cambridge*, 2004.

G. Creamer and Y. Freund. Automated trading with boosting and expert weighting. *Quantitative Finance*, 4(10):401–420, 2010.

F. Cross. The behavior of stock prices on fridays and mondays. *Financial Analysts Journal*, 29(1):67–69, 1973.

W. Cui and A. Brabazon. An agent-based modeling approach to study price impact. In *2012 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pages 1–8. Ieee, mar 2012. ISBN 978-1-4673-1803-7.

P. Cunningham, J. Carney, and S. Jacob. Stability problems with artificial neural networks and the ensemble solution. *Artificial Intelligence in Medicine*, 20:217–225, 2000.

V. Darley and A. V. Outkin. *A NASDAQ Market Simulation: Insights on a Major Market from the Science of Complex Adaptive Systems*. 2007. ISBN 9789812700018.

R. Das, J. E. Hanson, J. O. Kephart, and G. Tesauro. Agent-human interactions in the continuous double auction. In *The Proceedings of the International Joint Conferences on Artifi- cial Intelligence (IJCAI)*, 2001.

K. A. De Jong and W. M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of mathematics and Artificial intelligence*, 5(1):1–26, 1992.

M. de la Maza and B. Tidor. An analysis of selection procedures with particular attention paid to proportional and Boltzmann selection. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 124–131. Morgan Kaufmann Publishers Inc., 1993.

M. De Luca, C. Szostek, J. Cartlidge, and D. Cliff. Studies of interactions between human traders and algorithmic trading systems. *UK Government Foresight Project*, 2011.

K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Lecture notes in computer science*, 1917:849–858, 2000.

Y. Demazeau and J.-P. Müller. *Decentralized AI*, volume 2. Elsevier, 1990.

M. A. H. Dempster, T. W. Payne, Y. Romahi, and G. W. P. Thompson. Computational learning techniques for intraday FX trading using popular technical indicators. *IEEE Transaction on Neural Networks*, 12(4):744–754, 2001.

H. Demsetz. The cost of transacting. *Quarterly Journal of Economics*, 82:33–53, 1968.

R. Díaz-Uriarte and S. Alvarez De Andrés. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006.

S. Drozdz, M. Forczek, J. Kwapien, P. Oswiecimka, and R. Rak. Stock market return distributions: From past to present. *Physica A: Statistical Mechanics and its Applications*, 383(1):59–64, 2007.

C. L. Dunis, J. Laws, and G. Sermpinis. Higher order and recurrent neural architectures for trading the EUR/USD exchange rate. *Quantitative Finance*, 11(4):615–629, apr 2011.

C. L. Dunis and V. Morrison. The economic value of advanced time series methods for modelling and trading 10-year government bonds. *European Journal of Finance*, 21 (5):333–352, 2004.

D. Easley, M. de Prado, and M. OHara. Measuring flow toxicity in a high frequency world. Technical report, unpublished Cornell University working paper (October), 2010.

D. Easley, M. M López De Prado, and M. O'Hara. The Microstructure of the "Flash Crash": Flow Toxicity, Liquidity Crashes, and the Probability of Informed Trading. *Journal of Portfolio Management*, 37(2):118–128, 2011.

Z. Eisler, J.-P. Bouchaud, and J. Kockelkoren. The price impact of order book events: market orders, limit orders and cancellations. *Quantitative Finance*, 12(9):1395–1419, sep 2012.

D. Enke and S. Thawornwong. The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications*, 29(4):927–940, nov 2005.

K. Erol, R. Levy, and J. Wentworth. Application of agent technology to traffic simulation. 2000.

European Union. Proposal for a directive of the European Parliment and of the council on markets in financial instruments repealing Directive 2004/39/EC of the European Parliament and of the Council (Recast). *Official Journal of the European Union*, 2011.

European Union. Directive 2014/65/EU of the European Parliament and of the Council of 15 May 2014 on markets in financial instruments and amending Directive 2002/92/EC and Directive 2011/61/EUNo Title. Technical report, 2014a.

European Union. Markets in Financial Instruments (MiFID): Commissioner Michel Barnier welcomes agreement in trilogue on revised European rules. *Memo*, 2014b.

