

https://dspace.org/

Agent-based learning for pattern matching in high-frequency trade data

Loonat, Fayyaaz

2017

http://hdl.handle.net/10539/23537

Loonat, Fayyaaz (2017) Agent-based learning for pattern matching in high-frequency trade data, University of the Witwatersrand, Johannesburg, http://hdl.handle.net/10539/23537

Downloaded from a DSpace repository. https://dspace.org/

UNIVERSITY OF THE WITWATERSRAND

SCHOOL OF COMPUTER SCIENCE AND APPLIED MATHEMATICS

Agent-Based Learning for Pattern Matching in High-Frequency Trade Data

A dissertation submitted to the Faculty of Science, University of the Witwatersrand, in fulfilment of the requirements for the degree of Master of Science.

Author:
FAYYAAZ LOONAT
316324

Supervisor:
PROF. T. GEBBIE.

May 14, 2017



WITS
UNIVERSITY

Abstract

Previous research of sequential investment strategies for portfolio selection have shown that there are strategies that exist that can beat the best stock in the market. In this dissertation, an algorithm is presented that uses a nearest neighbour approach similar to the one used by Györfi *et al* [20, 21, 22]. The approach is however extended to include zero-cost portfolios and uses a quadratic approximation, instead of an optimisation step, to determine how capital should be allocated in the portfolio based on the neighbours that have been found. A portfolio that results in an increase in the investor's capital and compares favourably to certain benchmarks, such as the best stock, indicates that there are patterns in the time series data. Other features of the algorithm presented is to allow for the data to be clustered by a selection of stocks or partitioned based on time. The algorithm is tested on synthetic datasets that depict different market types and is shown to accurately determine trends in the data. The algorithm is then tested on real data from the New York Stock Exchange (NYSE) and data from the Johannesburg Stock Exchange (JSE). The results of the algorithm from the real datasets are compared to implemented versions of past strategies from the literature and compares favourably.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Tim Gebbie, for his continuous support and guidance during my MSc journey. I also thank him for providing me with this topic and for the initial research and code development that was required for this topic. I am truly grateful to have had Prof. Tim Gebbie as my supervisor for my MSc study.

I would also like to thank Prof Turgay Celik for his input in various conversations. I would like to thank Prof Diane Wilcox and the entire QuERI Lab team for the helpful discussions and suggestions that they offered. I extend this thanks to the entire Computer Science and Applied Mathematics department for their hospitality and fantastic academic environment that they have created.

My sincere thanks go to Mr. Shunmuga Pillay, Mr. Brian Maistry and Mr. Mohsin Desai for their assistance in setting up a MATLAB Distributed Computing Server on the Hydra cluster and for all the help that they offered during my time in the Mathematical Sciences Labs.

I would like to thank the National Research Foundation (NRF) for the financial assistance provided towards this research[1].

Finally, I would like to thank my parents, my sisters and my friends for their moral support and motivation offered during my post graduate studies.

[1]This research is not necessarily attributed to the NRF

Declaration

I, Fayyaaz Loonat, hereby declare that this dissertation titled *Agent-Based Learning for Pattern Matching in High-Frequency Trade Data* is my own work except where reference has been made to published literature. I further declare that this dissertation has not been previously submitted for any degree at any university.

Candidate: Fayyaaz Loonat

Signature: _____

Date: _____

Publications and Pre-Prints

There is one pre-print available for this research at arxiv.org [18] and is titled “Learning zero-cost portfolio selection with pattern matching”.

Contents

1 Introduction **1**

2 Literature Review **3**

2.1 Universal Portfolios 3

2.1.1 Recovering Prior Results 4

2.2 Györfi *et al* Algorithms 5

2.2.1 Histogram Strategy 6

2.2.2 Kernel Strategy 7

2.2.3 Nearest Neighbour Strategy 7

2.2.4 First Neighbour Strategy 8

2.2.5 Recovering Prior Results 8

2.3 Thick Modelling 9

2.4 "Can We Learn to Beat the Best Stock" 10

2.5 "Online Portfolio Selection: A Survey" 10

3 Algorithm **12**

3.1 The Algorithm and Mathematical Model 12

3.2 Portfolio Selection 15

3.3 Pattern Matching 17
3.4 Partitioning 17
3.5 Clustering 18

4 Data **21**

4.1 Synthetic Data 21
4.1.1 Synthetic Data Case 1 (SDC 1) 21
4.1.2 Synthetic Data Case 2 (SDC 2) 22
4.1.3 Synthetic Data Case 3 (SDC 3) 22
4.1.4 Synthetic Data Case 4 (SDC 4) 22
4.2 Real Data 23
4.2.1 NYSE Data 23
4.2.2 NYSE Merged Data 23
4.2.3 JSE OHLC Data 24
4.2.4 JSE Intraday Data 26

5 Results **27**

5.1 Implementation 27
5.1.1 Parameters Chosen 27
5.1.2 Comparison to Results from the Literature 27
5.1.3 Implementation Process 28
5.1.4 Pattern Class 28
5.2 Synthetic Data 29
5.2.1 Wealth Achieved from Synthetic Datasets 30

5.2.2 Kolmogorov-Smirnov Tests 30
5.3 NYSE Data 32
5.3.1 Iroquois Brands Ltd. and Kin Ark Corp. 33
5.3.2 Commercial Metals and Mei Corp. 34
5.3.3 Commercial Metals and Kin Ark Corp. 35
5.3.4 IBM and Coca-Cola 36
5.3.5 All 36 Stocks 36
5.4 NYSE Merged Data 38
5.4.1 Commercial Metals and Kin Ark Corp. 38
5.4.2 IBM and Coca-Cola 39
5.4.3 All 23 Stocks 40
5.5 JSE Daily Data 41
5.5.1 ANG and AGL 42
5.5.2 SBK and FSR 46
5.5.3 TBS and WHL 50
5.5.4 10 Stocks 53
5.5.5 20 Stocks 57
5.5.6 30 Stocks 61
5.6 JSE Intraday Data 66
5.6.1 ANG and AGL 67
5.6.2 SBK and FSR 68
5.6.3 TBS and WHL 69
5.6.4 MTN and VOD 70

5.6.5 10 Stocks 71

5.6.6 20 Stocks 72

5.6.7 30 Stocks 73

5.6.8 20 Stocks Clustered 75

6 Conclusion **77**

6.1 Overall Summary 77

6.2 Further Work 78

A MATLAB Code **83**

A.1 Pattern Class 83

A.2 Universal Portfolios 101

List of Figures

2.1 Wealth gained from the Universal Portfolios strategy 4

2.2 Universal Portfolio Portfolio Allocation 5

2.3 Price relative partitioning for the Histogram Strategy 6

4.1 NYSE Data Screenshot 24

4.2 JSE OHLC data sample 24

4.3 JSE *close-to-close* data sample 25

5.1 Process Flow Diagram 28

5.2 NYSE Data: Wealth Gained from Iroquois Brands Ltd. and Kin Ark Corp. 33

5.3 NYSE Data: Wealth Gained from Commercial Metals and Mei Corp. 34

5.4 NYSE Data: Wealth Gained from Commercial Metals and Kin Ark Corp. 35

5.5 NYSE Data: Wealth Gained from IBM and Coca-Cola 36

5.6 NYSE Data: Wealth Gained from All 36 Stocks 37

5.7 NYSE Merged Data: Wealth Gained from Commercial Metals and Kin Ark Corp. . . . 39

5.8 NYSE Merged Data: Wealth Gained from IBM and Coca-Cola 40

5.9 NYSE Merged Data: Wealth Gained from All 23 Stocks 41

5.10 JSE *close-to-close* Data: Wealth Gained from ANG and AGL 42

5.11 JSE *close-to-open* Data: Wealth Gained from ANG and AGL 43

5.12 JSE *open-to-close* Data: Wealth Gained from ANG and AGL 44

5.13 JSE *open-to-open* Data: Wealth Gained from ANG and AGL 45

5.14 JSE OHLC Dataset Comparison: Wealth Gained from ANG and AGL 45

5.15 JSE *close-to-close* Data: Wealth Gained from SBK and FSR 46

5.16 JSE *close-to-open* Data: Wealth Gained from SBK and FSR 47

5.17 JSE *open-to-close* Data: Wealth Gained from SBK and FSR 48

5.18 JSE *open-to-open* Data: Wealth Gained from SBK and FSR 49

5.19 JSE OHLC Dataset Comparison: Wealth Gained from SBK and FSR 49

5.20 JSE *close-to-close* Data: Wealth Gained from TBS and WHL 50

5.21 JSE *close-to-open* Data: Wealth Gained from TBS and WHL 51

5.22 JSE *open-to-close* Data: Wealth Gained from TBS and WHL 52

5.23 JSE *open-to-open* Data: Wealth Gained from TBS and WHL 53

5.24 JSE OHLC Dataset Comparison: Wealth Gained from TBS and WHL 53

5.25 JSE *close-to-close* Data: Wealth Gained from 10 Stocks 54

5.26 JSE *close-to-open* Data: Wealth Gained from 10 Stocks 55

5.27 JSE *open-to-close* Data: Wealth Gained from 10 Stocks 56

5.28 JSE *open-to-open* Data: Wealth Gained from 10 Stocks 57

5.29 JSE OHLC Dataset Comparison: Wealth Gained from 10 Stocks 57

5.30 JSE *close-to-close* Data: Wealth Gained from 20 Stocks 58

5.31 JSE *close-to-open* Data: Wealth Gained from 20 Stocks 59

5.32 JSE *open-to-close* Data: Wealth Gained from 20 Stocks 60

5.33 JSE *open-to-open* Data: Wealth Gained from 20 Stocks 61

5.34 JSE OHLC Dataset Comparison: Wealth Gained from 20 Stocks 61

5.35 JSE *close-to-close* Data: Wealth Gained from 30 Stocks 63

5.36 JSE *close-to-open* Data: Wealth Gained from 30 Stocks 64

5.37 JSE *open-to-close* Data: Wealth Gained from 30 Stocks 65

5.38 JSE *open-to-open* Data: Wealth Gained from 30 Stocks 66

5.39 JSE OHLC Dataset Comparison: Wealth Gained from 30 Stocks 66

5.40 JSE Intraday Data: Wealth Gained from ANG and AGL 68

5.41 JSE Intraday Data: Wealth Gained from SBK and FSR 69

5.42 JSE Intraday Data: Wealth Gained from TBS and WHL 70

5.43 JSE Intraday Data: Wealth Gained from MTN and VOD 71

5.44 JSE Intraday Data: Wealth Gained from 10 Stocks 72

5.45 JSE Intraday Data: Wealth Gained from 20 Stocks 73

5.46 JSE Intraday Data: Wealth Gained from 30 Stocks 74

5.47 JSE Intraday Data: Wealth Gained from 20 Stocks with Clusters 76

List of Tables

2.1 Recovering the prior results of the Györfi *et al* [21] Nearest Neighbour algorithm using the MATLAB code in Appendix A.1 9

4.1 Summary of Random Datasets 23

4.2 Description of the fields available in the CSV file. 26

4.3 Summary of Real Datasets 26

5.1 Properties contained in the pattern class 29

5.2 Methods contained in the pattern class 29

5.3 Wealth Gained (S) from Investing in Synthetic Data 30

5.4 Comparison of Average p Values of Wealth Gained (S) from the Active Portfolio . . 31

5.5 Comparison of Average p Values of Wealth Gained (S) from the Absolute Portfolio . 31

5.6 Comparison of p Values of Wealth Gained (S) from the Active Portfolio 32

5.7 Comparison of p Values of Wealth Gained (S) from the Absolute Portfolio 32

5.8 NYSE Data: Wealth Gained from Iroquois Brands Ltd. and Kin Ark Corp. 33

5.9 NYSE Data: Wealth Gained from Commercial Metals and Mei Corp. 34

5.10 NYSE Data: Wealth Gained from Commercial Metals and Kin Ark Corp. 35

5.11 NYSE Data: Wealth Gained from IBM and Coca-Cola 36

5.12 NYSE Data: Wealth Gained from All 36 Stocks 37

5.13 NYSE Merged Data: Wealth Gained from Commercial Metals and Kin Ark Corp. . . 38

5.14 NYSE Merged Data: Wealth Gained from IBM and Coca-Cola 39

5.15 NYSE Merged Data: Wealth Gained from All 23 Stocks 41

5.16 JSE *close-to-close* Data: Wealth Gained from ANG and AGL 42

5.17 JSE *close-to-open* Data: Wealth Gained from ANG and AGL 43

5.18 JSE *open-to-close* Data: Wealth Gained from ANG and AGL 43

5.19 JSE *open-to-open* Data: Wealth Gained from ANG and AGL 44

5.20 JSE *close-to-close* Data: Wealth Gained from SBK and FSR 46

5.21 JSE *close-to-open* Data: Wealth Gained from SBK and FSR 47

5.22 JSE *open-to-close* Data: Wealth Gained from SBK and FSR 47

5.23 JSE *open-to-open* Data: Wealth Gained from SBK and FSR 48

5.24 JSE *close-to-close* Data: Wealth Gained from TBS and WHL 50

5.25 JSE *close-to-open* Data: Wealth Gained from TBS and WHL 51

5.26 JSE *open-to-close* Data: Wealth Gained from TBS and WHL 51

5.27 JSE *open-to-open* Data: Wealth Gained from TBS and WHL 52

5.28 JSE *close-to-close* Data: Wealth Gained from 10 Stocks 54

5.29 JSE *close-to-open* Data: Wealth Gained from 10 Stocks 55

5.30 JSE *open-to-close* Data: Wealth Gained from 10 Stocks 55

5.31 JSE *open-to-open* Data: Wealth Gained from 10 Stocks 56

5.32 JSE *close-to-close* Data: Wealth Gained from 20 Stocks 58

5.33 JSE *close-to-open* Data: Wealth Gained from 20 Stocks 59

5.34 JSE *open-to-close* Data: Wealth Gained from 20 Stocks 59

5.35 JSE *open-to-open* Data: Wealth Gained from 20 Stocks 60

5.36 JSE *close-to-close* Data: Wealth Gained from 30 Stocks 62

5.37 JSE *close-to-open* Data: Wealth Gained from 30 Stocks 63

5.38 JSE *open-to-close* Data: Wealth Gained from 30 Stocks 64

5.39 JSE *open-to-open* Data: Wealth Gained from 30 Stocks 65

5.40 JSE Intraday Data: Wealth Gained from ANG and AGL 67

5.41 JSE Intraday Data: Wealth Gained from SBK and FSR 68

5.42 JSE Intraday Data: Wealth Gained from TBS and WHL 69

5.43 JSE Intraday Data: Wealth Gained from MTN and VOD 70

5.44 JSE Intraday Data: Wealth Gained from 10 Stocks 71

5.45 JSE Intraday Data: Wealth Gained from 20 Stocks 73

5.46 JSE Intraday Data: Wealth Gained from 30 Stocks 74

5.47 JSE Intraday Data: Wealth Gained from 20 Stocks (Active Portfolio with Clusters) . 75

5.48 JSE Intraday Data: Wealth Gained from 20 Stocks (Absolute Portfolio with Clusters) 75

Chapter 1

Introduction

In this dissertation, an algorithm that uses a pattern matching approach for zero-cost portfolio selection for sequential investment strategies is presented. Sequential investment strategies are strategies that use past behaviour to determine how an investor can allocate their wealth amongst a number of assets in a portfolio [6, 10, 45, 25, 8]. Previous strategies of portfolio selection for sequential investment presented by several authors [6, 10, 45, 25, 8] only considered investors holding long only positions on the assets. The algorithm presented in this dissertation and described in Gebbie and Loonat [18] allows the investor to hold a short position on some assets and a long position on the remainder of the assets to arrive at a portfolio that is a zero cost portfolio.

As described in a survey by Li *et al* [33], the Universal Portfolios approach investigated by Cover [10] and the Exponentiated Gradient approach investigated by Helmbold *et al* [25] can be classified as follow the winner approaches. Various agent-based ('experts') pattern matching approaches, each using a different criteria for classifying matches and patterns, have been investigated by Györfi *et al* [20, 21, 22]. Similar to the strategies discussed in the survey by Li *et al* [33], this dissertation aims to demonstrate that the algorithm can beat the best stock in the market, this would infer that the algorithm has successfully found patterns in the time series data. This dissertation also aims to investigate how an analytic approach, derived using the mutual fund separation theorem [31], compares to an optimisation approach in terms of accuracy versus running time.

The literature review in Chapter 2 has an in depth discussion of the Universal Portfolios approach by Cover *et al* [10] and the four algorithms (Histogram, Kernel, Nearest Neighbour and First Neighbour) introduced by Györfi *et al* [20, 21, 22]. These approaches are discussed in depth because the results obtained by the Universal Portfolios approach presented in Cover *et al* [10] and results obtained from the Nearest Neighbour approach presented in Györfi *et al* [21] are compared to the results obtained from the algorithm in this dissertation. Attempts at recovering the results in [10, 21] are achieved by implementing a version of the Universal Portfolios approach and Nearest Neighbour approach, these recovered implementations are used to compare results achieved by the algorithm on datasets not investigated in Cover *et al* [10] and Györfi *et al* [21], namely the extended NYSE dataset (referred to as NYSE Merged) and the JSE datasets. A discussion on thick modelling based on the work of [3] and [46] is also discussed. An overview of the Borodin et al paper titled "Can We Learn to Beat the Best

Stock" is presented to demonstrate an alternate approach to portfolio selection and the interesting extensions that their ANTICOR algorithm implements. A summary of the Li and Hoi [33] survey is also included in Chapter 2 to give an overview of portfolio selection techniques from published literature.

Chapter 3 provides a detailed description of the derivation of the algorithm that is presented as well as pseudocode for the algorithms. The algorithm is inspired by the Györfi *et al* Nearest Neighbour algorithm [21] but has been extended to include zero-cost (active) portfolios and a quadratic approximation instead of an optimisation step to determine the portfolio used. The algorithm is also presented in an online form, however it is important to note that while the algorithm does use online learning [8, 41, 40, 15], the pattern matching is not truly online.

A summary of the datasets used is given in Chapter 4. The data used is in price relative format or is converted into price relative format. A price relative is the ratio of the price of a stock at time t to the price of the stock at time $t-1$. There are four sets of real data, two New York Stock Exchange (NYSE) datasets obtained from [19] and two Johannesburg Stock Exchange (JSE) datasets obtained from [47] and they will be referred to as:

1. NYSE
2. NYSE Merged
3. JSE OHLC
4. JSE Intraday

There are four sets on synthetic data (SDC) each portraying a market with a certain characteristic, the characteristics are as follows:

i. A market without significant growth in the price of the stocks. (see section 4.1.1)

ii. A market where the price increases over time. (see section 4.1.2)

iii. A market where a stock can either increase or decrease over time based on a random variable. (see section 4.1.3)

iv. A market where a few stocks increase over time and a few stocks decrease over time. (see section 4.1.4)

Chapter 5 discusses the results of Kolmogorov-Smirnov tests on the synthetic datasets to ensure that the algorithm behaves as expected for each of the different market characteristics. Chapter 5 also describes the process used to implement the algorithms presented in Chapter 3, and compares the performance of the methods discussed in Chapter 2 to the actual results obtained from the algorithms on the datasets described in Chapter 4.

The MATLAB code that has been used is included in Appendix A of this dissertation.

Chapter 2

Literature Review

2.1 Universal Portfolios

Thomas M. Cover describes the Universal Portfolios algorithm for portfolio selection in [10]. The goal wealth of the Universal Portfolio is the wealth achieved by the best constant rebalanced portfolio. A constant rebalanced portfolio is a portfolio that fixes the fraction of an investor's wealth that is assigned to each stock and rebalances the wealth at these same fractions at the beginning of each trading period. The notation used by Cover in [10] represents the stock market vector by $\mathbf{x} = (x_1, x_2, ..., x_m)$, where x_i is the price relative of the ith stock. $\mathbf{b} = (b_1, b_2, ..., b_m)$ denotes the portfolio of the investor, where b_i is the fraction of the investors wealth assigned to stock i. S_n denotes the wealth achieved by the portfolio at time n and the total number of stocks is m. $\hat{\mathbf{b}}$ and $\hat{S}_n$ represent the universal portfolio and the wealth achieved by the universal portfolio respectively.

The Universal Portfolio strategy is a performance weighted strategy and is initialised by assigning the same fraction of wealth to each of the stocks as follows:

$$\hat{\mathbf{b}}_1 = (\frac{1}{m}, \frac{1}{m}, ..., \frac{1}{m}) \tag{2.1}$$

$$\hat{\mathbf{b}}_{k+1} = \frac{\int \mathbf{b} S_k(\mathbf{b}) d\mathbf{b}}{\int S_k(\mathbf{b}) d\mathbf{b}} \tag{2.2}$$

where $S_k(\mathbf{b}) = \prod_{i=1}^{k} \mathbf{b}^t \mathbf{x_i}$

The integration is over the set $B = \{\mathbf{b} \in \Re^{\mathbf{m}} : b_i \geq 0, \sum_{i=1}^{m} b_i = 1\}$ and the wealth of the universal portfolio is given by $\hat{S}_n = \prod_{k=1}^{n} \hat{\mathbf{b}}_{\mathbf{k}}^{\mathbf{t}} \mathbf{x_k}$.

Cover [10] tests the Universal Portfolios algorithm on real data from the New York Stock Exchange over a 22 year period from 1963 to 1985. The results obtained are for $m = 2$ stocks and the integrals are all quantised. The quantisation is done as follows:

$$\hat{b}_{k+1} = \int_0^1 \frac{bS_k(b)d\mathbf{b}}{\int_0^1 S_k(b)d\mathbf{b}} \text{ is replaced by } \hat{b}_{k+1} = \sum_{i=0}^{20} \frac{\frac{i}{20}S_k(\frac{i}{20})}{\sum_{i=0}^{20} S_k(\frac{i}{20})} \tag{2.3}$$

$$\hat{S}_n = \prod_{k=1}^{n} \hat{\mathbf{b}}_\mathbf{k}^\mathbf{t}\mathbf{x}_\mathbf{k} \text{ is calculated using } \hat{b}_k = \sum_{i=0}^{20} \frac{\frac{i}{20}S_k(\frac{i}{20})}{\sum_{i=0}^{20} S_k(\frac{i}{20})} \tag{2.4}$$

Thomas M. Cover shows that the Universal Portfolio method can perform better than a buy and hold strategy of a single stock.

2.1.1 Recovering Prior Results

Using the same data from the New York Stock Exchange that was used in [10], the results that were obtained were recovered. Figure 2.1 shows that the Universal Portfolio strategy when tested on 2 stocks, Iroquois Brands Ltd. and Kin Ark Corp., outperforms buy and hold strategies for those stocks.

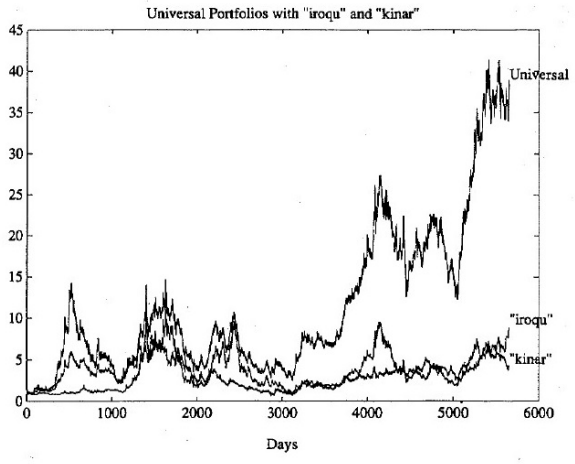
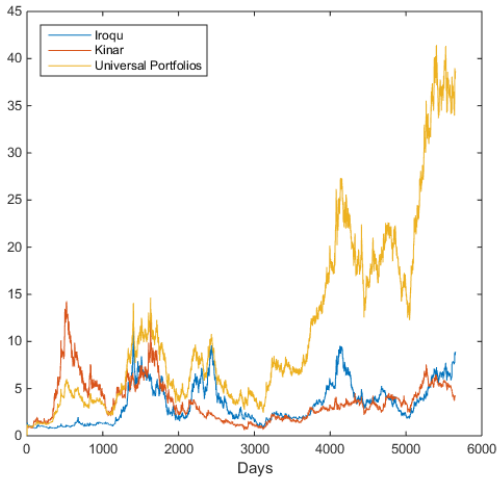


FIGURE 8.3. Performance of universal portfolio.