E. F. Fama. Efficient capital markets: A review of theory and emperical work. *Journal of Finance*, 25(2):383–417, 1970.

E. F. Fama and M. E. Blume. Filter rules and stock-market trading. *Journal of Business*, 39(1):226–241, 1966.

J. D. Farmer and D. Foley. The economy needs agent-based modelling. *Nature*, 460: 685–686, 2009.

J. D. Farmer, A. Gerig, F. Lillo, and H. Waelbroeck. How efficiency shapes market impact. *Quantitative Finance*, 13(11):1743–1758, nov 2013.

J. D. Farmer and F. Lillo. On the origin of power-law tails in price fluctuations. *Quantitative Finance*, 4(1):7–11, 2004.

J. D. Farmer, P. Patelli, and I. I. Zovko. The predictive power of zero intelligence in financial markets. *Proceedings of the National Academy of Sciences of the United States of America*, 102(6):2254–9, feb 2005.

J. Farmer and S. Joshi. The price dynamics of common trading strategies. *Journal of Economic Behavior & Organization*, 49(2):149–171, 2002.

F. Fernández-Rodríguez. On the profitability of technical trading rules based on artificial neural networks: Evidence from the Madrid Stock Market. *Economics Letters*, 69(1): 89–94, 2000.

M. J. Fields. Stock prices: A problem in verification. *The Journal of Business of the University of Chicago*, 4(4):415–418, 1931.

M. J. Fields. Security prices and stock exchange holidays in relation to short selling. *The Journal of Business of the University of Chicago*, 7(4):328–338, 1934.

D. B. Fogel. An introduction to simulated evolutionary optimization. *Neural Networks, IEEE Transactions on*, 5(1):3–14, 1994.

T. Foucault. Order flow composition and trading costs in a dynamic limit order market. *Journal of Financial Markets*, 2(2):99–134, 1999.

K. R. French. Stock returns and the weekend effect. *Journal of Financial Economics*, 8 (1):55–69, 1980.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119–139, 1997.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

J. Gatheral. No-dynamic-arbitrage and market impact. *Quantitative Finance*, 10:749–759, 2010.

J. Geanakoplos, R. Axtell, J. Farmer, P. Howitt, B. Conlee, J. Goldstein, M. Hendrey, N. Palmer, and C.-Y. Yang. Getting at systemic risk via an agent-based model of the housing market. 2012.

S. Gjerstad and J. Dickhaut. Price formation in double auctions. *Games and Economic Behavior*, 22(1):1–29, 1998.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *AISTATS*, 9:249–256, 2010.

X. Glorot, A. Bordes, and Y. Bengio. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. *Proceedings of the 28th International Conference on Machine Learning*, (1):513–520, 2011.

D. Gode and S. Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 101(1):119–137, 1993.

D. E. Goldberg and R. Lingle. Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications*, pages 154–159. Lawrence Erlbaum Associates, Publishers, 1985.

D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, 167(1):77–95, 2005.

P. Gopikrishnan, V. Plerou, L. A. Nunes Amaral, M. Meyer, and H. E. Stanley. Scaling of the distribution of fluctuations of financial market indices. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, 60(5):5305–5316, 1999.

P. Gopikrishnan, M. Meyer, L. A. N. Amaral, and H. E. Stanley. Inverse cubic law for the distribution of stock price variations. *The European Physical Journal B-Condensed Matter and Complex Systems*, 3(2):139–140, 1998.

V. Grimm, E. Revilla, U. Berger, F. Jeltsch, W. M. Mooij, S. F. Railsback, H.-H. Thulke, J. Weiner, T. Wiegand, and D. L. DeAngelis. Pattern-oriented modeling of agent-based complex systems: lessons from ecology. *Science (New York, N.Y.)*, 310 (5750):987–91, nov 2005.

G. F. Gu, W. Chen, and W. X. Zhou. Empirical distributions of Chinese stock returns at different microscopic timescales. *Physica A: Statistical Mechanics and its Applications*, 387(2):495–502, 2008.

G.-F. Gu and W.-X. Zhou. Emergence of long memory in stock volatility from a modified Mike-Farmer model. *EPL (Europhysics Letters)*, 86(4):48002, 2009.