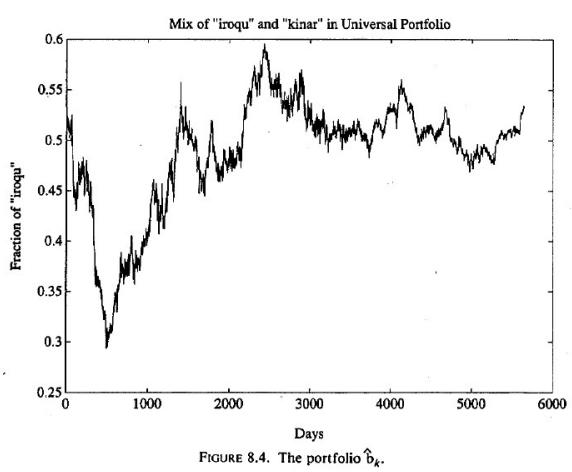
(a) Figure taken from Cover [10].



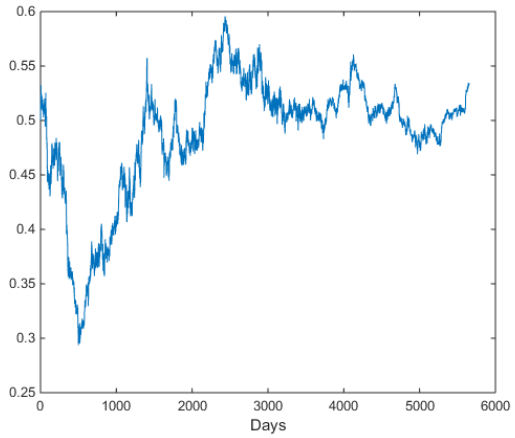
(b) Figure generated in MATLAB using the code from Appendix A.2

Figure 2.1: Wealth gained from the Universal Portfolios strategy.

Figure 2.2 shows the fraction of the investors wealth assigned to Iroquois Brands Ltd. on each trading day using the Universal Portfolio strategy. Using the method outlined in [10], the results were recovered using MATLAB.



(a) Figure taken from Cover [10].



(b) Figure generated in MATLAB using the code from Appendix A.2

Figure 2.2: Fraction of wealth allocated to Iroquois Brands Ltd.

2.2 Györfi *et al* Algorithms

Four algorithms for portfolio selection are introduced by Györfi *et al* in [20], [21] and [22]. They all have a similar structure, where the algorithm looks at the past k days and searches through the entire history to find k days in the history that are similar to the past k days. The algorithms differ in the manner in which matches are defined, how many matches to find and which partitions of the history the algorithm should search. Each algorithm defines an array of 'experts' $\mathbf{H}^{(\mathbf{k},\ell)} = \{\mathbf{h}^{(k,\ell)}(\cdot)\}$ where $k, \ell > 0$ are integers, k is the period length and ℓ is used to determine the matching criteria. A positive probability distribution $q_{k,\ell}$ is used to combine the experts such that the investor's capital becomes:

$$S_n(\mathbf{B}) = \sum_{k,\ell} q_{k,\ell} S_n(H^{(k,\ell)}) \tag{2.5}$$

The following notation is used to describe these algorithms in [20], [21] and [22]:

i. $\mathbf{x}_i$ denotes the return vector at time i. The return vector contains the price relatives for each of the stocks.

ii. $\mathbf{x}_{i-k}^{i-1}$ denotes the return vectors from time $i-k$ to time $i-1$.

iii. $\mathbf{b}$ denotes the portfolio vector, the fraction of wealth assigned to each stock.

2.2.1 Histogram Strategy

The idea of the histogram strategy is to assign integer values to the price relatives in the stock market vector. The partitions used in [20] is determined by using constants a and w which are determined by $\ell \in 1, 2, ..., L$.

$$a = \frac{1}{2(1 + 2\log_{10}\ell)} \quad \text{and} \quad w = a^{\frac{-1}{\ell}}$$

Here x is the price relative of a stock and q is the integer assigned to the corresponding price relative. $M = 2 + 2(\ell)$ represents the total number of price relative partitions.

$$q = \begin{cases} 0 & : x \leq a \\ 1 + \left\lfloor \frac{\log_{10}\left(\frac{x}{a}\right)}{\log_{10}(w)} \right\rfloor & : x \in (a, 1] \\ \ell + 1 + \left\lfloor \frac{\log_{10}(ax) + \ell\log_{10}(w)}{\log_{10}(w)} \right\rfloor & : x \in (1, \frac{1}{a}] \\ 2\ell + 1 & : x > \frac{1}{a} \end{cases}$$

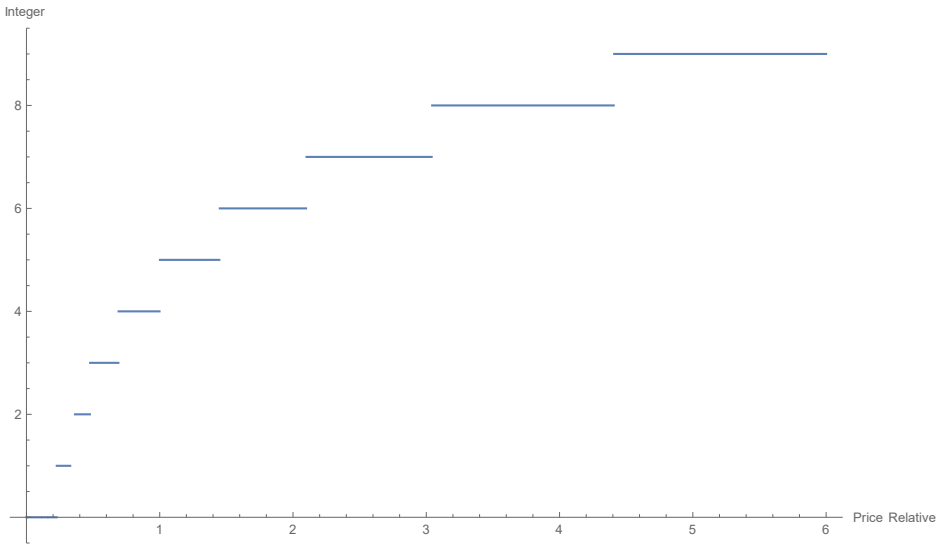


Figure 2.3: A plot of the partitioning function produced in `Mathematica` when $\ell = 4$ and $M = 2 + 2(4) = 10$.

Figure 2.3[1] shows a possible way of partitioning the stock market vectors, the x-axis represents the value of the price relative and the y-axis represents the integer associate with the price relative. It can be observed that there is more integer bands closer to one. A function $G_\ell(\mathbf{x})$ represents the

[1]The `Mathematica` notebook used to produced the figure is available at https://github.com/FayyaazL/Pattern

discretisation of the stock market vector and the set of matches J is defined as follows:

$$J_{k,\ell,n} = \{k < i < n : G_\ell(x_{i-k}^{i-1}) = G_\ell(x_{n-k}^{n-1})\}$$

For each $n > k+1$ the best constant rebalanced portfolio is found, using optimisation, over the stock return vectors at times $i \in J_{k,\ell,n}$. The experts can be formally defined as:

$$\mathbf{h}^{(k,\ell)}(\mathbf{x}_1^{n-1}) = \arg\max_b \prod_{i \in J_{k,\ell,n}} \langle \mathbf{b}, \mathbf{x}_i \rangle$$

2.2.2 Kernel Strategy

The kernel strategy defines the set of matches at time n with the following set:

$$J_{k,\ell,n} = \{k < i < n : \|x_{i-k}^{i-1} - x_{n-k}^{n-1}\| \leq r_{k,\ell}\}$$

The general idea with the kernel method is to find the times i in the history where the norm of the difference between the past k days at time n and the past k days at times i is less than a radius $r_{k,\ell}$ which is defined in terms of k and ℓ. A disadvantage of this method is that if $r_{k,\ell}$ is too small then very few or no matches could be found and if $r_{k,\ell}$ is too large then too many matches will be found. The experts are formally defined as follows and solved using optimisation:

$$\mathbf{h}^{(k,\ell)}(\mathbf{x}_1^{n-1}) = \arg\max_b \prod_{i \in J_{k,\ell,n}} \langle \mathbf{b}, \mathbf{x}_i \rangle$$

2.2.3 Nearest Neighbour Strategy

The nearest neighbour algorithm that is introduced in [21] and [22] is similar to the kernel strategy, however instead of using a radius the nearest neighbour strategy uses an integer $\hat{\ell}$ to determine how many nearest neighbour matches to use. [21] and [22] formally define the strategy as follows:

$$\hat{\ell} = \lfloor p_\ell n \rfloor$$

Where $p_\ell \in (0, 1)$ and n is time.
The p_ℓ used in the results section of [21] is defined as:

$$p_\ell = 0.02 + 0.5\frac{\ell - 1}{L - 1}$$

where $\ell = 1, ..., L$.
The set of matches is defined as:

$$J_{k,\ell,n} = \{i : k+1 < i < n \text{ such that } x_{i-k}^{i-1} \text{ is among the } \hat{\ell} \text{ nearest neighbours } \}$$

The experts are defined as follows and is solved using optimisation:

$$\mathbf{h}^{(k,\ell)}(\mathbf{x}_1^{n-1}) = \arg\max_b \prod_{i \in J_{k,\ell,n}} \langle \mathbf{b}, \mathbf{x}_i \rangle$$

2.2.4 First Neighbour Strategy

The first neighbour strategy is a variant of the nearest neighbour strategy and is implemented because it is computationally faster than the nearest neighbour strategy. The idea is to split the history up into segments and to search for the best match in each of the segments. This is done by using a term s_ℓ to denote the segment length and is defined in the paper as $s_\ell = 2 + 50\frac{\ell-1}{L-1}$. The strategy can be formally written as follows:

$$N_i = \underset{(i-1)s_\ell + k \leq j \leq i s_\ell}{\arg\min} \|x_{j-k}^{j-1} - x_{n-k}^{n-1}\|$$

Here N_i is the instant of the nearest neighbour match in the ith segment and the experts are defined as follows and solved using optimisation:

$$\mathbf{h}^{(k,\ell)}(\mathbf{x}_1^{n-1}) = \arg\max_b \prod_{1 \leq i \leq \left\lfloor \frac{n-1}{s_\ell} \right\rfloor} \langle \mathbf{b}, \mathbf{x}_i \rangle$$

2.2.5 Recovering Prior Results

In table 2.1, a comparison of the results obtained from the nearest neighbour strategy by Györfi *et al* in [22] and the results obtained from implementing a recovered version of the Nearest Neighbour algorithm is shown. The experiments were carried out for the experts using $k = 1, ..., 5$ and $\ell = 1, ..., 10$. It can be observed that values obtained in the recovery attempt has the same order of magnitude as the values obtained in [22]. Only the Nearest Neighbour Strategy has been recovered as this showed to be the best performer [21] and is the approach that the algorithm in this dissertation extends by using a cash neutral portfolio (active portfolio) and a quadratic approximation instead of an optimisation step.

Stocks	**Györfi *et al***		**Recovered**	
Iroquois Kin Ark	Best Agent Total Wealth:	1.439e+13 1.156e+12	Best Agent Total Wealth:	1.629e+13 8.9729e+11
Com Met. Mei. Corp	Best Agent Total Wealth:	3.148e+4 3.505e+3	Best Agent Total Wealth:	2.601e+04 3.8402e+3
Com. Met. Kin Ark	Best Agent Total Wealth:	8.257e+13 4.781e+12	Best Agent Total Wealth:	3.748e+13 2.7121e+12
IBM Coca-Cola	Best Agent Total Wealth:	296.3 74.37	Best Agent Total Wealth:	267.440 80.9688

Table 2.1: Recovering the prior results of the Györfi *et al* [21] Nearest Neighbour algorithm using the MATLAB code in Appendix A.1

The difference between the results from Györfi *et al* [22] and the recovered results may be attributed to two possible reasons, the optimisation method used and the initialisation of the agents. The optimisation method used in the literature as stated in [22] is Spellucci's `DONLP2` [51] method which is a variant of the sequential quadratic programming method, while the method used to recover the results is MATLAB's `fmincon` method which uses a gradient descent method [44]. In Györfi *et al* [21], it is suggested that the agents are initialised at time $k + 1$ and in the implementation of the algorithm in this dissertation the algorithm is initialised at time 3ℓ.

Although the results from the literature and the results obtained from the recovered implementation of the Nearest Neighbour are not exact, it does have the same order of magnitude as observed in table 2.1. This allows for an accurate comparison of the methods on datasets not experimented on in [21].

2.3 Thick Modelling

Pesaran and Timmerman [46] investigate the predictability of stock returns by applying a recursive modelling approach to predict all possible forecasting models and the stock returns are predicted based on the best model. Aiolfi and Favero [3] describe this approach as thin modelling because the approach uses only the best model to forecast the returns.

Aiolfi and Favero [3] extend the approach of predicting stock returns of Pesaran and Timmerman [46] to one that is based on thick modelling. Thick modelling is achieved by considering more multiples of the predicted possible forecasting models [2, 3].

An interesting result in [3] is that the results of thin modelling approach of Pesaran and Timmerman [46] are weakened in the decade 1990-2000.

It can be observed that the approach used in this dissertation is one of thick modelling that does not

rely on the best result but rather multiple results to determine portfolio selection.

2.4 "Can We Learn to Beat the Best Stock"

Borodin, Yaniv and Gogan [8] present the ANTICOR method in their paper *Can we learn to beat the best stock*. An alternate approach to portfolio selection is taken where the ANTICOR algorithm uses statistical relations between stocks to determine the portfolio to be used in the next trading period.

The approach of the ANTICOR algorithm is to define two consecutive trading windows formulated as $[t-2w+1, t-w]$ and $[t-w+1, t]$, where w is the number of trading days in the window. Borodin *et al* [8] uses the logarithm of the price relatives in these trading windows and calculates the cross correlation matrix of the two trading windows [33]. Borodin *et al* [8] shows that the performance of the ANTICOR algorithm relies on the choice of the trading window w. In order to learn the best w, Borodin *et al* [8] defines $\texttt{ANTI}^1$ which uses experts (agents) with window lengths ranging from $2 \leq w \leq 30$ that each has a uniform buy and hold investment.

An interesting approach used by Borodin *et al* [8] is to extend the approach from purely trading on the stocks to trading on the algorithms. This approach is denoted as $\texttt{ANTI}^2$ and trades on the algorithms from $\texttt{ANTI}^1$. It is shown that the results achieved from this compounding significantly improves the performance of the portfolio.

Another interesting result of [8] is that the algorithm is not only tested on the NYSE data set [19] but is further tested on the Toronto Stock Exchange (TSE), 25 stocks from the SP500 and 30 stocks composing the Dow Jones Industrial Average (DJIA). The four markets are then reversed and experimented on to ensure that the algorithm performs favourably on markets that are different in nature.

2.5 "Online Portfolio Selection: A Survey"

Li and Hoi [33] provide a comprehensive survey on portfolio selection that discusses 25 techniques from published literature. These techniques are classified into the following 5 categories:

1. Benchmarks
2. Follow the Winner
3. Follow the Loser
4. Pattern-Matching-Based Approaches
5. Meta-Learning Algorithms

The techniques classified as benchmarks are the buy-and-hold strategy, the best stock strategy and the constant rebalanced portfolios strategy. These strategies are merely used, as the classification

suggests, as a benchmark for the other strategies. Cover [10] states that his Universal Portfolios strategy aims to beat the best stock and sets the best constant rebalanced portfolio as a target wealth for the strategy. A buy-and-hold strategy is where stocks are bought in the first period and no rebalancing is done.

Techniques that rebalance the portfolio based on the assets that are performing well are classified as Follow the Winner approaches. The Universal Portfolios strategy of Cover [10] can be classified as such an approach and is discussed in detail in Section 2.1. Another strategy classified as a Follow the Winner approach is the Exponential Gradient strategy proposed by Helmbold *et al* [25] which tracks the stock with the best performance but does not drastically change the updated portfolio in comparison to the previous portfolio. The approach proposed by Gaivoronski and Stella [17] that tracks the best constant rebalanced portfolio, the Online Newton Step approach proposed by Agarwal *et al* [1] and the aggregating approach proposed by Vovk and Watkins [53] that uses a finite set of investment strategies that an investor would consider to be reasonable have also been classified as strategies that are Follow the Winner approaches.

Follow the Loser approaches, based on the principle of mean reversion, uses the idea that if a stock performs poorly in the current period it will perform well in the following periods and that if a stock is performing well in the current period it will perform poorly in the following periods. The ANTICOR strategy presented by Borodin *et al* [8] and discussed in section 2.4 has been classified as a Follow the Loser approach. Other strategies classified as Follow the Loser approaches are the Passive Aggressive Mean Reversion approach in [36] and the Confidence Weighted Mean Reversion approach in [35] proposed by Li *et al* which is named based on the online learning chosen. The Online Moving Average Reversion proposed by Li and Hoi [32] is also classified as a Follow the Loser approach.

The Györfi *et al* [20, 21, 22] approaches, outlined in Section 2.2, along with the correlation driven nonparametric learning approach proposed by Li *et al* [34] have been classified as pattern matching based approaches and are described to have two steps. The first step is the sample selection and the second step is the portfolio optimisation.

Approaches classified as Meta-Learning Algorithms are approaches similar to Follow the Winner approaches but has a broader class of experts. Vovk and Watkins [53] Aggregating Algorithms is classified as both a Follow the Winner approach and a Meta-Learning Algorithm. Fast Universalisation proposed by Akcoglu *et al* [4, 5], an extension of Universal Portfolios [10], is also classified as a Meta-Learning Algorithm because it uses a wide class of investment strategies apart from the best constant rebalanced portfolios. Online Gradient and Newton Updates proposed by Das and Banerjee [14], which is an extension of the Exponential Gradient [25] and Online Newton Step [1], and the Follow the Leading history approach proposed by Hazan and Seshadhri [24], which has a working set of experts dynamically flowed in and dropped out, are also classified as Meta-Learning Algorithms.

Chapter 3

Algorithm

3.1 The Algorithm and Mathematical Model

The model that is used in this dissertation has a similar structure the model used by Györfi *et al* [20, 21, 22] and Cover *et al* [10]. The price relative of each stock is denoted by:

$$x_{m,t} = \frac{p_{m,t}}{p_{m,t-1}} \tag{3.1}$$

Where $x_{m,t}$ is the price relative of stock m at time t and $p_{m,t}$ is the price of stock m at time t. Equivalently written in vector form as $\mathbf{x_t}$.

The algorithm is formally defined as 3 algorithms and is based on the Györfi *et al* [21] Nearest Neighbour algorithm, it is however extended to include zero cost (cash neutral) portfolios and uses an analytic quadratic approximation instead of an optimisation step to determine the portfolio selection. Algorithm 1 is named the Online-Learning Algorithm (OLA), Algorithm 2 is named the Pattern Matching Algorithm (PMA) and Algorithm 3 is named the Matching Algorithm (MTA).

The Online-Learning Algorithm serves the purpose of managing the portfolio. It takes in as input the portfolio controls $b_{m,t}$, the agent controls $H_{n,t}$, the updated agent controls $H_{n,t+1}$, the feature realisation x_t, the past agent wealth $S_{n,t-1}$ and the past portfolio wealth S_{t-1}.

$b_{m,t}$ denotes the fraction of the portfolio's wealth allocated to stock m at time t and is referred to as the portfolio controls. S_t denotes the portfolio wealth and is updated as follows:

$$\Delta S_t = \left[\sum_m b_{m,t}(x_{m,t} - 1)\right] + 1 \tag{3.2}$$

$$S_t = S_{t-1}\Delta S_t. \tag{3.3}$$

$H_{m,t}$ denotes the fraction of an agents wealth allocated to stock m at time t and is referred to as the

agents controls. $S_{n,t}$ denotes the agent wealth and is updated as follows:

$$\Delta S_{n,t} = \left[\sum_m H_{nm,t}(x_{m,t} - 1)\right] + 1. \tag{3.4}$$

$$S_{n,t} = S_{n,t-1}\Delta S_{n,t}. \tag{3.5}$$

The portfolio controls $b_{m,t}$ is updated using the agent controls as follows:

$$b_{m,t+1} = \sum_n q_{n,t+1} H_{nm,t+1}. \tag{3.6}$$

Where $q_{n,t+1}$ represents the agent mixer control for agent n at time $t+1$. The agent mixer control is determined by an agent mixer update rule which will be referred to as rule g. The following mixture update rules are considered:

(a) **Universal Portfolio**
The Universal Portfolios update mixture rule, inspired by Cover *et al* [10], is calculated as follows:

$$q_{n,t+1} = S_{n,t} \tag{3.7}$$

The agent mixtures at time t+1 is proportional to the wealth achieved by the agents at time t.

(b) **Exponential Gradient**

$$q_{n,t+1} = q_{n,t} e^{\left(\frac{\eta S_{n,t}}{\sum_n q_{n,t} S_{n,t}}\right)} \tag{3.8}$$

(c) **Exponential Weighted Moving Average**

$$q_{n,t+1} = \lambda q_{n,t} + (1-\lambda)\left(\frac{q_{n,t} S_{n,t}}{\sum_n q_{n,t} S_{n,t}}\right) \tag{3.9}$$

In the experiments the Universal Portfolios update mixture rule is used. The agent mixtures will need to be re-normalised. Similar to having absolute portfolios and active portfolios, the agents can also be absolute or active. In the case of absolute agents it is required that $\sum_n q_n = 1$ and all $q_n \geq 0$, the normalisation is as follows:

$$q_{n,t+1} = \frac{q_{n,t+1}}{\sum_n q_{n,t+1}}. \tag{3.10}$$

In the case of active agents it is required that $\sum_n q_n = 0$ and that $\sum_n |q_n| = 1$, the normalisation is as follows:

$$q_{n,t+1} = \frac{q_{n,t+1} - \frac{1}{N}\sum_n q_{n,t+1}}{\sum_n |q_{n,t+1} - \frac{1}{N}\sum_n q_{n,t+1}|} \tag{3.11}$$

The Online-Learning Algorithm then returns the updated portfolio controls $b_{m,t}$, the past agent wealth $S_{n,t-1}$ and the past portfolio wealth S_{t-1}.

Algorithm 1 Online-Learning Algorithm (OLA)

Require:

1. updated agent-controls $\boldsymbol{H}_{n,t+1}$
2. current feature realisation $\boldsymbol{x}_t$
3. current portfolio controls $\boldsymbol{b}_t$
4. current agent-controls $\boldsymbol{H}_{n,t}$
5. past agent-wealth $S_{n,t-1}$
6. past portfolio wealth S_{t-1}

for t-state **do**

Step 1: The portfolio wealth is updated

$S_t = S_{t-1}(\boldsymbol{b}_t(\boldsymbol{x}_t^T - 1) + 1)$

Step 2: The agent wealth is updated

$S_{n,t} = S_{n,t-1}(\boldsymbol{H}_{n,t}(\boldsymbol{x}_t^T - 1) + 1)$

Step 3: The agent mixture is updated for rule g

$q_{n,t+1} = g(q_{n,t}, S_{n,t})$

Step 4: The agent mixtures are re-normalised

$$q_{n,t+1} = \begin{cases} \sum_n q_{n,t+1} = 1,\ q_{n,t+1} \geq 0 \\ \sum_n |q_{n,t+1}| = 1, \sum_n q_{n,t+1} = 0. \end{cases}$$

Step 5: The portfolio controls are updated

$\boldsymbol{b}_{t+1} = \sum_n q_{n,t+1} \boldsymbol{H}_{n,t+1}$

Leverage corrections

if $(\nu = \sum_m |b_{m,t}|) \neq 1$ **then**

renormalise controls

$\boldsymbol{b}_{n,t+1} = \frac{1}{\nu} \boldsymbol{b}_{n,t+1}$

renormalise mixtures

$q_{n,t+1} = \frac{1}{\nu} q_{n,t+1}$

end if

end for

return ($\boldsymbol{b}_{t+1}$,$S_{n,t}$,S_t,$q_{n,t+1}$)

The agent controls are determined using algorithm 2 and algorithm 3. The Pattern Matching Algorithm is used to iterate over the agents and returns the agent controls of each of the agents by calling on the Matching Algorithm.

There are a total number of N agents and each agent is represented by parameters k, ℓ, w and τ. The parameter k is used to determine the window length of the agent, the parameter ℓ is used to determine the amount of matching times of the agent, the parameter w is used to represent the stock cluster and the parameter τ is used to represent the horizon parameter. There is a total of K values of k, L values of ℓ, W values of w and τ_n values of τ. The total number of agents is $N = \tau_n WKL$, however in this dissertation only the default choice of $\tau_n = 1$ is used in the experiments.

Algorithm 2 PATTERN-MATCHING Algorithm (PMA)

Require:

1. features $\boldsymbol{x}_t$
2. n-agent parameters k, ℓ, $s(n)$, τ
3. partitions $\{p_\ell\}$

for n-agents **do**

$\boldsymbol{H}_{n,t+1}$=MATCHING(τ,$\{p_\ell\}$,$\{\boldsymbol{x}_{t-k}^t\}_{s(n)}$,$\{\boldsymbol{x}_t\}_{(p_\ell,s(n))}$)

end for agents

return $\boldsymbol{H}_{t+1}$

The Matching Algorithm contains the actual process of finding matches in the history of the price relative data and calculating the covariance, Σ, and mean, μ, of the matches (see Section 3.3) to update the controls using equation 3.23.

3.2 Portfolio Selection

A derivation of a quadratic approximation for determining the portfolio controls, as outlined in [18], will now be presented.

Algoet and Cover [6] shows that if a market is stationary ergodic, a log-optimal strategy is the best choice of strategy over the long term. The log optimal portfolio selection problem is given by:

$$\boldsymbol{b}^*(X_1^{t-1}) = \arg\max_{\boldsymbol{b}} \mathrm{E}\left[\log(\boldsymbol{b}(X_1^{t-1})\boldsymbol{X}_t)|\boldsymbol{X}_1^{t-1}\right]. \tag{3.12}$$

Where $\boldsymbol{X}_t$ denotes the random expectation return vector given some past $\boldsymbol{X}_1^{t-1}$ and $\boldsymbol{b}^*$ denotes the log optimal portfolio.

Györfi *et al* [23] shows the semi-log-optimal strategies and log-optimal strategies have the same performance when tested on NYSE data. The semi-log-optimal portfolio takes on the following form:

$$\boldsymbol{b}^*(X_1^{t-1}) = \arg\max_{\boldsymbol{b}} \mathrm{E}\left[h(\boldsymbol{b}(X_1^{t-1})\boldsymbol{X}_t)|\boldsymbol{X}_1^{t-1}\right]. \tag{3.13}$$

We can consider the semi-log-optimal portfolio optimisation problem [45, 31]

$$\max_{\boldsymbol{\omega}}\{\boldsymbol{\omega}^{\boldsymbol{T}}\boldsymbol{\mu} - \frac{\gamma}{2}\boldsymbol{\omega}^{\boldsymbol{T}}\Sigma\boldsymbol{\omega}\} \text{ s.t. } \boldsymbol{\omega}^{\boldsymbol{T}}\mathbf{1} = 1 \tag{3.14}$$

the following notation changes have been made:

1. The return expectation vector is denoted by μ.
2. Σ denotes the asset return covariance matrix.

3. The portfolio control vector is denoted by ω.

4. γ denotes the risk aversion parameter.

Equation (3.14) can be rewritten as the mutual-fund Lagrangian

$$L = \boldsymbol{\omega^T \mu} - \frac{\gamma}{2}\boldsymbol{\omega^T}\Sigma\boldsymbol{\omega} - \lambda_\omega(\boldsymbol{\omega^T 1} - 1) \tag{3.15}$$

and can be solved by using Kuhn-Tucker Methods. Equation (3.15) leads to the following two equations:

$$\boldsymbol{\omega^*} = \frac{1}{\gamma}\Sigma^{-1}\left(\boldsymbol{\mu} - \lambda_\omega \mathbf{1}\right) \tag{3.16}$$

$$\boldsymbol{\omega^{*T}}\mathbf{1} = 1. \tag{3.17}$$

We can use (3.16) and (3.17) to find the Lagrange multiplier:

$$\lambda_\omega = \frac{\mathbf{1}^T\Sigma^{-1}\boldsymbol{\mu}}{\mathbf{1}^T\Sigma^{-1}\mathbf{1}} - \frac{\gamma}{\mathbf{1}\Sigma^{-1}\mathbf{1}}. \tag{3.18}$$

We can substitute (3.18) into (3.16) to find a formulation of the mutual fund separation theorem:

$$\boldsymbol{\omega}^* = \frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}^T\Sigma^{-1}\mathbf{1}} + \frac{1}{\gamma}\Sigma^{-1}\left(\boldsymbol{\mu} - \mathbf{1}\frac{\mathbf{1}^T\Sigma^{-1}\boldsymbol{\mu}}{\mathbf{1}^T\Sigma^{-1}\mathbf{1}}\right). \tag{3.19}$$

Equation (3.19) is a form of the active fund separation theorem and can be rewritten as :

$$\boldsymbol{\omega}^* = \boldsymbol{\omega}_B + \boldsymbol{\omega}_A \tag{3.20}$$

where

$$\boldsymbol{\omega}_B = \frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}^T\Sigma^{-1}\mathbf{1}} \tag{3.21}$$

$$\boldsymbol{\omega}_A = \frac{\Sigma^{-1}}{\gamma}\left(\frac{\boldsymbol{\mu}\mathbf{1}^T - \mathbf{1}\boldsymbol{\mu}^T}{\mathbf{1}_T\Sigma^{-1}\mathbf{1}}\right)\Sigma^{-1}\mathbf{1} \tag{3.22}$$

The equations for ω_A and $\boldsymbol{\omega}_B$ is used directly in the agent generating algorithms. μ denotes the mean of the portfolio return vector and Σ denotes the covariance matrix. The agent controls can now be calculated for either absolute or active agents as follows:

$$\boldsymbol{H}_{n,t} = \begin{cases} \boldsymbol{h}^T\mathbf{1} = 1, \boldsymbol{h} = \boldsymbol{\omega}_B(\Sigma) + \boldsymbol{\omega}_A(\gamma, \boldsymbol{\mu}, \Sigma) \text{ s.t. } \boldsymbol{h} \geq 0 \\ \boldsymbol{h}^T\mathbf{1} = 0, \boldsymbol{h} = \boldsymbol{\omega}_A(\gamma, \boldsymbol{\mu}, \Sigma) \text{ s.t. } \sum|\boldsymbol{h}| = 1 \Leftrightarrow |\boldsymbol{h}|^T\mathbf{1} = 1 \end{cases} \tag{3.23}$$

3.3 Pattern Matching

The pattern matching part of the algorithm involves each agent searching through the entire history for matching times. A time t in the history is a match depending on the k and ℓ terms of the agent. k is the window length of the agent that determines how many time periods in the window. ℓ is used to determine how many matches each agent should use, this is denoted by $\hat{\ell}$. The following choices of $\hat{\ell}$ have been investigated in this dissertation:

1. *Trivial*
$$\hat{\ell} = \ell \tag{3.24}$$

2. *Györfi et al Nearest Neighbour* [21] based where $\hat{\ell}$ is determined by a variable $p_\ell \in (0, 1)$. The choice of p_ℓ used in this dissertation is the same choice used in [21].
$$p_\ell = 0.02 + 0.5\frac{\ell - 1}{L - 1} \tag{3.25}$$
$$\hat{\ell} = \lfloor p_\ell t \rfloor \tag{3.26}$$

The agent thus uses the closest $\hat{\ell}$ matching times. In the event of a tie, the agent will use the older matching time only for the trivial case and the agent will use all matching times of the ties for the *Györfi et al Nearest Neighbour* [21] based case.

The matching criteria used in this dissertation is the Euclidean norm or Euclidean distance between two time periods formally defined as follows:

$$\|\mathbf{x}_{t-k}^{t-1} - \mathbf{x}_{T-k}^{T-1}\| \tag{3.27}$$

Where T represents the time that the agents are going to invest into, t represents times from the past and $\mathbf{x}_{t-k}^{t-1}$ represents a sequence of price relative return vectors for k consecutive time periods.

The set of matching times can be formally defined as:

$$J = \{t | t \text{ is amongst the} \hat{\ell} \text{ minimum values of} \|\mathbf{x}_{t-k}^{t-1} - \mathbf{x}_{T-k}^{T-1}\|\} \tag{3.28}$$

$\hat{\mathbf{x}}$ can be defined as a sequence of the price relative vectors at the matching times in set J. The mean and covariance of $\hat{\mathbf{x}}$ is used in equation 3.23 to calculate the agent controls.

3.4 Partitioning

The algorithm in this dissertation takes into account the possibility of partitioning the history. As discussed in section 3.3, each agent is required to search for a total of $\hat{\ell}$ matching times. One option,

the trivial option, is to have each agent search the entire history for the best $\hat{\ell}$ matching times. An alternate approach could be to partition the history into $\hat{\ell}$ partitions and have each agent search for the best match in each partition to get a total of $\hat{\ell}$ matching times. The choice of partition can be chosen with the intention of decreasing the computational similar to Györfi's first neighbour algorithm [21]. The partitioning may also be chosen with the intention of increasing the wealth of the portfolio, this could be based on side information or giving preference to matches that are more recent. The following are possible methods of partitioning the data:

1. Trivial Partitioning: Using the entire history

$$\{p_t\}_1 = \{(1, \ldots, 1, 1, 1)\}. \tag{3.29}$$

2. Over-lapping Partitioning

$$\{p_t\}_T = \left\{ \begin{array}{c} (0, \ldots, 0, 0, 1) \\ (0, \ldots, 0, 1, 1) \\ (0, \ldots, 1, 1, 1) \\ \vdots \\ (1, \ldots, 1, 1, 1) \end{array} \right\}. \tag{3.30}$$

3. Exclusive Partitioning

$$\{p_t\}_T = \left\{ \begin{array}{c} (0, \ldots, 0, 0, 1) \\ (0, \ldots, 0, 1, 0) \\ (0, \ldots, 1, 0, 0) \\ \vdots \\ (1, \ldots, 0, 0, 0) \end{array} \right\}. \tag{3.31}$$

4. Side-Information Partitioning

$$\{p_t\}_T = \left\{ \begin{array}{c} (0, 0, 0, \ldots 0, 0, 1, 1) \\ (0, 0, 0, \ldots 1, 1, 0, 0) \\ \vdots \\ (0, 1, 0, \ldots 0, 0, 0, 0) \\ (1, 1, 1, \ldots 0, 0, 0, 0) \end{array} \right\}. \tag{3.32}$$

   The partitions used for side-information partitioning can be based on observations made on the data or could be chosen to favour certain days of the week.

While various possible methods of partitioning the data is discussed in this dissertation, only the trivial method of partitioning the data is used in the experiments.

3.5 Clustering

As discussed in section 3.4, partitioning refers to when the data that an agent searches is dependent on time. Clustering is when the data that an agent searches is based on the selection of stocks.

Each cluster has an index of w and there are a total of W clusters. The choice of clusters used in the experiments is on classifying the stocks.

Algorithm 3 MATCHING Algorithm (MTA)

Require:

1. look-ahead-rule τ
2. partition $\{p_\ell\}$
3. k-tuple $\{\boldsymbol{x}_{t-k}^t\}_{s(n)}$
4. data partition $\{\boldsymbol{x}_t\}_{(p_\ell, s(n))}$

for t-state **do**
  for $p_\ell \in \{p_\ell\}$ **do**
    for j-states $\in p_\ell$ **do**
      find a test-tuple
      $\boldsymbol{s}_{t(j)-k}^{t(j)} = \{\boldsymbol{x}_{t(j)-k}^{t(j)}\}_{(p_\ell, s(n))}$
      find distance to k-tuple
      $\boldsymbol{\epsilon}_{k,j} = \{\boldsymbol{x}_{t-k}^{k}\}_{s(n)} - \boldsymbol{s}_{t(j)-k}^{t(j)}$
      if k=1 **then**
        compute the 2-norm for vector $\boldsymbol{\epsilon}_{1,j}$
        $\epsilon_j = \sum_{m \in \text{objects}} \sqrt{\epsilon_{m1,j}2} = \sqrt{\boldsymbol{\epsilon}_{1,j}\boldsymbol{\epsilon}_{1,j}^T}$
        distance measure of dim(objects)
        $\{\boldsymbol{\epsilon}_j\}_{p_\ell} \leftarrow \epsilon_{m,j} = \epsilon_j \forall m$
      else
        column-wise 2-norms for matrix $\boldsymbol{\epsilon}_{j,k}$
        $\{\boldsymbol{\epsilon}_j\}_{p_\ell} \leftarrow \epsilon_{m,j} = \sum_{k'=1}^{k} \sqrt{\epsilon_{mk',j}^2}$
      end if
    end for states
    if $\dim(\{p_\ell\}) = 1$ **then**
      Switch NN algorithm partition choice [21]
      $\hat{\ell} = P(\ell, t)$
      Find $\hat{\ell}$ matching-times in a single partition
      $\boldsymbol{j}_\ell = \min_{j \forall \dim(j) = \ell} \{\boldsymbol{\epsilon}_j\}$
    else
      Find the best match in each of the ℓ partitions
      $\boldsymbol{j}_\ell = \min_{j \forall p_\ell \in \{p_\ell\}} \{\boldsymbol{\epsilon}_j\}_{p_\ell}$
    end if
  end for partitions
  update the look-ahead-rule
  $j_n = j_\ell + \tau$
  Update the agent-tuple
  $\boldsymbol{x}_{n,t} = \{\boldsymbol{x}_1^t\}_{t \in j_n}$
  Update the mean and covariance
  $\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{x}_{n,t} - 1)$
  $\Sigma = \Sigma(\boldsymbol{x}_{n,t} - 1)$
  Update the agent-controls
  $\boldsymbol{H}_{n,t+1} = \boldsymbol{H}_{n,t+1}(\gamma, \boldsymbol{\mu}, \Sigma)$
end for t-state
return $\boldsymbol{H}_{n,t+1}$

Chapter 4

Data

4.1 Synthetic Data

The algorithm was tested on four synthetic data cases (SDC) for both active and absolute portfolios. The synthetic data was generated for 10 stocks over 1000 time periods. The price relatives $x_{m,t}$ for each stock at each time period was randomly generated from a lognormal distribution (`lognrnd` function in MATLAB generated using the Mersenne Twister pseudorandom number generator [38] and initialised using a specific seed value), each synthetic data case defines a mean, μ, and variance, v, used to generate the dataset. The mean, $\bar{\mu}$, and standard deviation, $\bar{\sigma}$, of the associated normal distribution is given by :

$$\bar{\mu} = \log\left(\frac{\mu^2}{\sqrt{v+\mu^2}}\right) \tag{4.1}$$

$$\bar{\sigma} = \sqrt{\log\left(\frac{v}{\mu^2}+1\right)} \tag{4.2}$$

Table 4.1 summarises the four synthetic data cases, each case was generated 30 times and initialised with seed values $1, 2, \ldots, 30$ respectively.

4.1.1 Synthetic Data Case 1 (SDC 1)

Synthetic Data Case 1 (SDC 1) was generated from a lognormal distribution with a mean price relative, $\mu = 1$, and a variance, $v = 0.0002$, to simulate a stock market where there is no significant increase or decrease in the value of a stock over time.

The expected outcome is that neither the active portfolio nor the absolute portfolio will be able to learn which stocks it should hold a long position or short position.

4.1.2 Synthetic Data Case 2 (SDC 2)

Synthetic Data Case 2 (SDC 2) was generated from a lognormal distribution with a mean price relative, $\mu = 1.001$, and a variance, $v = 0.0002$, to simulate a stock market where the value of the stocks are increasing over time.

The expected outcome is that the absolute portfolio will learn which stocks to hold a long position on, however the active portfolio will not be able to learn which stocks to hold a short position on as no stocks decrease in value over time.

4.1.3 Synthetic Data Case 3 (SDC 3)

Synthetic Data Case 3 (SDC 3) was generated from a lognormal distribution with a random mean price relative, $\mu \geq 1$, assigned to each stock and a variance, $v = 0.0002$. The random means is calculated as follows:

$$\mu_m = 1 + \max[0, \min(0.0005 + 0.0005\delta, 0.001)] \tag{4.3}$$

where δ is a random number generated from a standard normal distribution (using the `randn` function in MATLAB with the Mersenne Twister pseudorandom number generator [38] and initialised using a specific seed value). This simulates a stock market where some stocks are increasing in value and some stocks are decreasing in value over time.

The expected outcome is that both the active portfolio and the absolute portfolio will learn to hold a long position on the stocks increasing in value over time and hold a short position on the stocks decreasing in value over time, however it is expected that the absolute portfolio will beat the active portfolio due to the growth rate of the stocks increasing in value over time.

4.1.4 Synthetic Data Case 4 (SDC 4)

Synthetic Data Case 4 (SDC 4) was generated from a lognormal distribution with mixed means assigned to the price relatives, $\mu = 0.999$ was assigned to 3 stocks and $\mu = 1.001$ was assigned to the remaining stocks, and a variance, $v = 0.0002$. This dataset simulates a stock market where the value of some stocks are increasing and the value of some stocks are decreasing.

The expected outcome is that both the active portfolio and the absolute portfolio will learn to hold a long position on the stocks increasing in value over time and hold a short position on the stocks

decreasing in value over time.

Summary of Random Datasets

Dataset	μ	v
SDC 1	1	0.0002
SDC 2	1.001	0.0002
SDC 3	random≥ 1	0.0002
SDC 4	mixed	0.0002

Table 4.1: The means and variances that were chosen when generating the synthetic data sets. The random means for SDC 3 was calculated using Eqn. (4.3) and the means for SDC 4 was generated as described in section 4.1.4.

4.2 Real Data

The algorithm is tested on four sets of real data, summarised in Table 4.3, two data sets from the New York Stock Exchange (NYSE) obtained at [19] and two data sets from the Johannesburg Stock Exchange (JSE) obtained at [47].

4.2.1 NYSE Data

The NYSE dataset, as described in [19], comes from Yoram Singer and contains *close-to-close* price relatives for 36 stocks listed on the New York Stock Exchange from 1962-1984. This is the same data set used by Györfi *et al* in [20, 21] and Cover in [10].

4.2.2 NYSE Merged Data

The NYSE Merged dataset, as described in [19], was obtained from Yahoo! Finance and was cleaned and prepared by Gábor Gelencsér. This dataset contains *close-to-close* price relatives data for 23 stocks listed on the New York Stock Exchange from 1962-2006. The data of the 23 stocks during 1962-1984 is identical to the data described in section 4.2.1.

Date	Price Relative
1964-12-09	0.9979
1964-12-10	1.0046
1964-12-11	1.0104
1964-12-14	0.9946
1964-12-15	0.9991
1964-12-16	1.0009
1964-12-17	1.0049
1964-12-18	1.0048
1964-12-21	1.0000

Figure 4.1: A sample of the NYSE data for stock IBM. The data is in comma separated value (.csv) files, the first value is the date and the second value is the price relative of the stock.

4.2.3 JSE OHLC Data

The JSE OHLC dataset was obtained from Thomson Reuters Tick History (TRTH) [47] and contains daily data for 42 stocks listed on the Johannesburg Stock Exchange (JSE) from 1995-2015, however not all of the 42 stocks were listed in 1995 and the data for these stocks begins at a later time. The data lists the open, high, low and close prices for all of the 42 stocks, as shown in figure 4.2.

#RIC	Date[L]	Time[L]	Type	Qualifiers	Open	High	Low	Last	Volume	Open Inte	Settle	ISIN Code	Data Source
ANGJ.J	05-Oct-98		End Of Day		35020	35100	33400	34400	139282				Verified
ANGJ.J	06-Oct-98		End Of Day		34200	34200	31900	32520	122863				Verified
ANGJ.J	07-Oct-98		End Of Day		32700	34400	32700	33640	170720				Verified
ANGJ.J	08-Oct-98		End Of Day		33500	35500	33500	35200	223243				Verified
ANGJ.J	09-Oct-98		End Of Day		35000	35000	34000	34000	158311				Verified
ANGJ.J	10-Oct-98		End Of Da	No Trades									Verified
ANGJ.J	11-Oct-98		End Of Da	No Trades									Verified
ANGJ.J	12-Oct-98		End Of Day		32500	32500	31400	31800	107704				Verified
ANGJ.J	13-Oct-98		End Of Day		31000	31000	30000	30100	76331				Verified
ANGJ.J	14-Oct-98		End Of Day		30000	30000	29200	29300	117620				Verified
ANGJ.J	15-Oct-98		End Of Day		29000	29500	28600	29500	121299				Verified
ANGJ.J	16-Oct-98		End Of Day		30300	30560	29320	29700	220011				Verified
ANGJ.J	17-Oct-98		End Of Da	No Trades									Verified
ANGJ.J	18-Oct-98		End Of Da	No Trades									Verified
ANGJ.J	19-Oct-98		End Of Day		29640	29640	29020	29200	51805				Verified
ANGJ.J	20-Oct-98		End Of Day		29000	29300	28700	29300	149855				Verified

Figure 4.2: A sample of the original JSE OHLC data.(Screenshot from Microsoft Excel.

The raw data was processed into four datasets:

1. *close-to-close:*
   The price relative for the m-th asset is calculated using the closing price and calculated as follows:
   $$x_{m,t} = \frac{c_{m,t}}{c_{m,t-1}}. \tag{4.4}$$

where $x_{m,t}$ represents the price relative and $c_{m,t}$ represents the closing price at time t.

2. *open-to-close:*
   The price relative for the m-th asset is calculated using the open price and closing price of the asset from the same trading period and is calculated as follows:

$$x_{m,t} = \frac{c_{m,t}}{o_{m,t}}. \tag{4.5}$$

   where $x_{m,t}$ represents the price relative, $c_{m,t}$ represents the closing price at time t and $o_{m,t}$ represents the opening price at time t.

3. *close-to-open:*
   The price relative for the m-th asset is calculated using the opening price of the current trading period(t) and the closing price from the previous trading period ($t-1$) and is calculated as follows:

$$x_{m,t} = \frac{o_{m,t}}{c_{m,t-1}}. \tag{4.6}$$

   where $x_{m,t}$ represents the price relative, $c_{m,t}$ represents the closing price at time t and $o_{m,t}$ represents the opening price at time t.

4. *open-to-open:*
   The price relative for the m-th asset is calculated using the opening price and calculated as follows:

$$x_{m,t} = \frac{o_{m,t}}{o_{m,t-1}}. \tag{4.7}$$

   where $x_{m,t}$ represents the price relative and $o_{m,t}$ represents the opening price at time t.

Splits, mergers and missing data[1] were handled by assigning a price relative of 1 for that day.

Date	Price Relative
1998-09-01	1.01235
1998-09-02	1.03659
1998-09-03	1.01059
1998-09-04	1.05471
1998-09-07	1.04305
1998-09-08	1.01587
1998-09-09	0.95833
1998-09-10	1.01196
1998-09-11	0.99356

Figure 4.3: A sample of the processed JSE OHLC data to *close-to-close* price relatives for Anglo American PLC (AGLJ.J). The data is stored in Microsoft Excel (.xls) files, the first value is the date and the second value is the price relative of the stock.

[1]Splits and mergers were identified as having a $x_{m,t} < 0.7$ and $x_{m,t} > 1.3$ respectively.

4.2.4 JSE Intraday Data

The JSE Intraday dataset was obtained from Thomson Reuters Tick History (TRTH) [47], in a comma-separated values (CSV) file, and consists of high frequency data for 40 stocks listed on the Johannesburg Stock Exchange (JSE) during 2013. The file contains the following fields:

Field	**Description**
RIC	Reuters Instrument Price
Date[L]	Date of order book event
Time[L]	time (in microseconds) of order book event
Type	Type of LOB event (*Trade*, *Auction* or *Quote*)
Price	The volume of the trade or auction
Volume	The price at which the trade or auction was executed
Market VWAP	Traded volume weighted average price (VWAP) of the share
Bid Price	level 1 bid price
Bid Size	level 1 bid size
Ask Price	level 1 ask price
Ask Size	level 1 ask volume

Table 4.2: Description of the fields available in the CSV file.