R. Guobuzaite, K. Byrne, and A. Freyre-Sanders. A review of trading cost models. *The Journal of Investing*, 13(3):93–115, 2004.

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

S. Gutta and H. Wechsler. Face recognition using hybrid classifier systems. In *Neural Networks, 1996., IEEE International Conference on*, volume 2, pages 1017–1022. IEEE, 1996. ISBN 0780332105.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

L. K. Hansen and P. Salamon. Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(10):993–1001, 1990.

J. Hasbrouck. Measuring the information content of stock trades. *The Journal of Finance*, 46:179–207, 1991.

J. Hasbrouck and G. Saar. Low-latency trading. *Journal of Financial Markets*, 16: 646–679, 2013.

N. Hautsch and R. Huang. The market impact of a limit order. *Journal of Economic Dynamics and Control*, 36:501–522, 2012.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *Ieee Signal Processing Magazine*, (November): 82–97, 2012.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–54, 2006.

H. S. Hippert, C. E. Pedreira, and R. C. Souza. Neural networks for short-term load forecasting: A review and evaluation. *Power Systems, IEEE Transactions on*, 16(1): 44–55, 2001.

T. K. Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 278 – 282, 1995. ISBN 0818671289.

J. H. Holland. *Adaptation in natural and artificial systems.* Number 53. University of Michigan Press, 1975. ISBN 0262581116.

H. Hong and J. Yu. Gone fishin': Seasonality in trading activity and asset prices. 2009.

C. Hopman. Do supply and demand drive stock prices? *Quantitative Finance*, 7(1): 37–53, 2007.

J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 82–87. Ieee, 1994.

C. R. Houck, J. Joines, and M. G. Kay. A genetic algorithm for function optimization: a Matlab implementation. *NCSU-IE TR*, 95(09), 1995.

H. Huang and A. N. Kercheval. A generalized birthdeath stochastic model for high-frequency order book dynamics, 2012.

W. Huang, Y. Nakamori, and S.-Y. Wang. Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513–2522, oct 2005.

D. H. Hubel and T. N. Wiesel. Receptive field and functional architechture in two nonstriate visual areas (18 and 19) of the cat. *Journal of neurophysiology*, 28:229–289, 1965.

G. Huberman and W. Stanzl. Price manipulation and quasi-arbitrage, 2004.

H. P. N. Hughes, C. W. Clegg, M. a. Robinson, and R. M. Crowder. Agent-based modelling and simulation: The potential contribution to organizational psychology. *Journal of Occupational and Organizational Psychology*, 85:487–502, 2012.

E. Hunsader. May 6'th 2010 Flash Crash Analysis, 2012.

a. G. Ivakhnenko. Polynomial Theory of Complex Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(4):30–37, 1971.

P. K. Jain. Financial market design and the equity premium: Electronic versus floor trading. *The Journal of Finance*, 60(6):2955–2985, 2005.

P. C. Jain and G.-H. Joh. The dependence between hourly prices and trading volume. *The Journal of Financial and Quantitative Analysis*, 23:269–283, 1988.

N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998.

O. B. Jennings and J. E. Reed. An overloaded multiclass FIFO queue with abandonments. *Operations Research*, 60(5):1282–1295, 2012.

N. Johnson, G. Zhao, E. Hunsader, H. Qi, N. Johnson, J. Meng, and B. Tivnan. Abrupt rise of new machine ecology beyond human response time. *Scientific reports, Nature Publishing Group*, 3:2627, jan 2013.

T. Joyce, J. Nazarali, A. Levitt, F. Tomczyk, L. Harris, C. Spatt, D. Mathisson, G. Katz, P. Stevens, E. Sirri, J. Bethel, D. Niederauer, J. Giesea, and B. Mock. *Current perspectives on modern equity markets: A collection of essays by financial industry experts*. Mighty Media, Inc., 2010.

I. Kaastra and M. Boyd. Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 10(3):215–236, 1996.

K. Kamijo and T. Tanigawa. Stock price pattern recognition - A recurrent neural network approach. In *International Joint Conference on Neural Networks*, pages 215–221. Ieee, 1990.

E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of Neural Science*, volume 4. 2000. ISBN 0838577016.

L.-J. Kao, C.-C. Chiu, C.-J. Lu, and J.-L. Yang. Integration of nonlinear independent component analysis and support vector regression for stock price forecasting. *Neurocomputing*, 99:534–542, 2013.

Y. Kara, M. Acar Boyacioglu, and Ö. K. Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, 38(5): 5311–5319, 2011.

A. Kazem, E. Sharifi, F. K. Hussain, M. Saberi, and O. K. Hussain. Support vector regression with chaos-based firefly algorithm for stock market price forecasting. *Applied Soft Computing*, 13(2):947–958, 2013.

M. Kearns and L. Ortiz. The Penn-Lehman automated trading project. *IEEE Intelligent Systems*, pages 22–31, 2003.

D. B. Keim and A. Madhavan. Anatomy of the trading process empirical evidence on the behavior of institutional traders. *Journal of Financial Economics*, 37(3):371–398, 1995.

K.-j. Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307–319, sep 2003.

K.-j. Kim. Artificial neural networks with feature transformation based on domain knowledge for the prediction of stock index futures. *International Journal of Intelligent Systems in Accounting and Finance Management*, 12(3):167–176, 2004.

T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka. Stock market prediction system with modular neural networks. *1990 IJCNN International Joint Conference on Neural Networks*, pages 1–6 vol.1, 1990.

A. Kirilenko, A. S. Kyle, M. Samadi, and T. Tuzun. The flash crash: The impact of high frequency trading on an electronic market. *Available at SSRN 1686004*, 2014.

A. S. Kondrashov. Deleterious mutations and the evolution of sexual reproduction. *Nature*, 336(6198):435–440, 1988.

J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

A. Krizhevsky and G. E. Hinton. Using very deep autoencoders for content-based image retrieval. *Proceedings of 19th European Symposium on Artificial Neural Networks (ESANN)*, pages 489–494, 2011.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012.

A. Kulkarni. Applications of neural networks to stock market prediction. Technical report, 1996.

M. Kumar and M. Thenmozhi. Forecasting Stock Index Movement : a Comparision of Support Vector Machines and Random Forest. *Forest, Indian Institute of Capital Markets 9th Capital Markets Conference Paper.*, pages 1–16, 2005.

R. Kuo, C. Chen, and Y. Hwang. An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. *Fuzzy Sets and Systems*, 118(1):21–45, 2001.

A. S. Kyle. Continuous auctions and insider trading. *Econometrica: Journal of the Econometric Society*, 53:1315, 1985.

J. Lakonishok and S. Smidt. Are seasonal anomalies real? A ninety-year perspective. *Review of Financial Studies*, 1(4):403–425, 1988.

B. Lariviere and D. Vandenpoel. Predicting customer retention and profitability by using random forests and regression forests techniques. *Expert Systems with Applications*, 29(2):472–484, 2005.

P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

B. Lebaron. *An evolutionary bootstrap method for selecting dynamic trading strategies.* Social Systems Research Institute, University of Wisconsin, Norwell, MA, 1998.

H. Lee, Y. Largman, P. Pham, and A. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in Neural Information Processing Systems*, pages 1096–1104, 2009.

D. J. Leinweber. Stupid data miner tricks: Overfitting the S&P 500. *The Journal of Investing*, 16(1):15–22, 2007.

F. H. F. Leung, H.-K. Lam, S.-H. Ling, and P. K. S. Tam. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *Neural Networks, IEEE Transactions on*, 14(1):79–88, 2003.

A. Liaw and M. Wiener. Classification and regression by random forest. *R news*, 2(3): 18–22, 2002.

F. Lillo, J. D. Farmer, and R. N. Mantegna. Master curve for price impact function. *Nature*, 421(6919):129–130, 2003.

F. Lillo and J. D. Farmer. The long memory of the efficient market. *Studies in Nonlinear Dynamics & Econometrics*, 8(3), 2004.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. In *30th Annual Symposium on Foundations of Computer Science*, pages 256–261, 1989. ISBN 0818619821.

Y. Liu, P. Cizeau, M. Meyer, C.-K. Peng, and H. Eugene Stanley. Correlations in economic time series. *Physica A: Statistical Mechanics and its Applications*, 245(3): 437–440, 1997.