The CSV file was split into weekly data per RIC by the QuERI Lab team[2] and saved as MAT-Files. The high frequency data was converted into 5 minute bar data using the trade price. The 5 minute barred data starts at 9h30 and ends at 16h30 for normal trading days and starts at 9h30 and ends at 11h30 for early close days. The 5 minute bar data was then converted into price relative data for each stock and saved as MAT-Files.

Summary of Real Datasets

Data Set	**Time Period**	**Stocks**
NYSE [19]	1962-1984	36
NYSE Merged [19]	1962-2005	23
JSE OHLC [47]	1995-2015	42
JSE Intraday [47]	2013	40

Table 4.3: Description of the real data sets that the algorithm was tested on.

[2]Quantifying Emergence, Risk and Information. The data was prepared by Dieter Hendricks and Michael Harvey.

Chapter 5

Results

5.1 Implementation

5.1.1 Parameters Chosen

In all experiments the following parameters were used:

i. $k = (1, 2, \ldots, 5)$ which implies that $K = 5$

ii. $\ell = (1, 2, \ldots, 10)$ which implies that $L = 10$

iii. $w = 1$ for all experiments with the exception of the clustered results where $w = 3$. This implies that $W = 1$ for experiments with the trivial choice of clusters and $W = 3$ for the experiments that include clusters.

iv. $\tau_n = 1$

v. The pattern matching used is formulated in equations 3.25 and 3.26

This implies that the number of agents, $N = \tau_n WKL$, is 50 when the trivial choice of clusters ($W = 1$) are used and 150 for the experiments that include clusters ($W = 3$).

5.1.2 Comparison to Results from the Literature

As a measure of performance of the algorithm, the results are compared to Györfi *et al*'s nearest neighbour method [21, 22] and Thomas Cover's Universal Portfolio method [10]. It can be observed that if the algorithm described in this dissertation were to replace the analytic solution with an optimisation step for the absolute case, the algorithm would be analogous to Györfi's Nearest Neighbour

method [21, 22]. The pattern class, as described in section 5.1.4, therefore implements a recovered version of the Nearest Neighbour algorithm in this manner, using `fmincon` for the optimisation, as to allow for accurate comparisons of the computational running times. A MATLAB function implementing Thomas Cover's Universal Portfolio method [10] was created and recovered the exact results obtained in [10].

Results obtained from the algorithm presented in this dissertation that compare favourably to the results obtained from the methods presented in the literature and benchmarks such as the best stock would indicate that the algorithm has successfully exploited patterns in the time series data.

5.1.3 Implementation Process

Figure 5.1 outlines the process followed to conduct the experiments. Step 2 of the process checks for gaps in the data and inaccuracies caused by splits and mergers and replaces these values with a price relative of one.

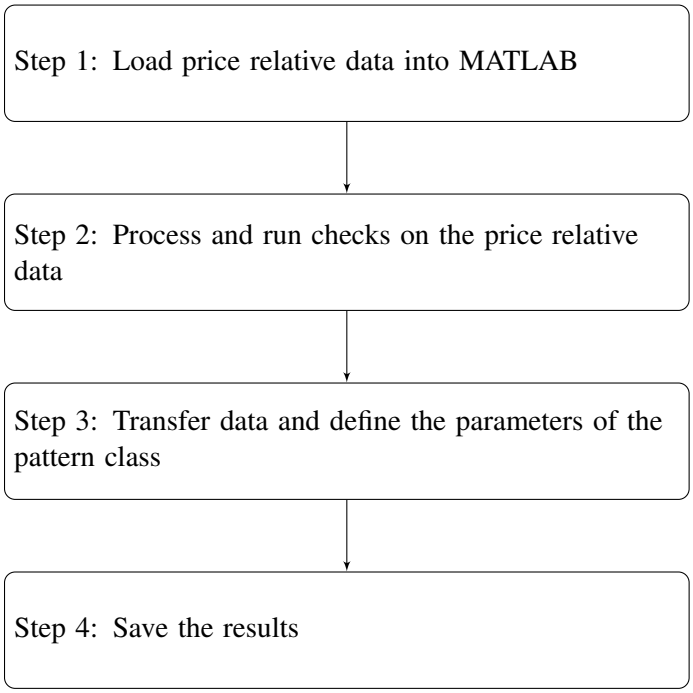


Figure 5.1: A process flow diagram illustrating the steps that have been followed in generating the results.

5.1.4 Pattern Class

A MATLAB class named *pattern* (see Appendix A.1) was created to implement the algorithms 1, 2 and 3. The class contains the following properties:

Property	Description
x	The price relative data
k	A list of the k values
ℓ	A list of the ℓ values
c	A tuple describing the clusters
NTYPE	The portfolio type (*active*, *absolute* or *gyorfi_opt*)
b	The portfolio controls
S	The portfolio wealth
h	The agent controls
SH	The agent wealth

Table 5.1: A summary of the properties contained in the pattern class.

The class contains the following methods:

Function	Description
`pattern`	The constructor of the pattern class
`offline`	Offline estimation of the wealth S and portfolio controls SH
`online`	Online estimation of the wealth S and portfolio controls SH
`learn`	Updates the agent controls h, the portfolio controls b and the wealth S
`partition`	Partitions the data
`quadbet`	Solves the quadratic approximation (Equation 3.23)
`match`	Implements the pattern matching

Table 5.2: A summary of the Methods contained in the pattern class.

5.2 Synthetic Data

Experiments were run for both the absolute and active portfolios on all four synthetic data sets. Each data set depicts a different behavioural characteristic of a market, as described in section 4.1. Table 5.3 shows the best wealth achieved from all of the 30 runs per dataset as well as the average wealth achieved from all of the 30 runs per dataset.

5.2.1 Wealth Achieved from Synthetic Datasets

Wealth Gained (S) from Investing in Synthetic Data

Data	**Port.**	Wealth		Best Agent	
		Best	Avg.	Best	Avg.
SDC 1	Abs.	1.2315	0.9917	1.8063	1.2505
	Act.	1.4507	1.0516	1.7532	1.3579
SDC 2	Abs.	3.2410	2.6119	4.6545	3.2696
	Act.	1.4508	1.0516	1.7532	1.3583
SDC 3	Abs.	2.3202	1.6850	3.0908	2.0899
	Act.	1.4901	1.1709	1.7816	1.4508
SDC 4	Abs.	2.4554	1.8964	2.9307	2.2501
	Act.	2.9275	2.0554	3.2698	2.2975

Table 5.3: Wealth achieved by the active and absolute portfolios for 30 runs of each synthetic data case.

5.2.2 Kolmogorov-Smirnov Tests

Two-sample Kolmogorov-Smirnov tests were run on the datasets because it is a non-parametric test and makes no assumption about the distribution of the datasets. The following notation is used:

1. S_1 refers to the total wealth gained from the portfolio.
2. S_2 refers to the wealth gained from the best agent of the portfolio.
3. S_3 refers to the wealth gained from the best stock.

The following combinations of two-sample Kolomogorov-Smirnov were performed:

1. $S_2 > S_1$: The alternative hypothesis that the cumulative distribution function (CDF) of the wealth gained from the best agent of the portfolio, S_2, is larger than the CDF of the total wealth gained from the portfolio, S_1, at the 5% significance level.
2. $S_2 > S_3$: The alternative hypothesis that the CDF of the wealth gained from the best agent of the portfolio, S_2, is larger than the CDF of the wealth gained from the best stock, S_3, at the 5% significance level.
3. $S_3 > S_1$: The alternative hypothesis that the CDF of the wealth gained from the best stock, S_3, is larger than the CDF of the total wealth gained from the portfolio, S_1, at the 5% significance level.

Comparison of Average p Values of Wealth Gained (S) from the Active Portfolio

	Best Agent vs. Total Wealth		**Best Agent vs. Best Stock**		**Best Stock vs. Total Wealth**	
Hyp.	$S_2 > S_1$		$S_2 > S_3$		$S_3 > S_1$	
	$\bar{p}$	$p > \bar{p}$	$\bar{p}$	$p > \bar{p}$	$\bar{p}$	$p > \bar{p}$
SDC 1	0.8085	0.1719	0.0310	0.0000	0.6540	0.0129
SDC 2	0.8087	0.1719	0.0000	0.0000	0.8731	0.4072
SDC 3	0.8304	0.1719	0.0000	0.0000	0.9044	0.5627
SDC 4	0.7254	0.0129	0.0000	0.0000	0.6218	0.0057

Table 5.4: Comparisons of the average p values of the wealth gained from the active portfolio. The first p value in each column is average p value, of the 30 data sets for each case, using two-sample Kolmogorov-Smirnov tests for the alternative hypotheses (Hyp.). The second p value is obtained from the two-sample Kolmogorov-Smirnov tests for the alternative hypothesis that the cumulative distribution function (CDF) of the p values for the 30 data sets for each case is larger than the CDF of the average p value at the 5% significance level.

Comparison of Average p Values of Wealth Gained (S) from the Absolute Portfolio

	Best Agent vs. Total Wealth		**Best Agent vs. Best Stock**		**Best Stock vs. Total Wealth**	
Hyp.	$S_2 > S_1$		$S_2 > S_3$		$S_3 > S_1$	
	$\bar{p}$	$p > \bar{p}$	$\bar{p}$	$p > \bar{p}$	$\bar{p}$	$p > \bar{p}$
SDC 1	0.8931	0.4072	0.0002	0.0000	0.6469	0.0129
SDC 2	0.6424	0.0129	0.0000	0.0000	0.5672	0.0009
SDC 3	0.5933	0.0023	0.0004	0.0000	0.7949	0.1003
SDC 4	0.8457	0.2742	0.0000	0.0000	0.6441	0.0057

Table 5.5: Comparisons of the average p values of the wealth gained from the absolute portfolio. The first p value in each column is average p value, of the 30 data sets for each case, using two-sample Kolmogorov-Smirnov tests for the alternative hypotheses (Hyp.). The second p value is obtained from the two-sample Kolmogorov-Smirnov tests for the alternative hypothesis that the cumulative distribution function (CDF) of the p values for the 30 data sets for each case is larger than the CDF of the average p value at the 5% significance level.

Two-sample Kolmogorov-Smirnov tests were also used to compare the wealth achieved from one of the synthetic data cases against the wealth achieved from the other three synthetic data cases per seed value as shown in tables 5.6 and 5.7. As expected, the results show that the active portfolio performs the best on SDC 4 as this dataset includes stocks that are decreasing in value over time and the absolute portfolio performs the best on SDC 2 as this dataset contains the most stocks that are increasing in value over time.

Comparison of p Values of Wealth Gained (S) from the Active Portfolio

	SDC 1	**SDC 2**	**SDC 3**	**SDC 4**
SDC 1	-	0.9765	0.0669	0
SDC 2	0.9786	-	0.0669	0
SDC 3	0.7928	0.7916	-	0
SDC 4	0.9653	0.9649	0.9535	-

Table 5.6: Comparison of the average p values from two-sample Kolmogorov-Smirnov tests for the alternative hypothesis that the cumulative distribution function (CDF) of wealth gained from the active portfolio on SDC i is larger than the CDF of wealth gained from the active portfolio on SDC j at the 5% significance level, where i represents the rows and j represents the columns of the table. The p values is the average of 30 comparisons, each comparison using a seed value of $1, 2, \ldots, 30$ respectively.

Comparison of p Values of Wealth Gained (S) from the Absolute Portfolio

	SDC 1	**SDC 2**	**SDC 3**	**SDC 4**
SDC 1	-	0	0	0
SDC 2	1.0000	-	1.0000	0.9997
SDC 3	1.0000	0	-	0.1289
SDC 4	1.0000	0	0.5055	-

Table 5.7: Comparison of the average p values from two-sample Kolmogorov-Smirnov tests for the alternative hypothesis that the cumulative distribution function (CDF) of wealth gained from the absolute portfolio on SDC i is larger than the CDF of wealth gained from the active portfolio on SDC j at the 5% significance level, where i represents the rows and j represents the columns of the table. The p values is the average of 30 comparisons, each comparison using a seed value of $1, 2, \ldots, 30$ respectively.

5.3 NYSE Data

Experiments were run on the same two stock combinations used by Thomas Cover [10] and Györfi *et al* [20, 21, 22] for both the absolute and active portfolios. The results of the wealth achieved from the active portfolio and the absolute portfolio are compared to the wealth achieved from the implemented versions of the Györfi nearest neighbour algorithm [21] and the Universal Portfolio method [10]. As this is the same dataset used in [21] and [10], the results are compared to the actual results published in [21] and [10]. However, since the results from the Universal Portfolio method [10] was identically recovered it is only shown once in the results tables.

The experiments were carried out on the following stock combinations:

i. Iroquois Brands Ltd. and Kin Ark Corp.

ii. Commercial Metals and Mei Corp.

iii. Commercial Metals and Kin Ark Corp.

iv. IBM and Coca-Cola

v. All 36 NYSE stocks

5.3.1 Iroquois Brands Ltd. and Kin Ark Corp.

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	1.02e+12	1.76e+13	12574.51
Active Portfolio	1.00e+11	4.97e+11	12450.93
Györfi Nearest Neighbour (R)	1.01e+12	1.63e+13	15624.21
Györfi Nearest Neighbour	1.16e+12	1.44e+13	
Universal Portfolio	38.67		
Best Stock	8.92		

Table 5.8: Comparison of wealth gained from Iroquois Brands Ltd. and Kin Ark Corp. for the different strategies.

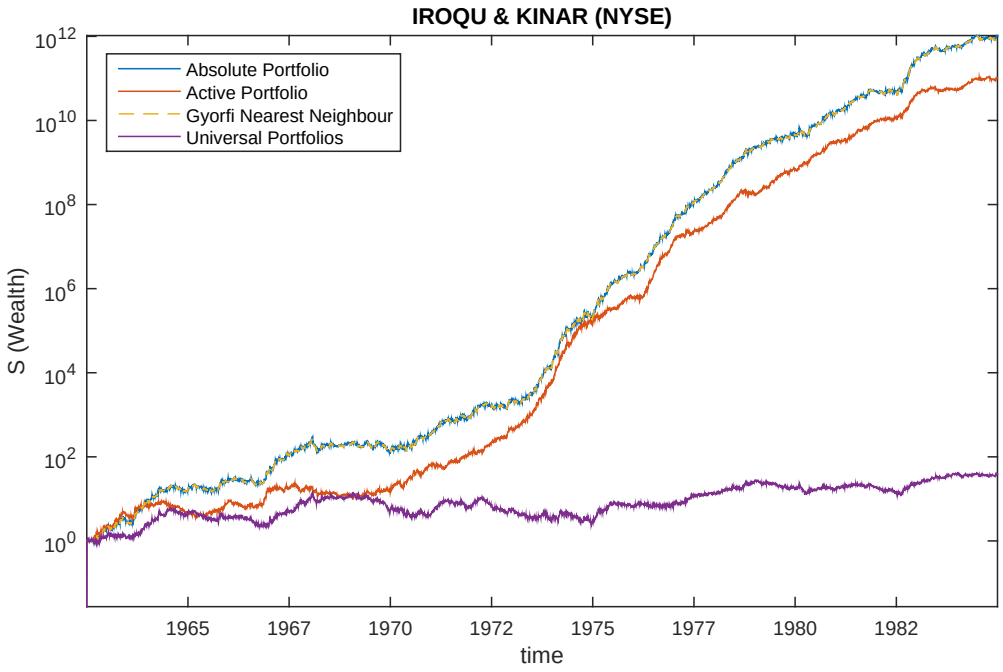


Figure 5.2: Wealth gained when investing in Iroquois Brands Ltd. and Kin Ark Corp. from the different strategies.

5.3.2 Commercial Metals and Mei Corp.

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	3.56e+03	2.61e+04	12503.29
Active Portfolio	4.28e+01	2.49e+02	12617.22
Györfi Nearest Neighbour (R)	3.58e+03	2.60e+04	17051.50
Györfi Nearest Neighbour	3.51e+3	3.15e+4	
Universal Portfolio	72.63		
Best Stock	52.02		

Table 5.9: Comparison of wealth gained from Commercial Metals and Kin Ark Corp. for the different strategies.

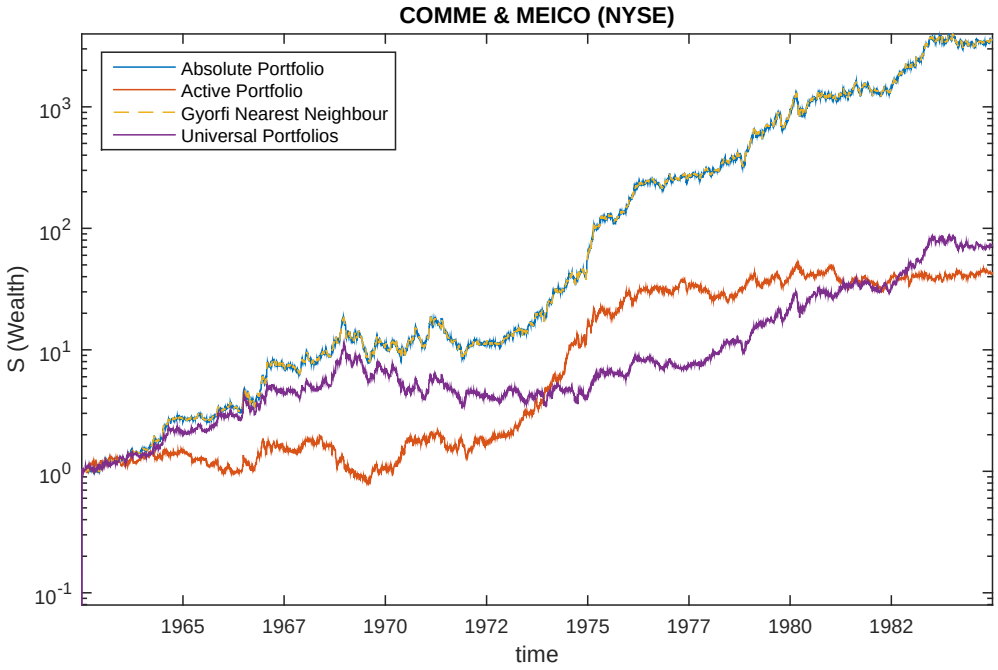


Figure 5.3: Wealth gained when investing in Commercial Metals and Mei Corp. from the different strategies.

5.3.3 Commercial Metals and Kin Ark Corp.

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	2.99e+12	3.46e+13	12711.53
Active Portfolio	1.05e+11	3.75e+11	12653.46
Györfi Nearest Neighbour (R)	3.09e+12	3.75e+13	15112.28
Györfi Nearest Neighbour	4.78e+12	8.26e+13	
Universal Portfolio	78.47		
Best Stock	52.02		

Table 5.10: Comparison of wealth gained from Commercial Metals and Kin Ark Corp. for the different strategies.

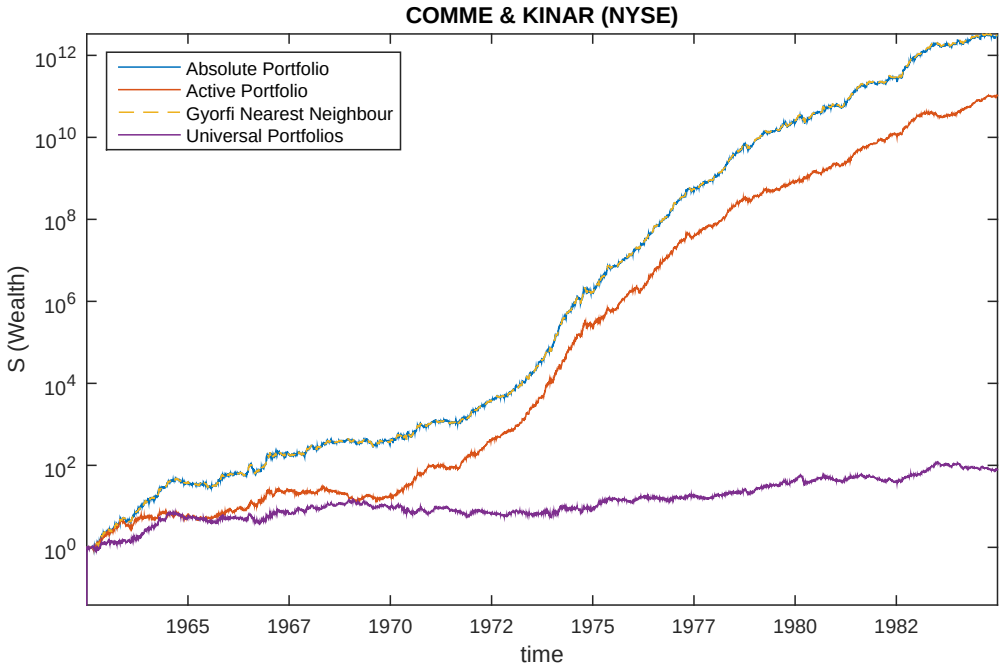


Figure 5.4: Wealth gained when investing in Commercial Metals and Kin Ark Corp. from the different strategies.

5.3.4 IBM and Coca-Cola

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	7.84e+01	2.74e+02	12759.65
Active Portfolio	9.70e+00	1.98e+01	12714.48
Györfi Nearest Neighbour (R)	7.83e+01	2.67e+02	17460.24
Györfi Nearest Neighbour	74.37	296.3	
Universal Portfolio	14.18		
Best Stock	13.36		

Table 5.11: Comparison of wealth gained from IBM and Coca-Cola for the different strategies.

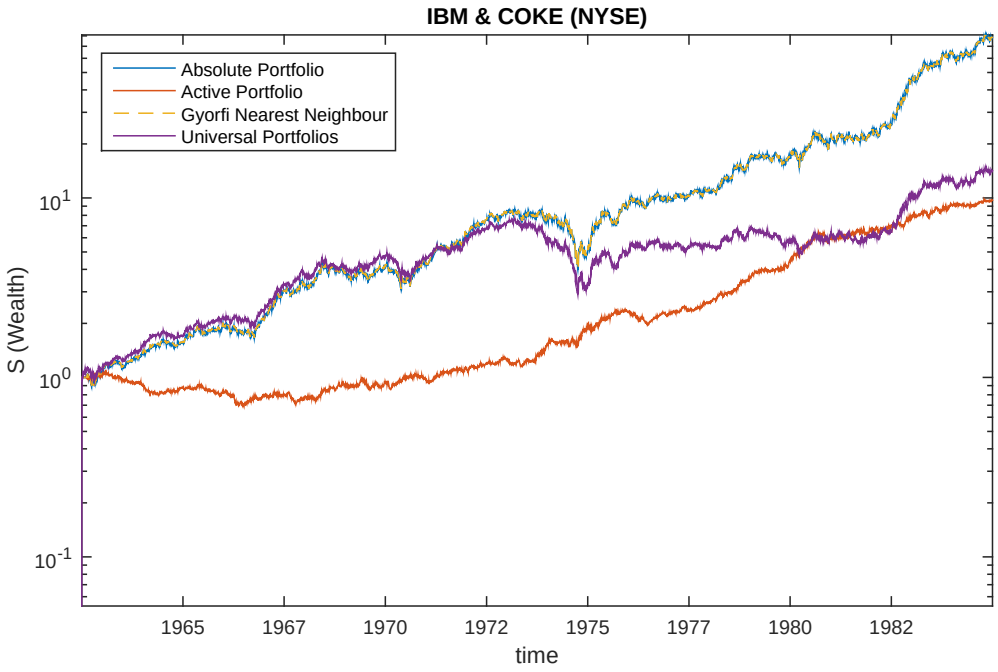


Figure 5.5: Wealth gained when investing in IBM and Coca-Cola from the different strategies.