A. Lo and A. MacKinlay. *A non-random walk down Wall Street*. Princeton University Press, Princeton, NJ, 2001.

S. Mahfoud and G. Mani. Applied artificial intelligence: Financial forecasting using genetic algorithms. *Applied Artificial Intelligence: An International Journal*, 10(6): 543–566, 1996.

J. Mao. A case study on bagging, boosting and basic ensembles of neural networks for OCR. In *The 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence.*, volume 3, pages 1828–1833. IEEE, 1998. ISBN 0780348591.

M. Maragoudakis and D. Serpanos. Towards stock market data mining using enriched random forests from textual resources and technical indicators. In H. Papadopoulos, A. Andreou, and M. Bramer, editors, *Artificial Intelligence Applications and Innovations*, volume 339 of *IFIP Advances in Information and Communication Technology*, pages 278–286. Springer, 2010.

J. Martens. Deep learning via Hessian-free optimization. *27th International Conference on Machine Learning*, 951:735–742, 2010.

I. Mastromatteo, B. Toth, and J.-P. Bouchaud. Agent-based models for latent liquidity and concave price impact. *Physical Review E*, 89(4):042805, 2014.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

T. H. McInish and R. A. Wood. An analysis of intraday patterns in bid/ask spreads for NYSE stocks. *The Journal of Finance*, 47:753–764, 1992.

G. Meissner and N. Kawano. Capturing the volatility smile of options on high-tech stocks: A combined GARCH-neural network approach. *Journal of Economics and Finance*, 25(3):276–292, 2001.

L. M. Menezes and N. Y. Nikolaev. Forecasting with genetically programmed polynomial neural networks. *International Journal of Forecasting*, 22(2):249–265, 2006.

A. J. Menkveld and B. Z. Yueshen. Anatomy of the Flash Crash. *SSRN Electronic Journal*, 2013.

S. Mike and J. D. Farmer. An empirical behavioral model of liquidity and volatility. *Journal of Economic Dynamics and Control*, 32(1):200–234, 2008.

B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3):193–212, 1995.

J. F. Miller and P. Thomson. Cartesian genetic programming. In *Genetic Programming*, pages 121–132. Springer, 2000.

M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*, volume 165. 1969. ISBN 0262631113.

S. Mitra. Improving accuracy of option price estimation using artificial neural networks. In *Indian Institure of Capital Markets*, 2006.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv: ...*, pages 1–9, 2013.

D. B. Nelson. Conditional heteroskedasticity in asset returns: A new approach. *Econometrica: Journal of the Econometric Society*, pages 347–370, 1991.

M. a. Niazi and A. Hussain. A novel agent-based simulation framework for sensing in complex adaptive environments. *IEEE Sensors Journal*, 11(2):404–412, 2011.

J. Noble. Finding Robust Texas Hold'em Poker Strategies Using Pareto Coevolution and Deterministic Crowding. pages 233–239, 2002.

W. Nuij, V. Milea, F. Hogenboom, F. Frasincar, and U. Kaymak. An automated framework for incorporating news into stock trading strategies. *IEEE Transactions on Knowledge and Data Engineering*, 2013.

NYSE EURONEXT. Fees for non-display use of NYSE EURONEXT information. Technical report, 2013.

A. a. Obizhaeva and J. Wang. Optimal trading strategy and supply/demand dynamics. *Journal of Financial Markets*, 16(1):1–32, feb 2013.

C. Oesch. An agent-based model for market impact. In *2014 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr)*, 2014.

R. R. Officer. The distribution of stock returns. *Journal of the american statistical association*, 67(340):807–812, 1972.

H. V. D. Parunak, R. Savit, and R. L. Riolo. Agent-based modeling vs. equation-based modeling: A case study and users guide. In *Multi-agent systems and agent-based simulation*, pages 10–25. Springer, 1998.

C. K. Peng, S. V. Buldyrev, S. Havlin, M. Simons, H. E. Stanley, and A. L. Goldberger. Mosaic organization of DNA nucleotides. *Physical Review E*, 49:1685–1689, 1994.

V. Plerou, P. Gopikrishnan, X. Gabaix, and H. E. Stanley. Quantifying stock-price response to demand fluctuations. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 66:027104, 2002.

V. Plerou and H. E. Stanley. Stock return distributions: Tests of scaling and universality from three distinct stock markets. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 77(3):037101, 2008a.