5.3.5 All 36 Stocks

The 36 stocks that are included in this data set are:

1. 3M Company
2. Alcoa
3. Altria Group
4. Arco
5. Coca Cola
6. Commercial Metals
7. Dow Chemical Company
8. DuPont
9. Eastman Kodak
10. Espey
11. Exxon Mobil
12. Fischback

13. Ford Motor Company
14. Fortune Brands
15. General Electric
16. General Motors
17. Gran Tierra Energy Inc.
18. Gulf Oill
19. Hewlett-Packard
20. IBM
21. Ingersoll-Rand Plc
22. Iroquois Ltd.
23. Johnson & Johnson
24. Kimberly-Clark Corp
25. Lukens
26. Mei Corp.
27. Merck & Company
28. Mobil
29. Kin Ark Corp.
30. Pillsbury
31. Procter & Gamble
32. Schlumberger N.V.
33. Sears
34. Sherwin-Williams
35. Texaco
36. Wyeth

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	5.42e+01	1.36e+02	20231.17
Active Portfolio	5.29e+01	7.13e+01	19783.48
Györfi Nearest Neighbour (R)	3.43e+11	7.45e+12	42604.55
Györfi Nearest Neighbour	3.3e+11	7.7e+12	
Best Stock	54.14		

Table 5.12: Comparison of wealth gained from all 36 stocks for the different strategies.

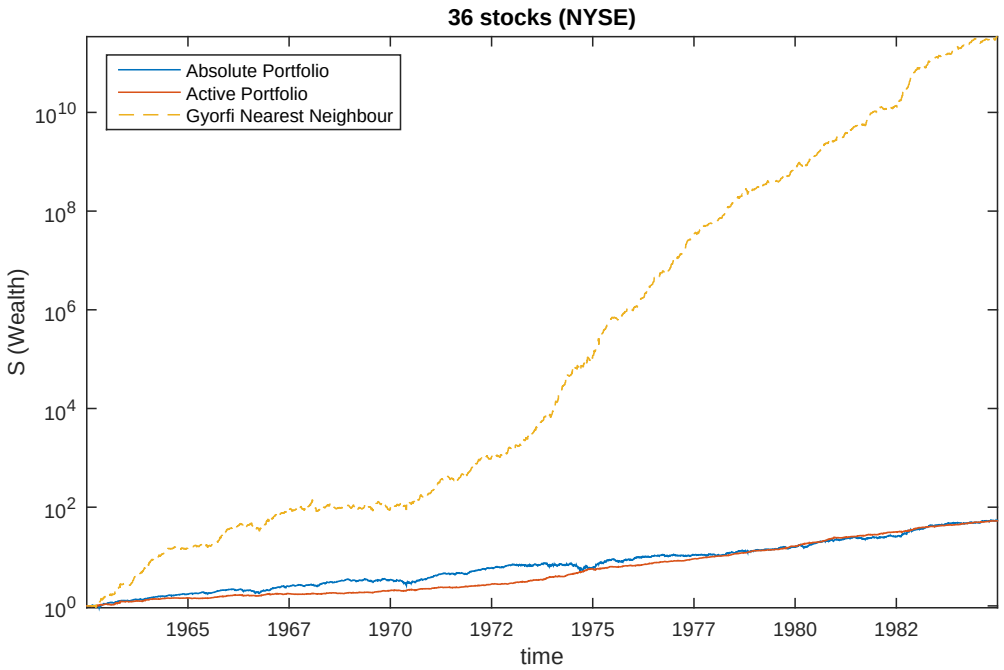


Figure 5.6: Wealth gained when investing in all 36 stocks from the NYSE Dataset using the different strategies.

5.4 NYSE Merged Data

Experiments were run for both the absolute and active portfolios and compared with the results of the wealth achieved from the implemented versions of the Györfi nearest neighbour algorithm [21] and the Universal Portfolio method [10].

The experiments were carried out on the following stock combinations:

i. Commercial Metals and Kin Ark Corp.

ii. IBM and Coca-Cola

iii. All 23 NYSE stocks

5.4.1 Commercial Metals and Kin Ark Corp.

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	3.07e+19	4.37e+20	52490.41
Active Portfolio	7.36e+16	7.93e+16	52393.40
Györfi Nearest Neighbour (R)	3.19e+19	4.73e+20	53655.29
Universal Portfolio	2192.43		
Best Stock	1344.3		

Table 5.13: Comparison of wealth gained from Commercial Metals and Kin Ark Corp. for the different strategies.

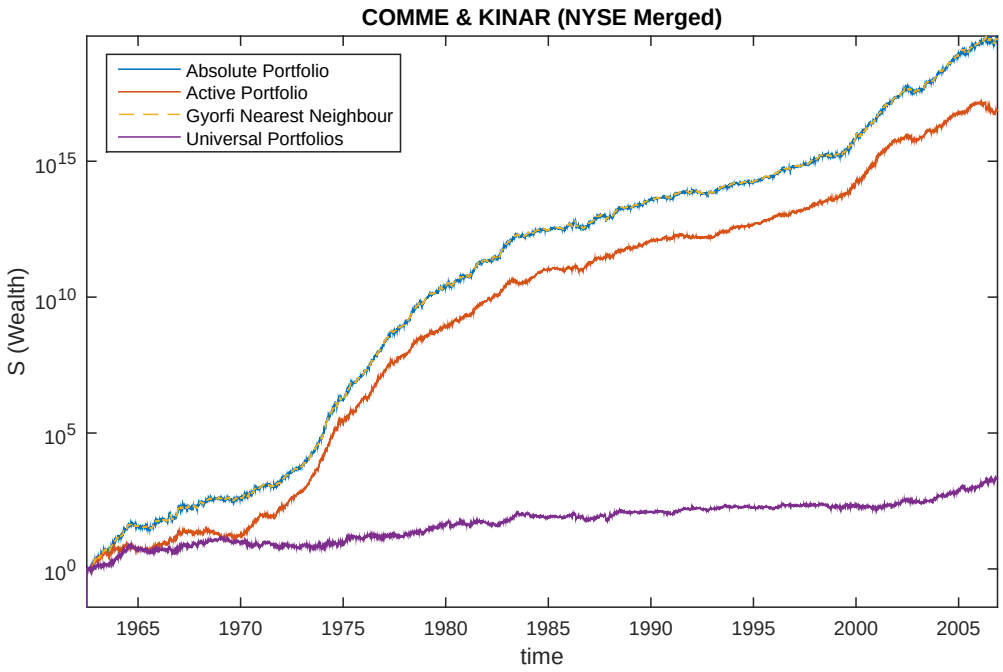


Figure 5.7: Wealth gained when investing in Commercial Metals and Kin Ark Corp. from the different strategies.

5.4.2 IBM and Coca-Cola

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	1.79e+03	5.12e+03	52785.56
Active Portfolio	7.79e+00	2.63e+01	52034.46
Györfi Nearest Neighbour (R)	1.79e+03	4.75e+03	57482.15
Universal Portfolio	229.13		
Best Stock	365.92		

Table 5.14: Comparison of wealth gained from IBM and Coca-Cola for the different strategies.

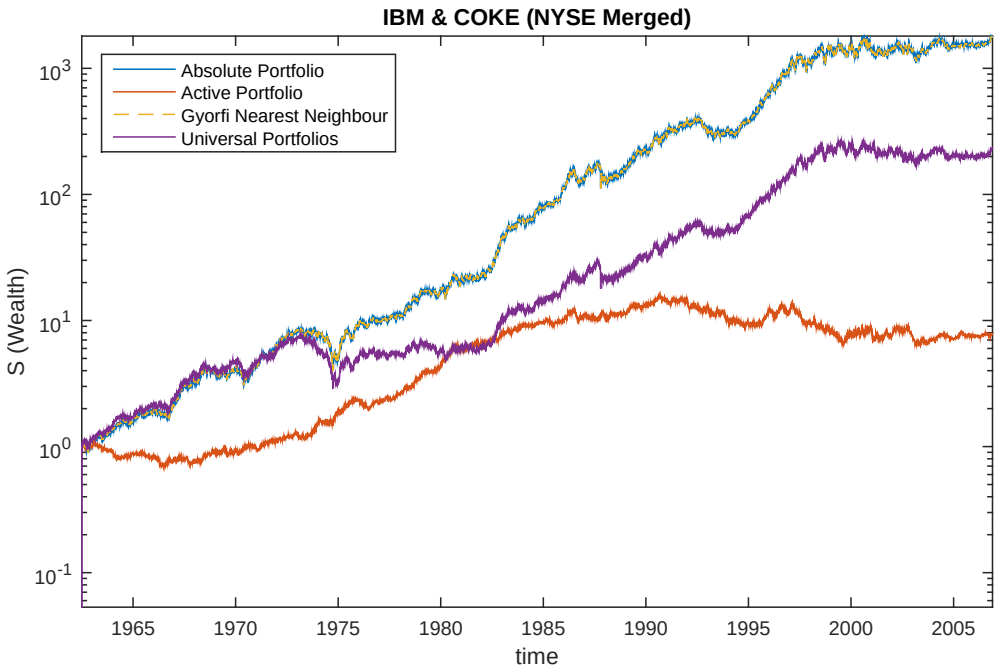


Figure 5.8: Wealth gained when investing in IBM and Coca-Cola from the different strategies.

5.4.3 All 23 Stocks

The 23 stocks included in this dataset are:

1. 3M Company
2. Alcoa
3. Altria Group
4. Coca Cola
5. Commercial Metals
6. Dow Chemical Company
7. DuPont
8. Eastman Kodak
9. Ford Motor Company
10. Fortune Brands
11. General Electric
12. General Motors
13. Hewlett-Packard
14. IBM
15. Ingersoll-Rand Plc
16. Johnson & Johnson
17. Kimberly-Clark Corp
18. Merck & Company
19. Kin Ark Corp.
20. Procter & Gamble
21. Schlumberger N.V.
22. Sherwin-Williams
23. Wyeth

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	8.05e+04	3.34e+05	82137.89
Active Portfolio	1.45e+06	5.42e+05	72899.44
Györfi Nearest Neighbour	3.68e+17	5.60e+18	109240.30
Best Stock	3496.7		

Table 5.15: Comparison of wealth gained from all 23 stocks for the different strategies.

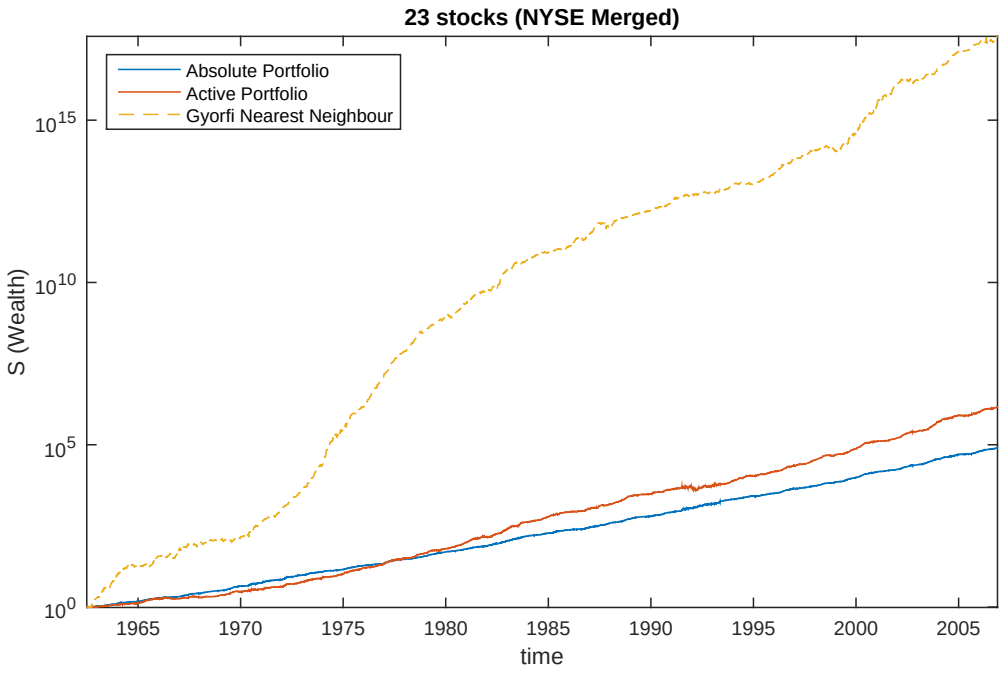


Figure 5.9: Wealth gained when investing in all 23 stocks from the different strategies.

5.5 JSE Daily Data

Experiments were run for both the absolute and active portfolios on all four datasets (*close-to-close*, *open-to-close*, *close-to-open* and *open-to-open*) and the results compared to the wealth achieved from the implemented versions of the Györfi nearest neighbour algorithm [21] and the Universal Portfolio method [10].

The experiments were carried out on the following stock combinations:

i. Anglogold Ashanti Ltd. (ANG) and Anglo American PLC (AGL)

ii. Standard Bank Grp. Ltd. (SBK) and Firstrand Ltd. (FSR)

iii. Tiger Brands Ltd. (TBS) and Woolworths Holdings (WHL)

iv. 10 JSE Stocks

v. 20 JSE Stocks

vi. 30 JSE Stocks

5.5.1 ANG and AGL

Close-to-Close

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	4.05	13.10	9635.16
Active Portfolio	1.28	3.40	9927.13
Györfi Nearest Neighbour (R)	4.02	13.27	13624.61
Universal Portfolio	2.52		
Best Stock	3.608		

Table 5.16: Comparison of wealth gained from ANG and AGL on the *close-to-close* dataset for the different strategies.

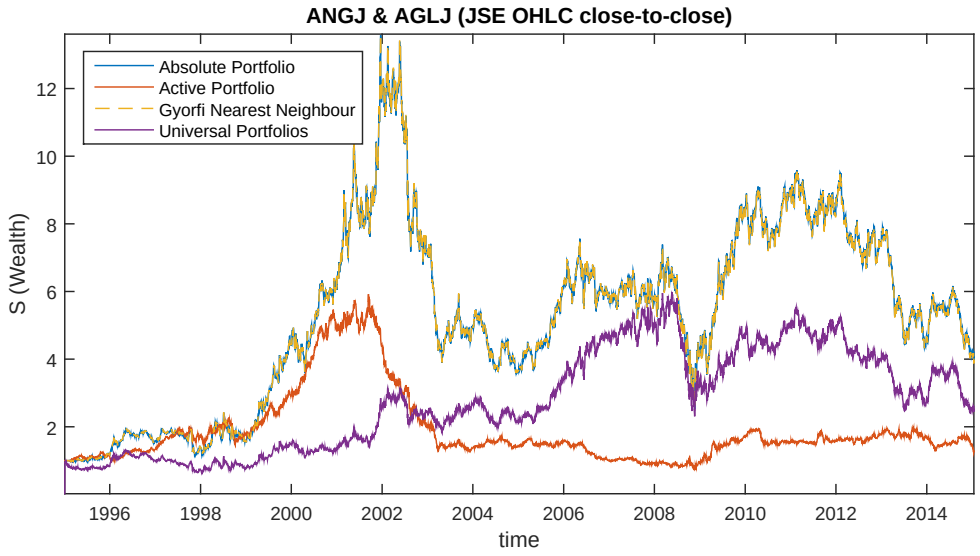


Figure 5.10: Wealth gained from ANG and AGL on the *close-to-close* dataset for the different strategies.

Close-to-Open

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	1.24	2.60	9790.80
Active Portfolio	1.04	2.82	9870.83
Györfi Nearest Neighbour (R)	1.24	2.56	12860.68
Universal Portfolio	0.97		
Best Stock	2.661		

Table 5.17: Comparison of wealth gained from ANG and AGL on the *close-to-open* dataset for the different strategies.

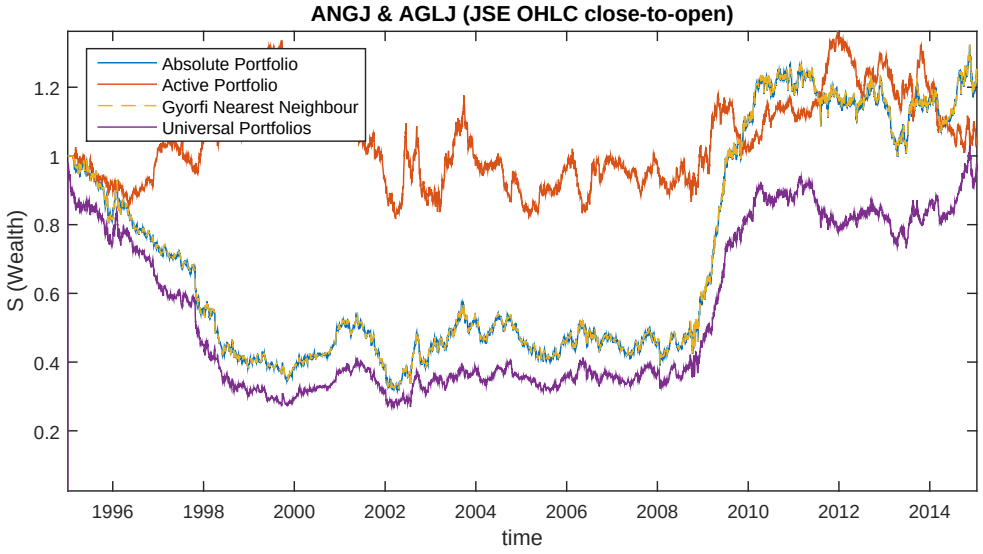


Figure 5.11: Wealth gained from ANG and AGL on the *close-to-open* dataset for the different strategies.

Open-to-Close

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	97.75	395.58	11191.85
Active Portfolio	50.21	91.82	10653.35
Györfi Nearest Neighbour (R)	97.53	398.74	16447.74
Universal Portfolio	3.41		
Best Stock	4.677		

Table 5.18: Comparison of wealth gained from ANG and AGL on the *open-to-close* dataset for the different strategies.

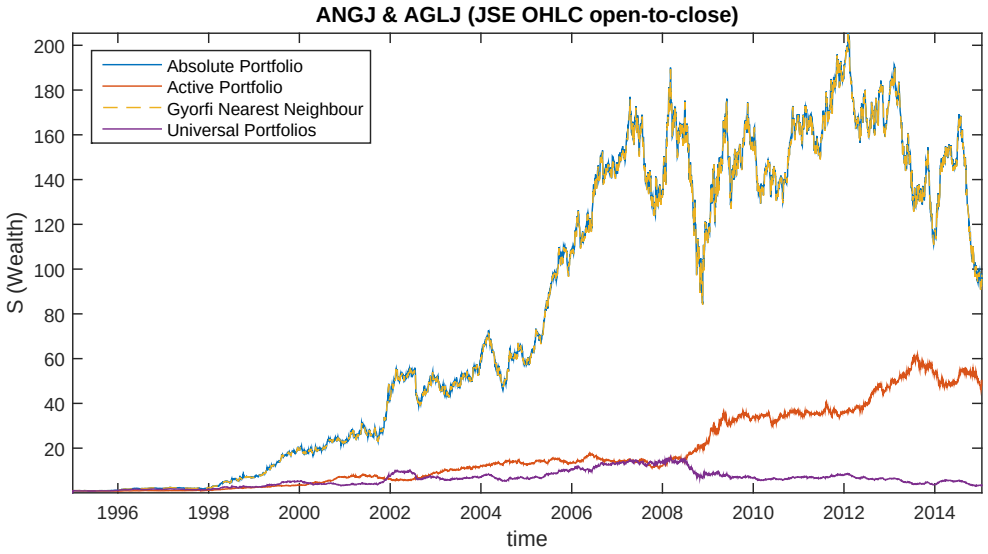


Figure 5.12: Wealth gained from ANG and AGL on the *open-to-close* dataset for the different strategies.

Open-to-Open

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	5.11	26.65	9618.84
Active Portfolio	1.43	9.10	10646.90
Györfi Nearest Neighbour (R)	5.07	25.66	13857.72
Universal Portfolio	2.28		
Best Stock	3.178		

Table 5.19: Comparison of wealth gained from ANG and AGL on the *open-to-open* dataset for the different strategies.

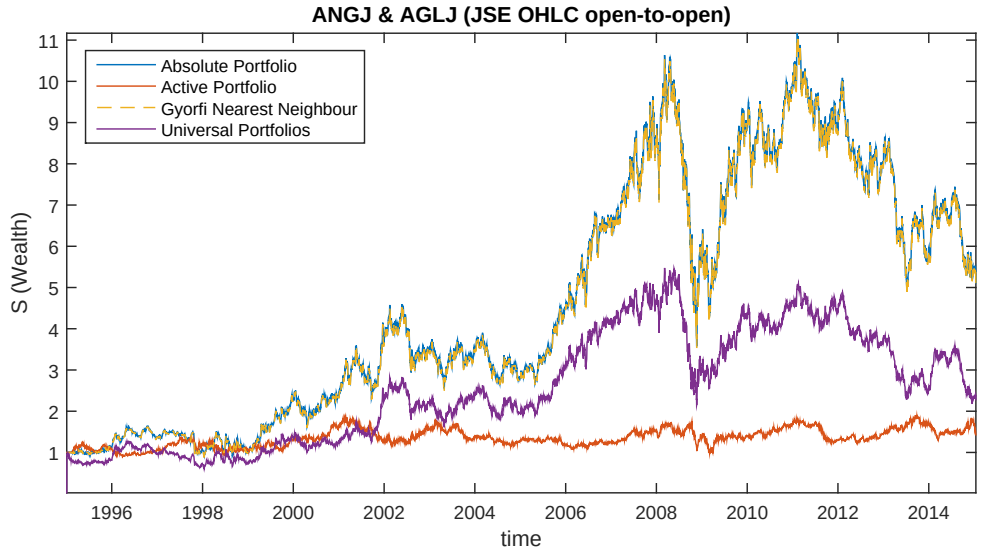


Figure 5.13: Wealth gained from ANG and AGL on the *open-to-open* dataset for the different strategies.

Comparison of Price Relative Datasets

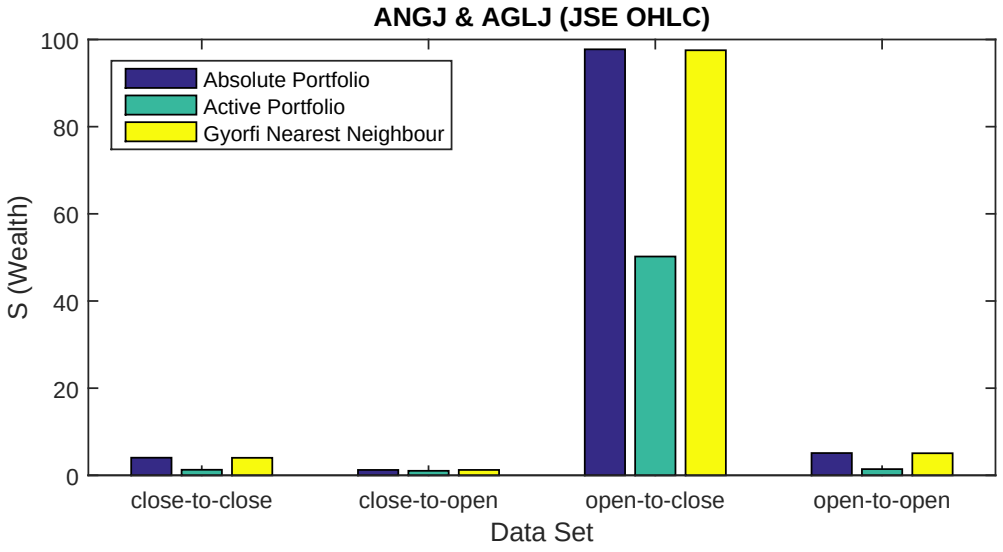


Figure 5.14: Comparison of wealth gained from ANG and AGL for the different datasets

5.5.2 SBK and FSR

Close-to-Close

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	55.53	187.62	11146.49
Active Portfolio	7.77	13.40	11844.85
Györfi Nearest Neighbour (R)	55.60	189.08	17033.11
Universal Portfolio	18.17		
Best Stock	21.090		

Table 5.20: Comparison of wealth gained from SBK and FSR on the *close-to-close* dataset for the different strategies.