V. Plerou and H. E. Stanley. Stock return distributions: Tests of scaling and universality from three distinct stock markets. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 77(3):037101, 2008b.

M. Potters and J.-P. Bouchaud. More statistical properties of order books and price impact. *Physica A: Statistical Mechanics and its Applications*, 324(1):133–140, 2003.

A. M. Prasad, L. R. Iverson, and A. Liaw. Newer classification and regression tree techniques: Bagging and random forests for ecological prediction. *Ecosystems*, 9(2): 181–199, 2006.

S. Predoiu, G. Shaikhet, and S. Shreve. Optimal execution in a general one-sided limit-order book. *SIAM Journal on Financial Mathematics*, 2(1):183–212, 2011.

T. Preis, S. Golke, W. Paul, and J. J. Schneider. Multi-agent-based order book model of financial markets. *Europhysics Letters (EPL)*, 75(3):510–516, 2006.

T. Preis, S. Golke, W. Paul, and J. J. Schneider. Statistical analysis of financial returns for a multiagent order book model of asset trading. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 76(1):016108, 2007.

Q. Qin, Q.-G. Wang, J. Li, and S. S. Ge. Linear and nonlinear trading models with gradient boosted random forests and application to singapore stock market. *Journal of Intelligent Learning Systems and Applications*, 5:1–10, 2013.

A.-P. Refenes, A. Zapranis, and G. Francis. Modelling stock returns in the framework of APT: A comparative study with regression models. In *Neural Networks in the Capital Markets*, pages 101–125. John Whiley and Sons, 1995.

C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34, aug 1987.

J. Robinson, S. Sinton, and Y. Rahmat-Samii. Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In *Antennas and Propagation Society International Symposium, 2002. IEEE*, volume 1, pages 314–317. IEEE, 2002.

P. N. Rodríguez and S. Sosvilla-Rivero. Using machine learning algorithms to find patterns in stock prices. *Documento de Trabajo*, page 12, 2006.

R. J. Rogalski. New findings regarding dayoftheweek returns over trading and nontrading periods: A note. *The Journal of Finance*, 39(5):1603–1614, 1984.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1 of *Parallel Distributed Processing: {E}xplorations in the Microstructure of Cognition, {V}ol. ˜1: {F}oundations*, chapter 8, pages 318–362. MIT Press, 1986. ISBN 026268053X.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 1995. ISBN 9780137903955.

J. Rust, R. Palmer, and J. H. Miller. Behaviour of trading automata in a computerized double auction market. Santa Fe Institute, 1992.

P. a. Samuelson. Interactions between the Multiplier Analysis and the Principle of Acceleration. *The Review of Economics and Statistics*, 21(2):75–78, 1939.

R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

T. Schelling. Dynamic models of segregation. *Journal of mathematical sociology*, 1(2): 143–186, 1971.

SEC and CFTC. Findings regarding the market events of May 6, 2010. Technical report, Report of the Staffs of the CFTC and SEC to the Joint Advisory Committee on Emerging Regulatory Issues, 2010.

A. F. Serban. Combining mean reversion and momentum trading strategies in foreign exchange markets. *Journal of Banking and Finance*, 34:2720–2727, 2010.

G. Sermpinis, J. Laws, A. Karathanasopoulos, and C. L. Dunis. Forecasting and trading the EUR/USD exchange rate with gene expression and psi sigma neural networks. *Expert Systems with Applications*, feb 2012.

T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:994–1000, 2005.

S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, pages 591–611, 1965.

K. Shin, K.-j. Kim, and I. Han. Financial data mining using genetic algorithms technique: Application to KOSPI 200. In *Proceedings of the Korea inteligen Information System Society Conference*, volume 200, pages 113–122, 1998.

E. Smith, J. Farmer, L. Gillemot, and S. Krishnamurthy. Statistical theory of the continuous double auction. *Quantitative Finance*, 3(6):481–514, dec 2003.

V. L. Smith. An experimental study of competitive market behavior. *Journal of Political Economy*, 70(2):111–137, 1962.

M. Srinivas and L. M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(4): 656–667, 1994.

H. E. Stanley, V. Plerou, and X. Gabaix. A statistical physics view of financial fluctuations: Evidence for scaling and universality. *Physica A: Statistical Mechanics and its Applications*, 387(15):3967–3981, 2008.