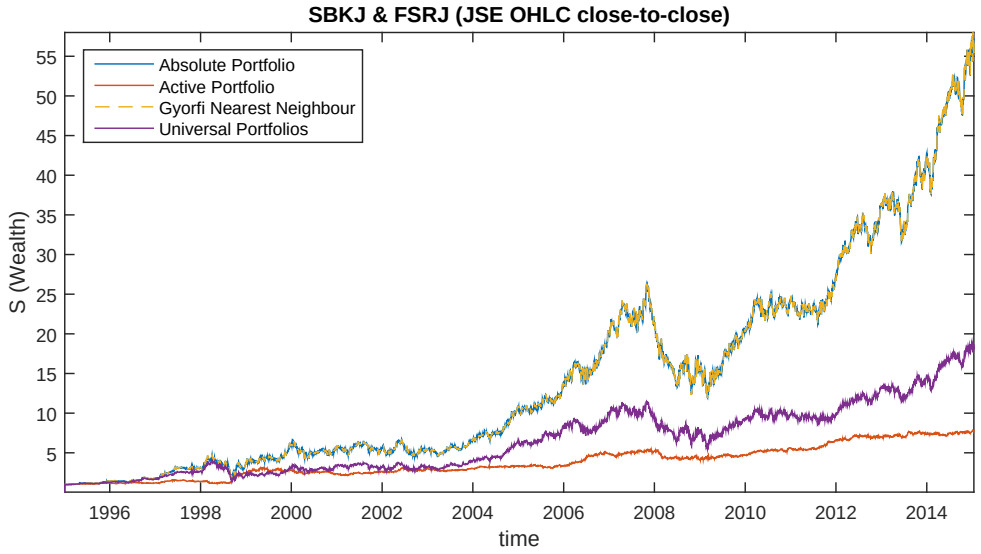


Figure 5.15: Wealth gained from ANG and AGL on the *close-to-close* dataset for the different strategies.

Close-to-Open

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	12.35	25.13	9316.14
Active Portfolio	1.57	3.06	9491.70
Györfi Nearest Neighbour (R)	12.31	25.14	14686.09
Universal Portfolio	7.89		
Best Stock	12.465		

Table 5.21: Comparison of wealth gained from SBK and FSR on the *close-to-open* dataset for the different strategies.

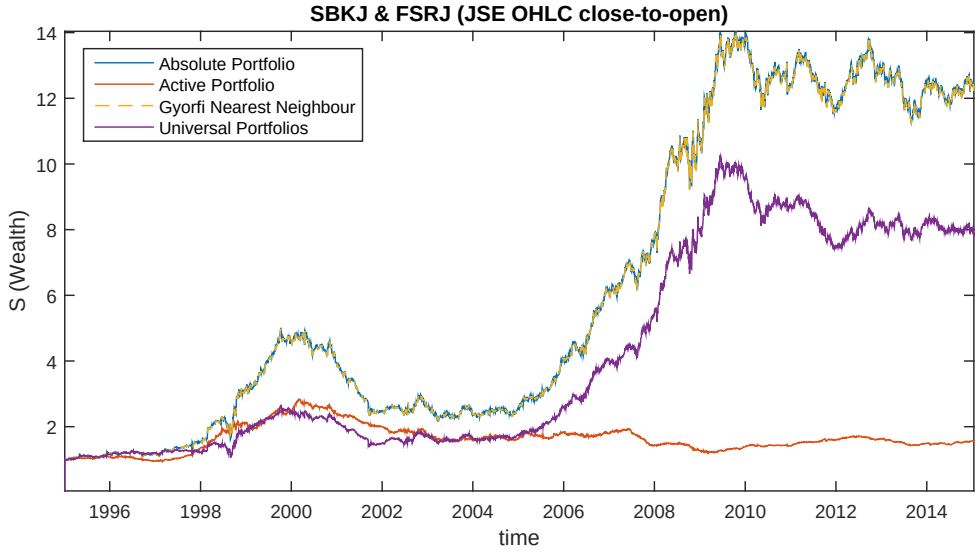


Figure 5.16: Wealth gained from ANG and AGL on the *close-to-open* dataset for the different strategies.

Open-to-Close

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	2.53	5.12	9450.77
Active Portfolio	1.04	1.80	9519.02
Györfi Nearest Neighbour (R)	2.53	5.00	15057.26
Universal Portfolio	2.66		
Best Stock	2.972		

Table 5.22: Comparison of wealth gained from SBK and FSR on the *open-to-close* dataset for the different strategies.

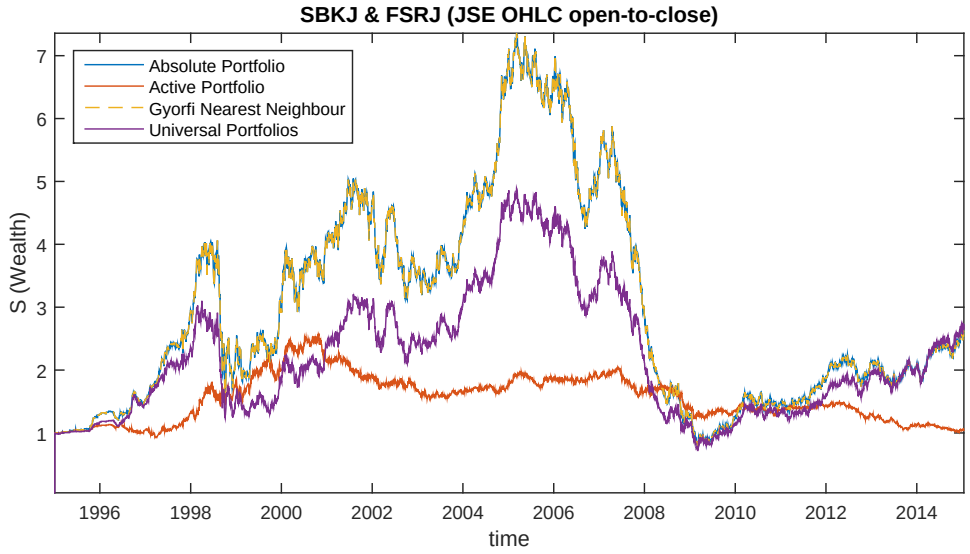


Figure 5.17: Wealth gained from ANG and AGL on the *open-to-close* dataset for the different strategies.

Open-to-Open

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	113.43	412.75	9463.65
Active Portfolio	13.97	18.87	9538.53
Györfi Nearest Neighbour (R)	113.02	415.45	13284.45
Universal Portfolio	20.29		
Best Stock	20.548		

Table 5.23: Comparison of wealth gained from SBK and FSR on the *open-to-open* dataset for the different strategies.

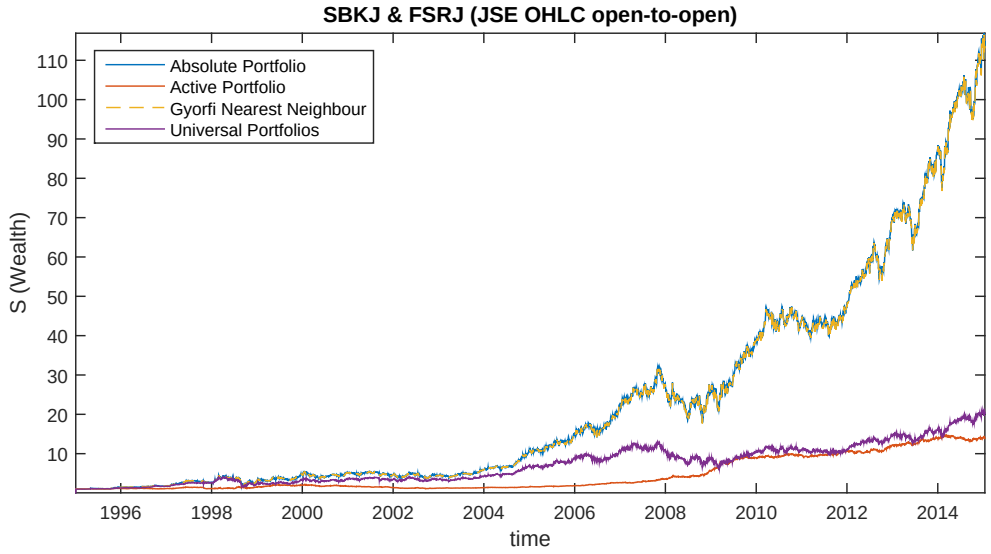


Figure 5.18: Wealth gained from SBK and FSR on the *open-to-open* dataset for the different strategies.

Comparison of Price Relative Datasets

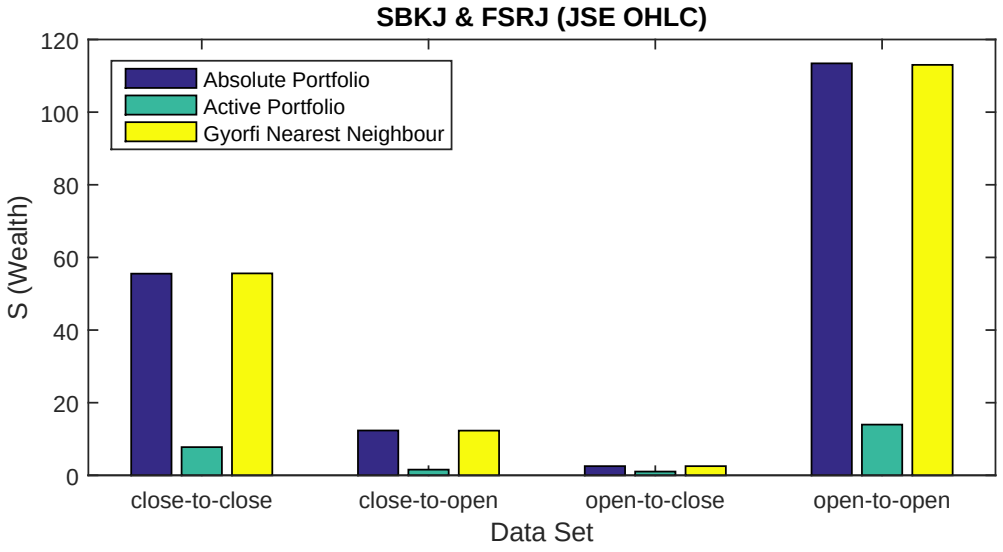


Figure 5.19: Comparison of wealth gained from SBK and FSR for the different datasets

5.5.3 TBS and WHL

Close-to-Close

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	7.07	19.97	8027.34
Active Portfolio	0.49	1.80	7689.70
Györfi Nearest Neighbour (R)	7.07	19.49	12436.46
Universal Portfolio	8.24		
Best Stock	8.971		

Table 5.24: Comparison of wealth gained from TBS and WHL on the *close-to-close* dataset for the different strategies.

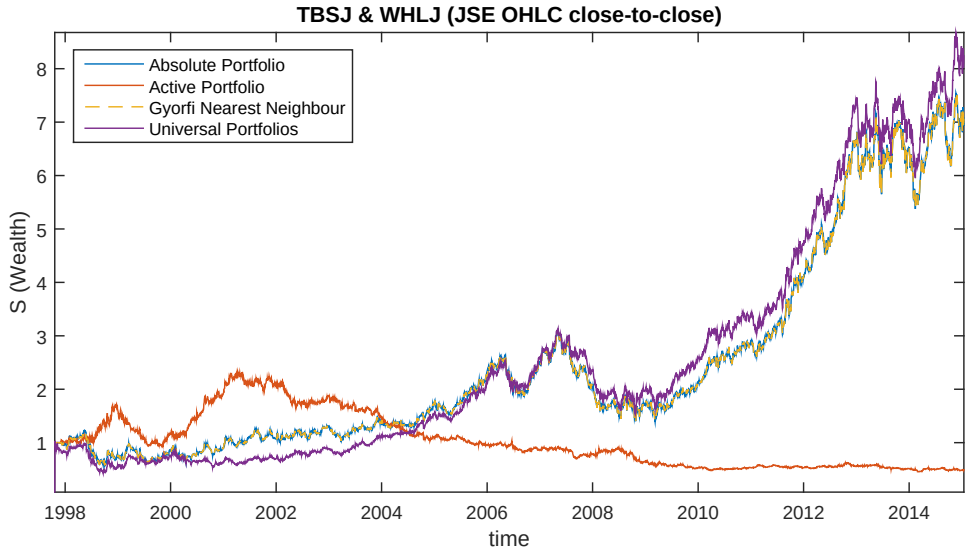


Figure 5.20: Wealth gained from TBS and WHL on the *close-to-close* dataset for the different strategies.

Close-to-Open

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	4.38	9.00	8395.69
Active Portfolio	0.79	1.88	8550.23
Györfi Nearest Neighbour (R)	4.46	8.83	9666.32
Universal Portfolio	4.89		
Best Stock	4.975		

Table 5.25: Comparison of wealth gained from TBS and WHL on the *close-to-open* dataset for the different strategies.

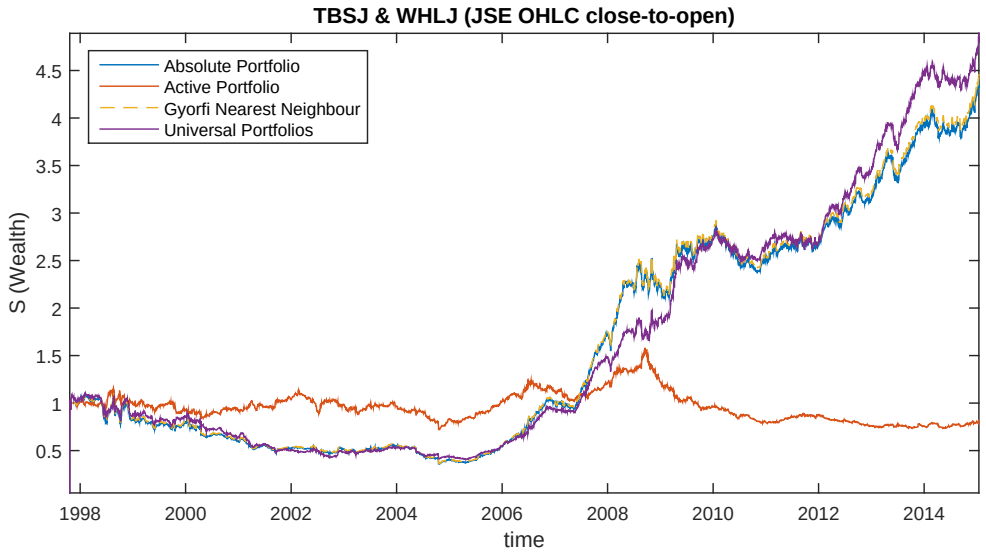


Figure 5.21: Wealth gained from TBS and WHL on the *close-to-open* dataset for the different strategies.

Open-to-Close

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	5.59	12.73	7929.85
Active Portfolio	5.00	7.19	8782.45
Györfi Nearest Neighbour (R)	5.59	12.75	10848.55
Universal Portfolio	1.94		
Best Stock	1.892		

Table 5.26: Comparison of wealth gained from TBS and WHL on the *open-to-close* dataset for the different strategies.

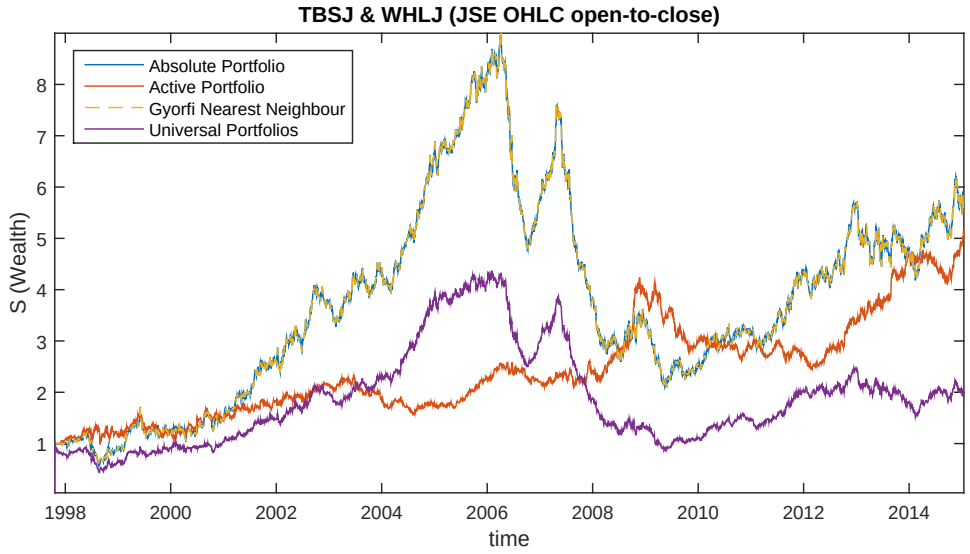


Figure 5.22: Wealth gained from TBS and WHL on the *open-to-close* dataset for the different strategies.

Open-to-Open

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	45.30	266.17	7170.39
Active Portfolio	5.52	28.54	7205.59
Györfi Nearest Neighbour (R)	45.80	273.44	12233.72
Universal Portfolio	8.81		
Best Stock	9.235		

Table 5.27: Comparison of wealth gained from TBS and WHL on the *open-to-open* dataset for the different strategies.

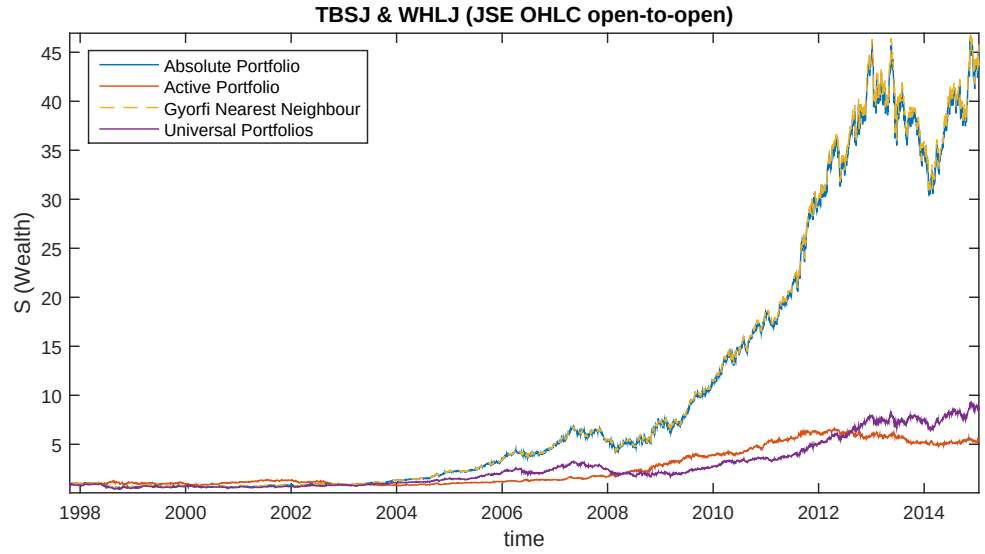


Figure 5.23: Wealth gained from TBS and WHL on the *open-to-open* dataset for the different strategies.

Comparison of Price Relative Datasets

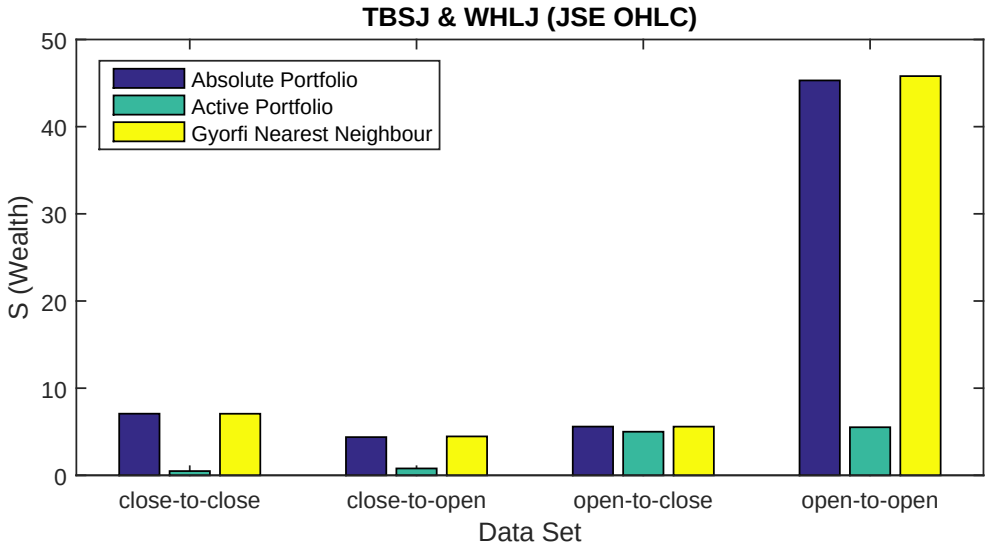


Figure 5.24: Comparison of wealth gained from TBS and WHL for the different datasets

5.5.4 10 Stocks

The following 10 stocks were selected:

1. Anglo American PLC
2. Anglogold Ashanti Ltd.
3. BHP Billiton PLC
4. Comp. Fin Richemont
5. Impala Platinum Hld. Ltd.
6. MTN Group Ltd.
7. Naspers Ltd.
8. SABMiller Plc
9. Sasol Ltd.
10. Standard Bank Grp. Ltd.

Close-to-Close

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	68.49	135.03	10346.55
Active Portfolio	9.28	12.84	10177.76
Györfi Nearest Neighbour (R)	194.76	854.62	19112.52
Best Stock	89.722		

Table 5.28: Comparison of wealth gained from 10 stocks on the *close-to-close* dataset for the different strategies.

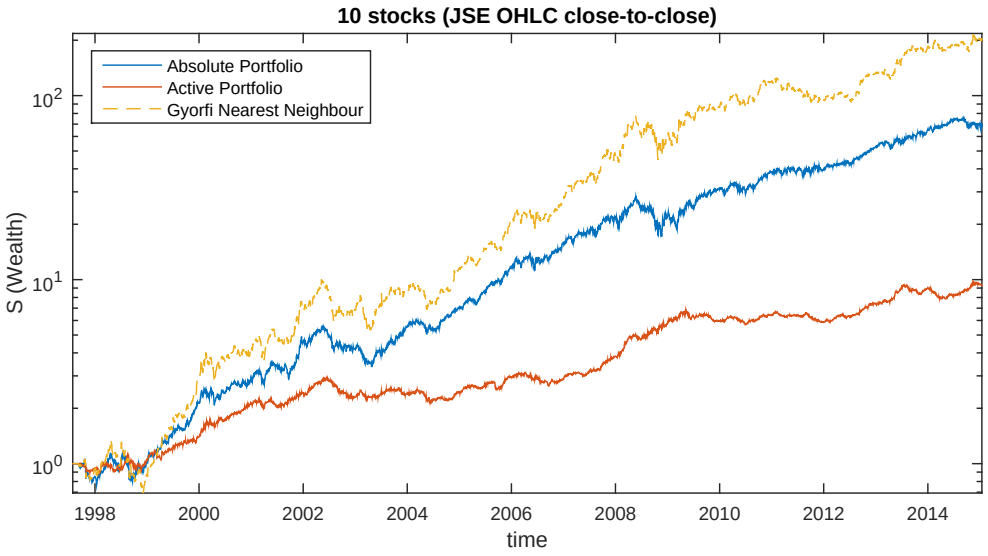


Figure 5.25: Wealth gained from 10 stocks on the *close-to-close* dataset for the different strategies.

Close-to-Open

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	7.11	11.43	9990.58
Active Portfolio	2.68	3.39	9944.98
Györfi Nearest Neighbour (R)	13.45	30.53	17016.54
Best Stock	56.363		

Table 5.29: Comparison of wealth gained from 10 stocks on the *close-to-open* dataset for the different strategies.

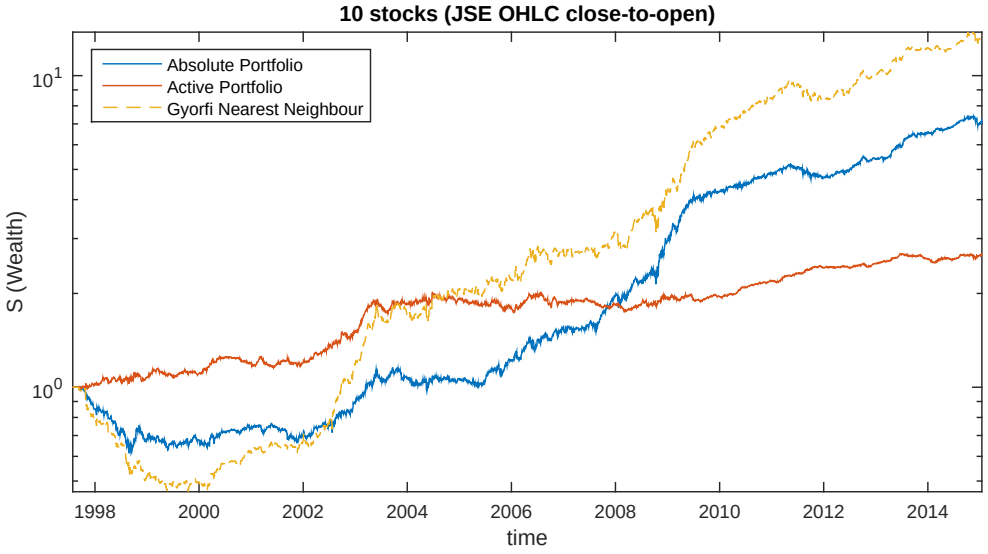


Figure 5.26: Wealth gained from 10 stocks on the *close-to-open* dataset for the different strategies.

Open-to-Close

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	15.90	33.40	10092.90
Active Portfolio	15.21	19.86	10228.13
Györfi Nearest Neighbour (R)	101.80	415.97	16014.21
Best Stock	169.830		

Table 5.30: Comparison of wealth gained from 10 stocks on the *open-to-close* dataset for the different strategies.

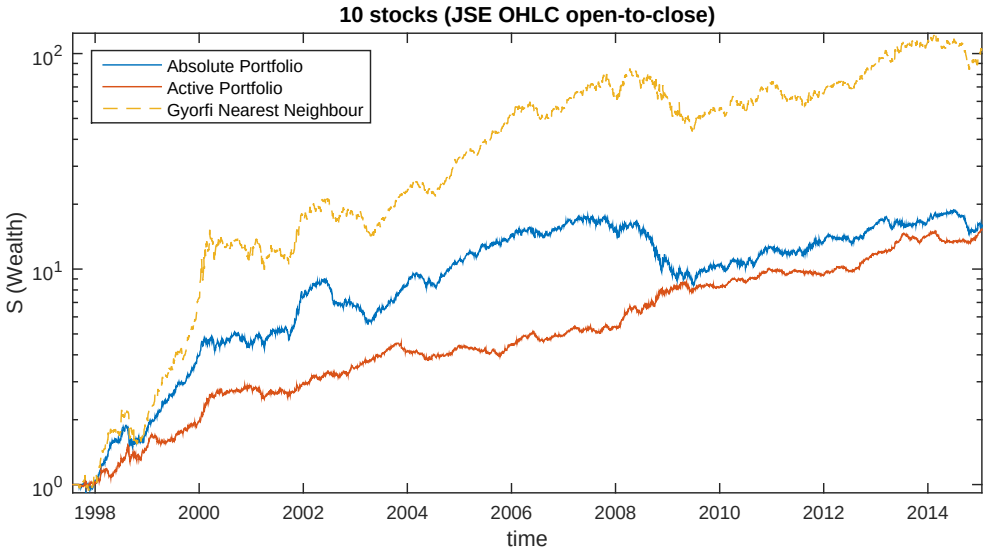


Figure 5.27: Wealth gained from 10 stocks on the *open-to-close* dataset for the different strategies.

Open-to-Open

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	60.62	342.40	9991.06
Active Portfolio	12.03	20.57	9959.97
Györfi Nearest Neighbour (R)	254.00	2292.47	15967.62
Best Stock	69.494		

Table 5.31: Comparison of wealth gained from 10 stocks on the *open-to-open* dataset for the different strategies.

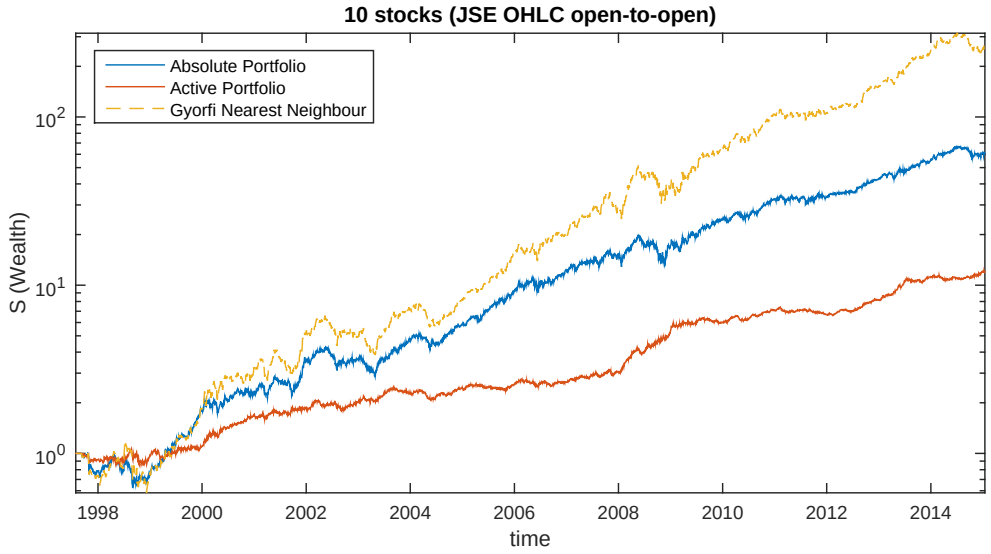


Figure 5.28: Wealth gained from 10 stocks on the *open-to-open* dataset for the different strategies.

Comparison of Price Relative Datasets

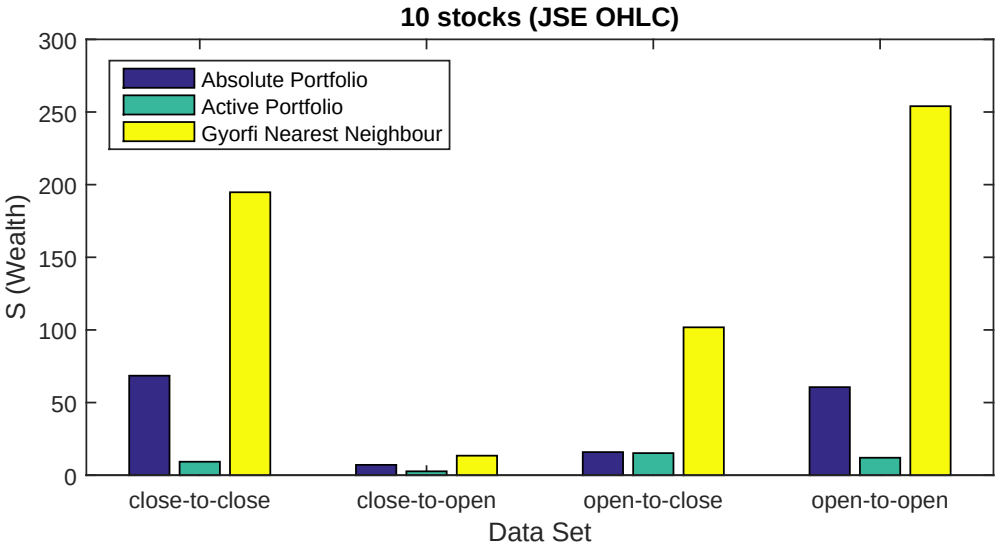


Figure 5.29: Comparison of wealth gained from 10 stocks for the different datasets

5.5.5 20 Stocks

The following 20 stocks were selected:

1. Absa Group Ltd.
2. Anglo American PLC
3. Anglo Platinum Ltd.
4. Anglogold Ashanti Ltd.
5. BHP Billiton PLC
6. Bidvest Ltd.
7. Comp. Fin Richemont
8. Firstrand Ltd.
9. Gold Fields Ltd.
10. Impala Platinum Hld. Ltd.
11. MTN Group Ltd.
12. Naspers Ltd.
13. Nedbank Group Ltd.
14. Old Mutual Plc
15. Remgro Ltd.
16. SABMiller Plc
17. Sanlam Ltd.
18. Sasol Ltd.
19. Standard Bank Grp. Ltd.
20. Steinhoff Inter. Hldgs

Close-to-Close

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	16.20	63.18	9286.16
Active Portfolio	9.52	12.11	8743.07
Györfi Nearest Neighbour (R)	98.84	330.37	16101.32
Best Stock	89.722		

Table 5.32: Comparison of wealth gained from 20 stocks on the *close-to-close* dataset for the different strategies.

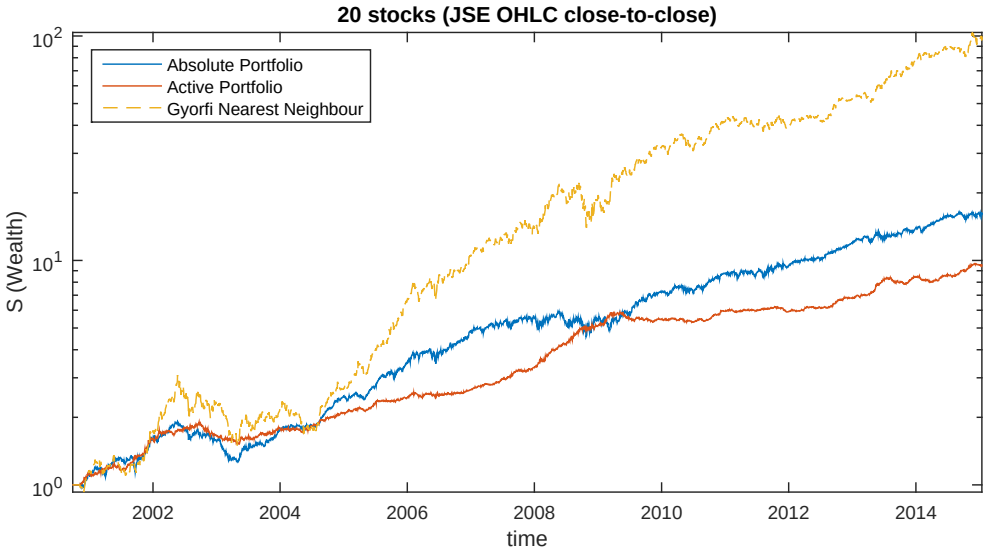


Figure 5.30: Wealth gained from 20 stocks on the *close-to-close* dataset for the different strategies.

Close-to-Open

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	8.82	16.87	9792.60
Active Portfolio	5.28	5.24	9211.79
Györfi Nearest Neighbour (R)	11.00	27.01	16951.11
Best Stock	56.363		

Table 5.33: Comparison of wealth gained from 20 stocks on the *close-to-open* dataset for the different strategies.

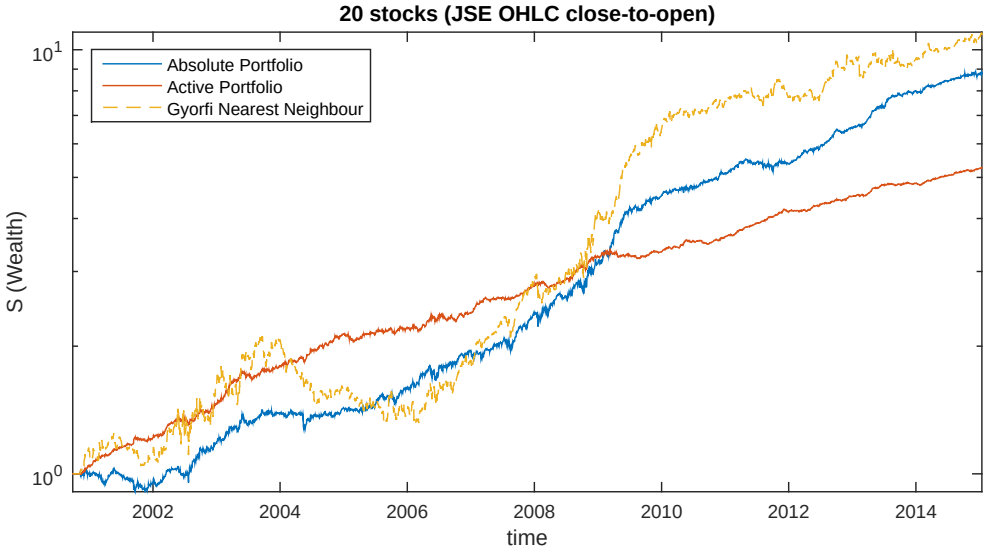


Figure 5.31: Wealth gained from 10 stocks on the *close-to-open* dataset for the different strategies.

Open-to-Close

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	4.80	8.88	6929.55
Active Portfolio	6.80	6.07	7947.06
Györfi Nearest Neighbour (R)	30.26	104.46	17818.14
Best Stock	169.830		

Table 5.34: Comparison of wealth gained from 20 stocks on the *open-to-close* dataset for the different strategies.

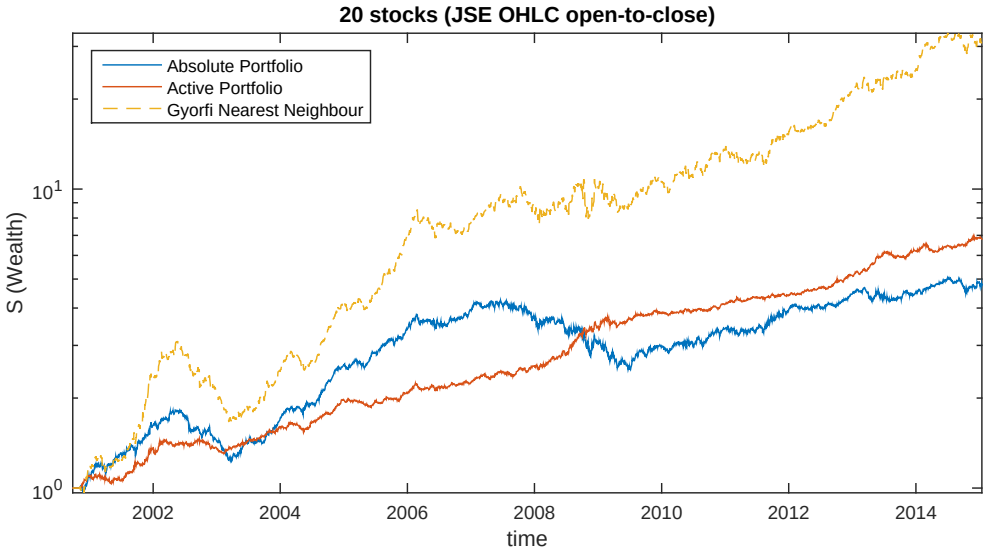


Figure 5.32: Wealth gained from 20 stocks on the *open-to-close* dataset for the different strategies.

Open-to-Open

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	14.73	73.61	7192.71
Active Portfolio	9.13	14.76	7279.49
Györfi Nearest Neighbour (R)	44.88	192.99	17479.31
Best Stock	69.494		

Table 5.35: Comparison of wealth gained from 20 stocks on the *open-to-open* dataset for the different strategies.

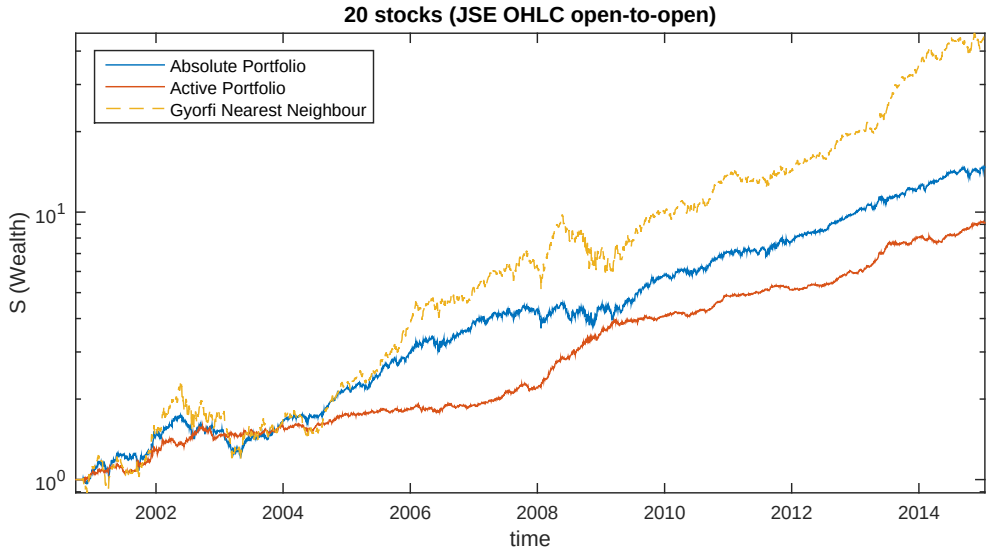


Figure 5.33: Wealth gained from 20 stocks on the *open-to-open* dataset for the different strategies.

Comparison of Price Relative Datasets

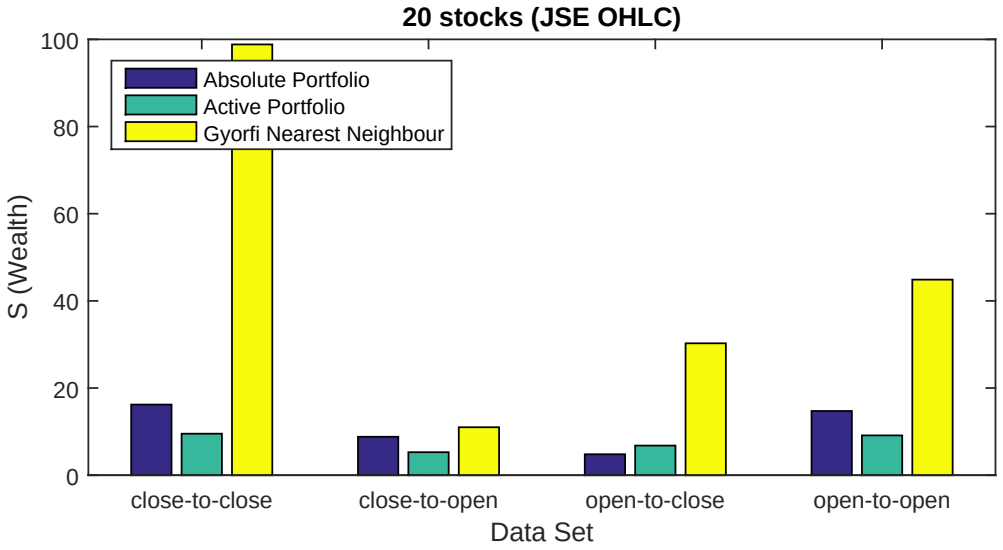


Figure 5.34: Comparison of wealth gained from 20 stocks for the different datasets

5.5.6 30 Stocks

The following 30 stocks were selected:

1. Absa Group Ltd.
2. African Rainbow Minerals
3. Anglo American PLC
4. Anglo Platinum Ltd.
5. Anglogold Ashanti Ltd.
6. Aspen Pharmacare Hldgs
7. BHP Billiton PLC
8. Bidvest Ltd.
9. Comp. Fin Richemont
10. Discovery Holdings Ltd.
11. Firstrand Ltd.
12. Gold Fields Ltd.
13. Growthpoint Prop Ltd.
14. Impala Platinum Hld. Ltd.
15. Imperial Hldgs Ltd.
16. Mediclinic International Ltd.
17. MTN Group Ltd.
18. Naspers Ltd.
19. Nedbank Group Ltd.
20. Old Mutual Plc
21. Remgro Ltd.
22. SABMiller Plc
23. Sanlam Ltd.
24. Sasol Ltd.
25. Shoprite Hldgs Ltd.
26. Standard Bank Grp. Ltd.
27. Steinhoff Inter. Hldgs
28. Tiger Brands Ltd.
29. Truworths International
30. Woolworths Hldgs

Close-to-Close

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	20.48	58.27	7582.84
Active Portfolio	7.34	6.24	7902.99
Györfi Nearest Neighbour (R)	124.91	590.51	22071.10
Best Stock	85.025		

Table 5.36: Comparison of wealth gained from 30 stocks on the *close-to-close* dataset for the different strategies.

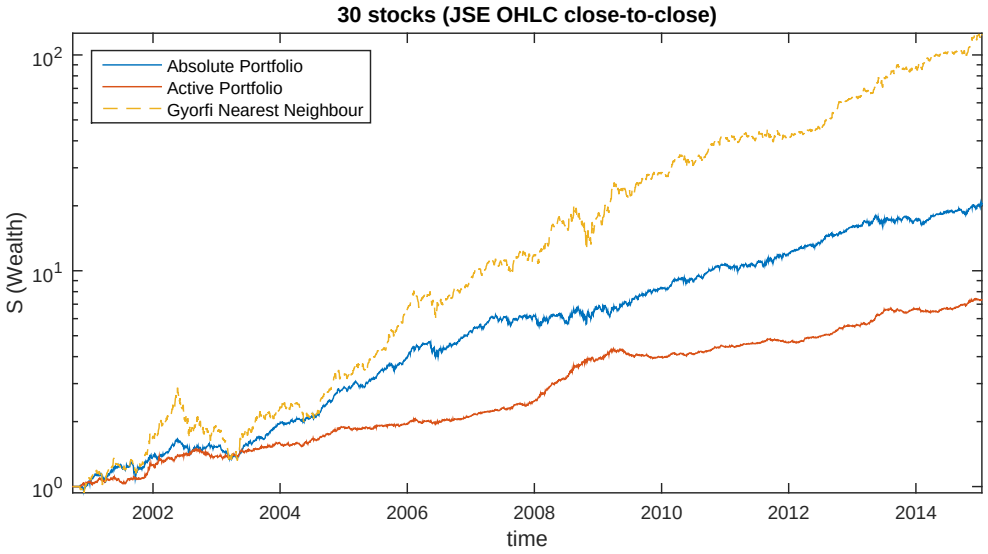


Figure 5.35: Wealth gained from 30 stocks on the *close-to-close* dataset for the different strategies.

Close-to-Open

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	8.07	15.80	7619.28
Active Portfolio	5.66	5.50	7926.28
Györfi Nearest Neighbour (R)	22.53	54.80	22668.78
Best Stock	51.018		

Table 5.37: Comparison of wealth gained from 30 stocks on the *close-to-open* dataset for the different strategies.

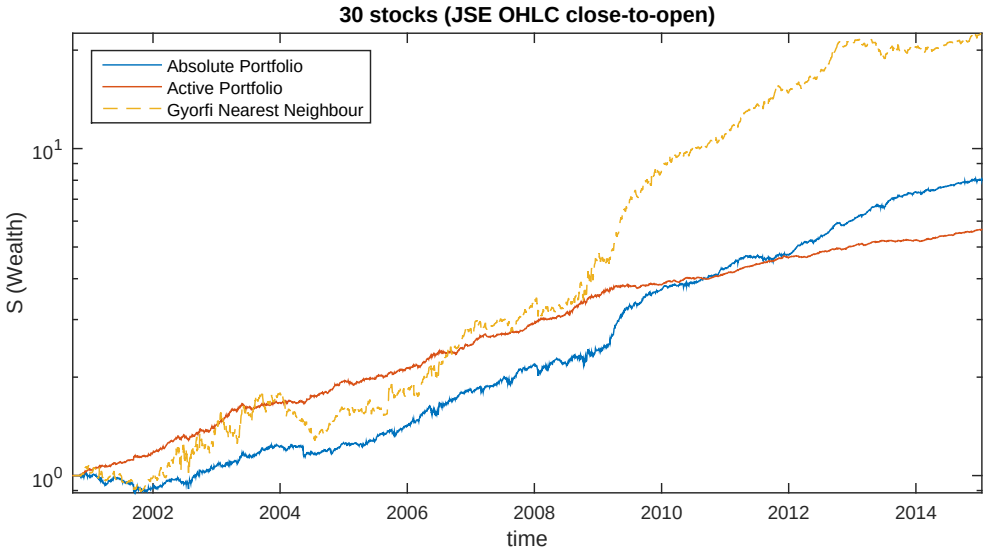


Figure 5.36: Wealth gained from 30 stocks on the *close-to-open* dataset for the different strategies.

Open-to-Close

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	5.16	11.98	7826.80
Active Portfolio	4.58	5.57	7926.12
Györfi Nearest Neighbour (R)	50.17	317.92	21033.86
Best Stock	1278.827		

Table 5.38: Comparison of wealth gained from 30 stocks on the *open-to-close* dataset for the different strategies.