S. Stoikov and R. Waeber. Optimal asset liquidation using limit order book information. *Available at SSRN 2113827*, pages 1–33, 2012.

F. Sun, Y. Tan, H. Liu, R. d. A. Araújo, and T. A. Ferreira. A Morphological-Rank-Linear evolutionary method for stock market prediction. *Information Sciences*, 237: 3–17, 2013.

J. Tang, H. Enderling, S. Becker-Weimann, C. Pham, A. Polyzos, C.-Y. Chen, and S. V. Costes. Phenotypic transition maps of 3D breast acini obtained by imaging-guided agent-based modeling. *Integrative biology : quantitative biosciences from nano to macro*, 3(4):408–21, 2011.

F. E. H. Tay and L. J. Cao. Modified support vector machines in financial time series forecasting. *Neurocomputing*, 48(1):847–861, 2002.

F. E. H. Tay and L. Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317, 2001.

L. Tesfatsion and K. L. Judd. *Handbook of Computational Economics*, volume 2. 2006. ISBN 9780444512536.

S. Thurner, J. D. Farmer, and J. Geanakoplos. Leverage causes fat tails and clustered volatility. *Quantitative Finance*, 12(5):695–707, 2012.

J. L. Ticknor. A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, 40(14):5501–5506, 2013.

B. Tóth, Y. Lempérière, C. Deremble, J. de Lataillade, J. Kockelkoren, and J.-P. Bouchaud. Anomalous price impact and the critical nature of liquidity in financial markets. *Phys. Rev. X*, 1(2):021006, oct 2011.

R. Trippi and D. DeSieno. Trading equity index futures with a neural network. *Journal of Portfolio Management*, 19(1):27–33, 1992.

C.-F. Tsai, Y.-C. Lin, D. C. Yen, and Y.-M. Chen. Predicting stock returns by classifier ensembles. *Applied Soft Computing*, 11(2):2452–2459, mar 2011.

V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

J. Z. Wang, J. J. Wang, Z. G. Zhang, and S. P. Guo. Forecasting stock indices with back propagation neural network. *Expert Systems with Applications*, 38(11):14346–14355, 2011.

P. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard, 1974.

H. White and S. Diego. Economic prediction using neural networks: The case of IBM daily stock returns. In *IEEE International Conference on Neural Networks*, pages 451–458. IEEE, 1988. ISBN 0780309995.

D. Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel computing*, 14(3):347–361, 1990.

M. B. Wilk and R. Gnanadesikan. Probability plotting methods for the analysis for the analysis of data. *Biometrika*, 55(1):1–17, 1968.

D. Witkowska and D. Marcinkiewicz. Construction and evaluation of trading systems: Warsaw index futures. *International Advances in Economic Research.*, 11:83–92, 2005.

World Bank. Data retrieved from World Bank http://data.worldbank.org/indicator/CM.MKT.LCAP.CD. 2012.

A. H. Wright. Genetic algorithms for real parameter optimization. In *Foundations of Genetic Agorithms*, pages 205–218. 1991. ISBN 1-55860-170-8.

Y. Xu, Z. Li, and L. Luo. A study on feature selection for trend prediction of stock trading price. *2013 International Conference on Computational and Information Sciences*, (2):579–582, jun 2013.

H. Yang, W. Zhou, L. Lu, and Z. Fang. Optimal sizing method for stand-alone hybrid solar–wind system with LPSP technology by using genetic algorithm. *Solar energy*, 82(4):354–367, 2008.

K. Zbikowski and P. Grzegorzewski. Stock trading with random forests, trend detection tests and force index volume indicators. In *Artificial Intelligence and Soft Computing*, volume I, pages 441–452. Springer, 2013.

S. Zemke. On developing a financial prediction system: Pitfalls and possibilities. In *Proceedings of DMLL-2002 Workshop at ICML-2002*, Sydney, 2002.

Z.-H. Zhou, Y. Jiang, Y.-B. Yang, and S.-F. Chen. Lung cancer cell identification based on artificial neural network ensembles. *Artificial Intelligence in Medicine*, 24(1):25–36, 2002.