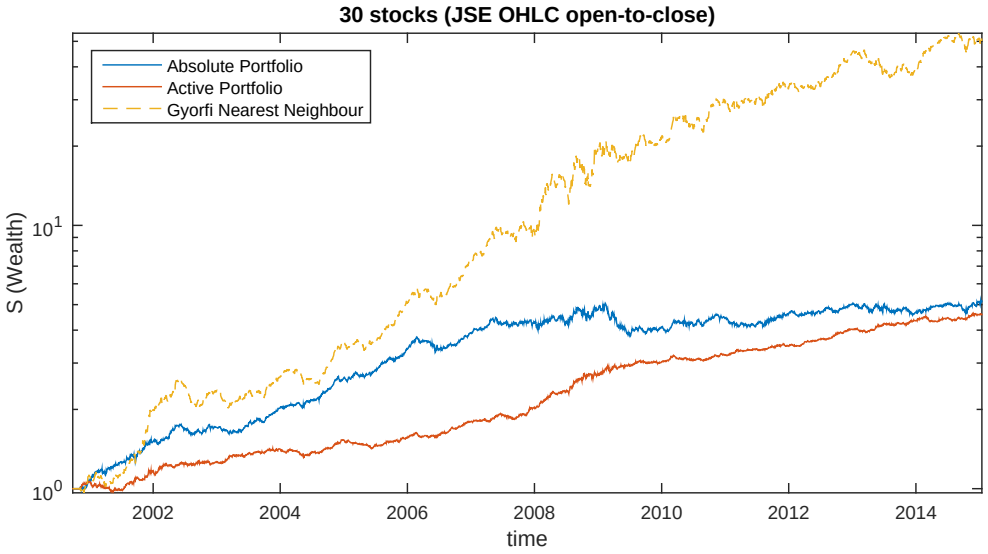


Figure 5.37: Wealth gained from 30 stocks on the *open-to-close* dataset for the different strategies.

Open-to-Open

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	21.54	73.88	7775.83
Active Portfolio	11.48	18.89	7701.35
Györfi Nearest Neighbour (R)	161.16	1069.81	20687.51
Best Stock	131.886		

Table 5.39: Comparison of wealth gained from 30 stocks on the *open-to-open* dataset for the different strategies.

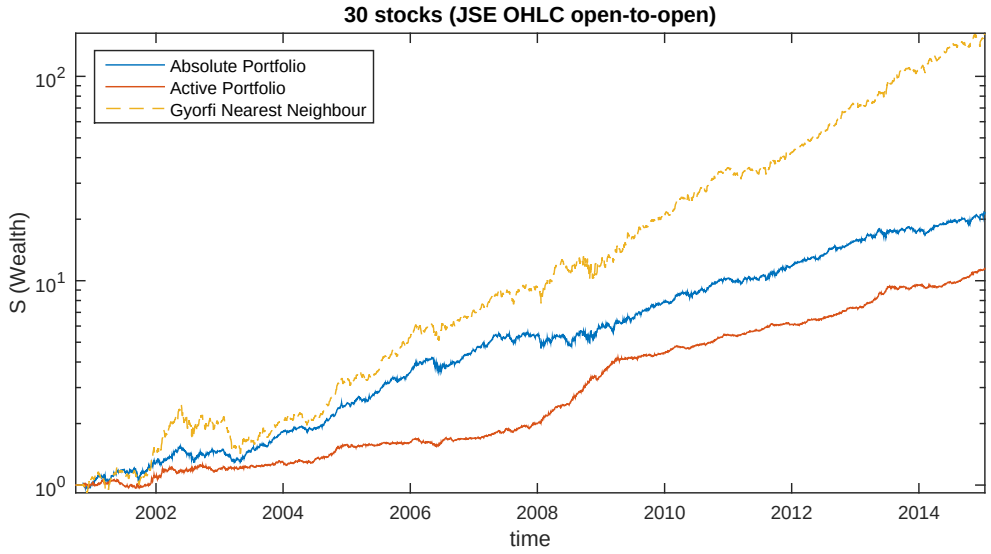


Figure 5.38: Wealth gained from 30 stocks on the *open-to-open* dataset for the different strategies.

Comparison of Price Relative Datasets

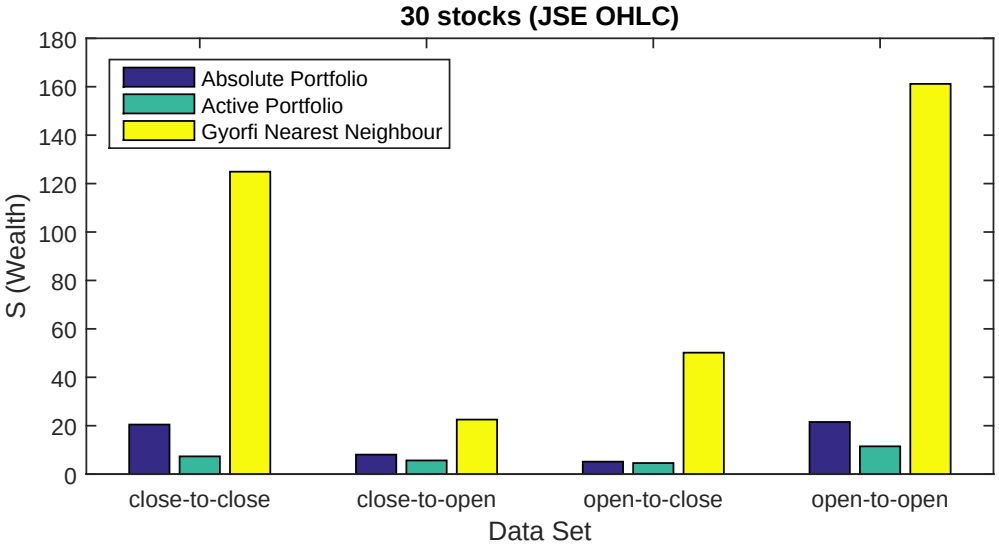


Figure 5.39: Comparison of wealth gained from 30 stocks for the different datasets

5.6 JSE Intraday Data

Experiments were run for both the absolute and active portfolios and the results compared to the wealth achieved from the implemented versions of the Györfi nearest neighbour algorithm [21] and the Universal Portfolio method [10].

The experiments were carried out on the following stock combinations:

i. Anglogold Ashanti Ltd. (ANG) and Anglo American PLC (AGL)

ii. Standard Bank Grp. Ltd. (SBK) and Firstrand Ltd. (FSR)

iii. Tiger Brands Ltd. (TBS) and Woolworths Holdings (WHL)

iv. 10 JSE Stocks

v. 20 JSE Stocks

vi. 30 JSE Stocks

vii. 20 JSE Stocks (with clusters)

5.6.1 ANG and AGL

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	1.38	3.33	177033.42
Active Portfolio	2.16	5.01	179485.68
Györfi Nearest Neighbour (R)	1.36	3.21	197655.19
Universal Portfolio	0.66		
Best Stock	0.874		

Table 5.40: Comparison of wealth gained from ANG and AGL for the different strategies.

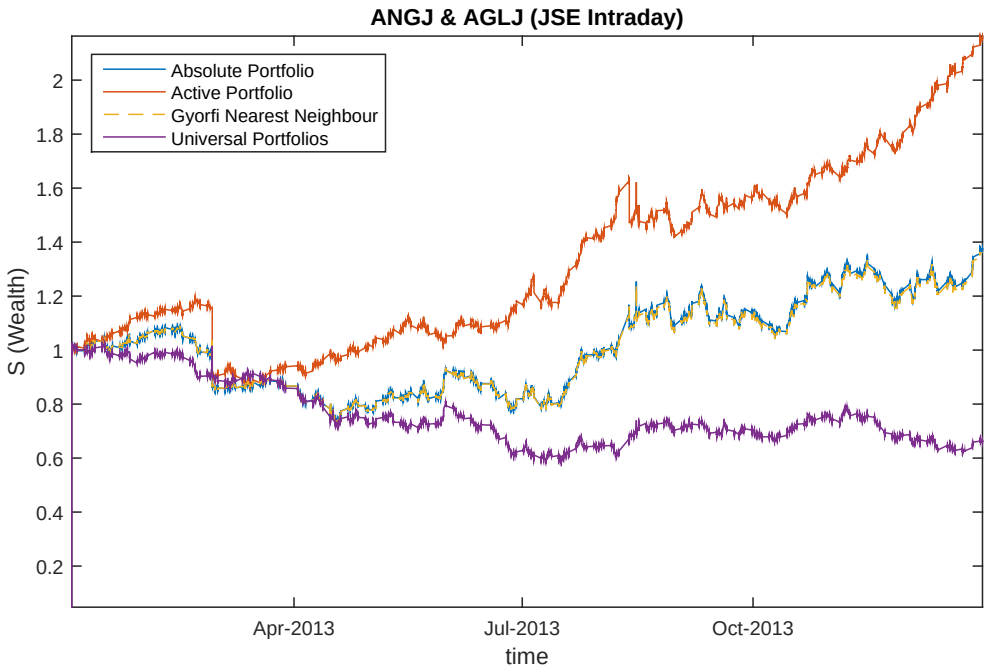


Figure 5.40: Wealth gained from ANG and AGL for the different strategies.

5.6.2 SBK and FSR

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	1.82	2.19	191591.89
Active Portfolio	2.02	2.01	192963.34
Györfi Nearest Neighbour (R)	1.82	2.15	211923.58
Universal Portfolio	1.08		
Best Stock	1.084		

Table 5.41: Comparison of wealth gained from SBK and FSR for the different strategies.

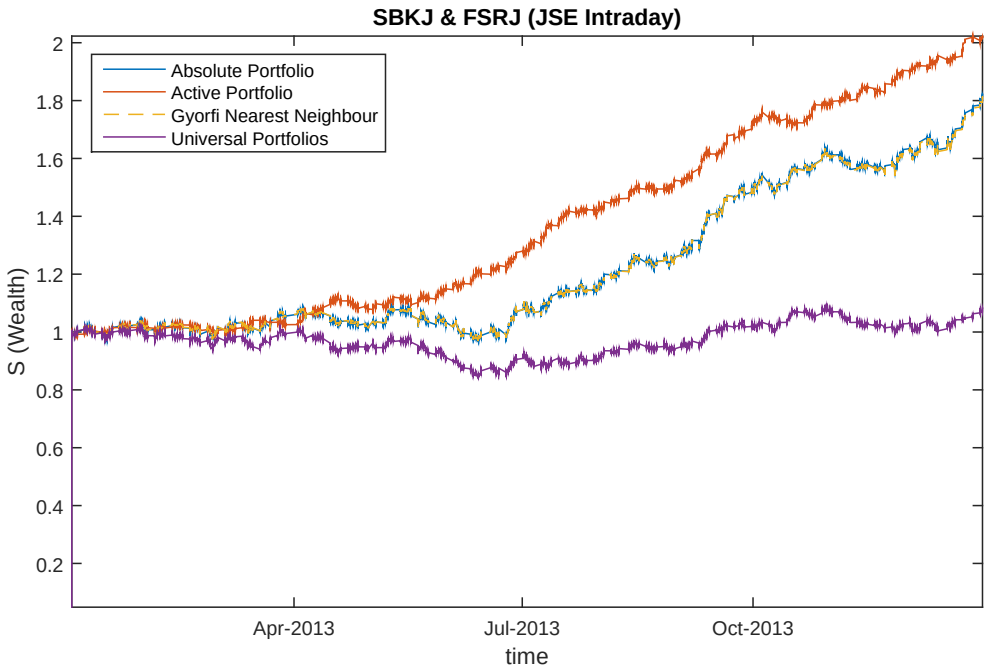


Figure 5.41: Wealth gained from SBK and FSR for the different strategies.

5.6.3 TBS and WHL

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	1.95	3.06	183586.69
Active Portfolio	2.24	3.29	184105.51
Györfi Nearest Neighbour (R)	1.95	2.99	192232.43
Universal Portfolio	0.91		
Best Stock	0.991		

Table 5.42: Comparison of wealth gained from TBS and WHL for the different strategies.

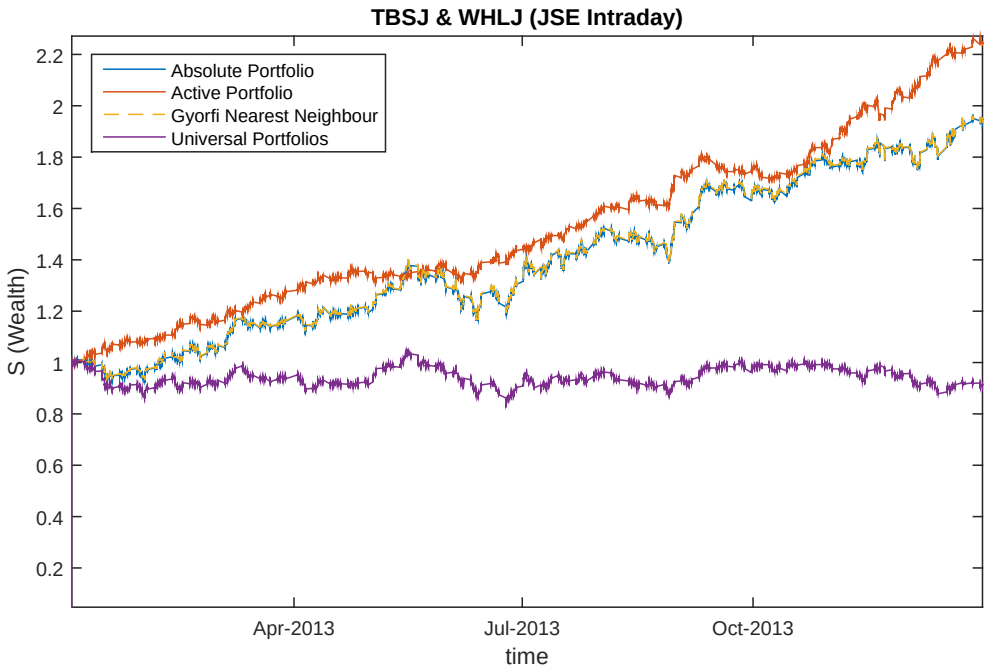


Figure 5.42: Wealth gained from TBS and WHL for the different strategies.

5.6.4 MTN and VOD

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	2.13	3.55	177338.23
Active Portfolio	2.13	3.25	178564.85
Györfi Nearest Neighbour (R)	2.13	3.57	188460.89
Universal Portfolio	1.12		
Best Stock	1.181		

Table 5.43: Comparison of wealth gained from MTN and VOD for the different strategies.

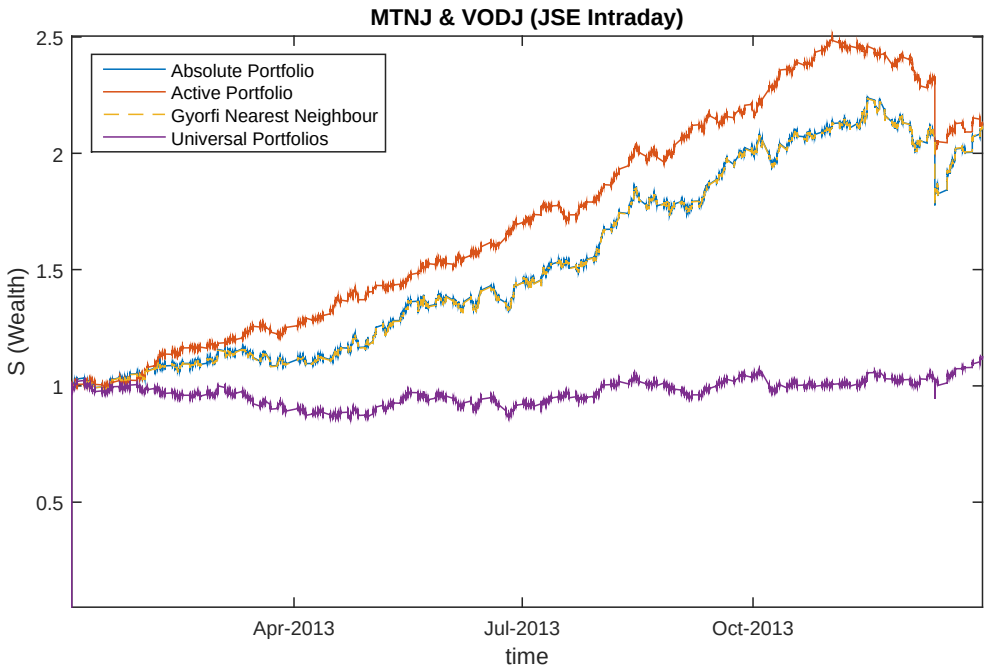


Figure 5.43: Wealth gained from MTN and VOD for the different strategies.

5.6.5 10 Stocks

The following 10 stocks were selected:

1. Anglo American PLC
2. Anglogold Ashanti Ltd.
3. BHP Billiton PLC
4. Comp. Fin Richemont
5. Impala Platinum Hld. Ltd.
6. MTN Group Ltd.
7. Naspers Ltd.
8. SABMiller Plc
9. Sasol Ltd.
10. Standard Bank Grp. Ltd.

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	1.89	2.79	235571.99
Active Portfolio	3.86	5.40	245606.82
Györfi Nearest Neighbour (R)	3.95	14.05	305828.26
Best Stock	1.934		

Table 5.44: Comparison of wealth gained from 10 stocks for the different strategies.

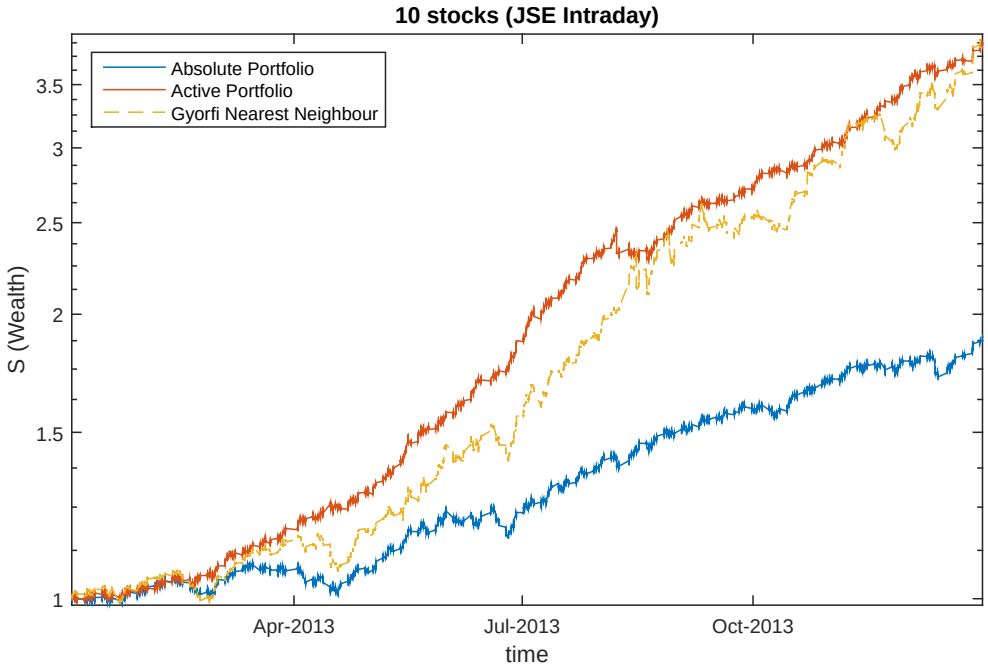


Figure 5.44: Wealth gained from 10 stocks for the different strategies.

5.6.6 20 Stocks

The following 20 stocks were selected:

1. Anglo American PLC
2. Anglo Platinum Ltd.
3. Anglogold Ashanti Ltd.
4. BHP Billiton PLC
5. Bidvest Ltd.
6. Comp. Fin Richemont
7. Firstrand Ltd.
8. Gold Fields Ltd.
9. Impala Platinum Hld. Ltd.
10. MTN Group Ltd.
11. Naspers Ltd.
12. Nedbank Group Ltd.
13. Old Mutual Plc
14. Remgro Ltd.
15. SABMiller Plc
16. Sanlam Ltd.
17. Sasol Ltd.
18. Shoprite Hldgs Ltd.
19. Standard Bank Grp. Ltd.
20. Steinhoff Inter. Hldgs

Strategy	Portfolio Wealth	Best Agent Wealth	Running Time
Absolute Portfolio	1.74	2.35	257512.67
Active Portfolio	3.42	4.29	249349.02
Györfi Nearest Neighbour (R)	5.68	21.62	361423.57
Best Stock	1.934		

Table 5.45: Comparison of wealth gained from 20 stocks for the different strategies.

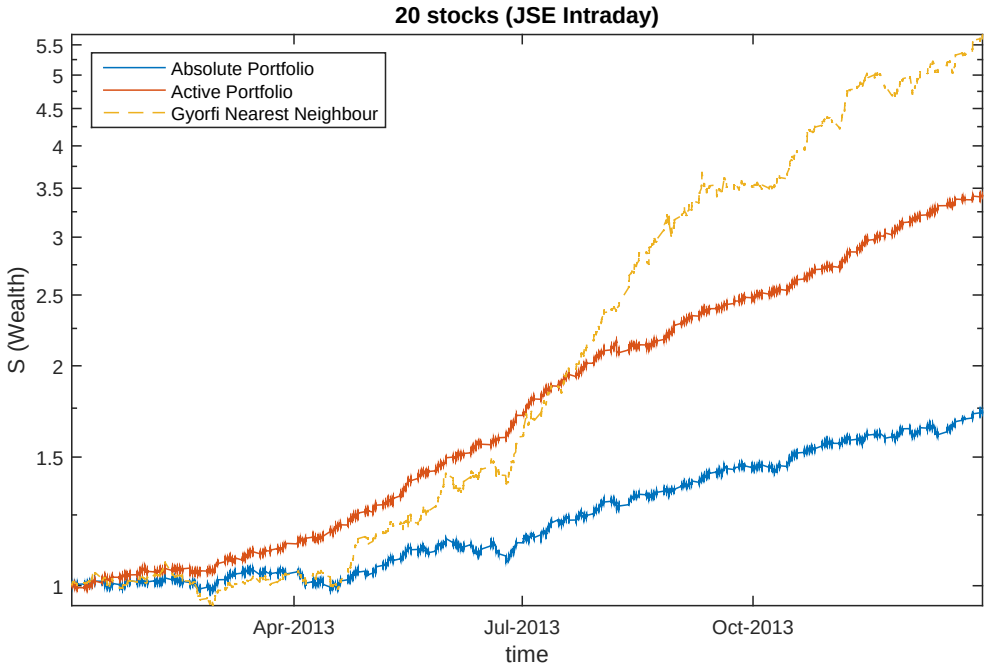


Figure 5.45: Wealth gained from 20 stocks for the different strategies.

5.6.7 30 Stocks

The following 30 stocks were selected:

1. African Rainbow Minerals
2. Anglo American PLC
3. Anglo Platinum Ltd.
4. Anglogold Ashanti Ltd.
5. Aspen Pharmacare Hldgs
6. BHP Billiton PLC
7. Bidvest Ltd.
8. Comp. Fin Richemont
9. Discovery Holdings Ltd.
10. Firstrand Ltd.
11. Gold Fields Ltd.
12. Growthpoint Prop Ltd.
13. Impala Platinum Hld. Ltd.
14. Imperial Hldgs Ltd.
15. Massmart Hldgs Ltd.
16. Mediclinic International Ltd.
17. MTN Group Ltd.

18. Naspers Ltd.
19. Nedbank Group Ltd.
20. Old Mutual Plc
21. Remgro Ltd.
22. SABMiller Plc
23. Sanlam Ltd.
24. Sasol Ltd.
25. Shoprite Hldgs Ltd.
26. Standard Bank Grp. Ltd.
27. Steinhoff Inter. Hldgs
28. Tiger Brands Ltd.
29. Truworths International
30. Woolworths Hldgs

Strategy	**Portfolio Wealth**	**Best Agent Wealth**	**Running Time**
Absolute Portfolio	1.67	2.16	288899.03
Active Portfolio	3.07	3.76	262565.41
Györfi Nearest Neighbour (R)	6.03	12.61	417960.19
Best Stock	1.934		

Table 5.46: Comparison of wealth gained from 30 stocks for the different strategies.

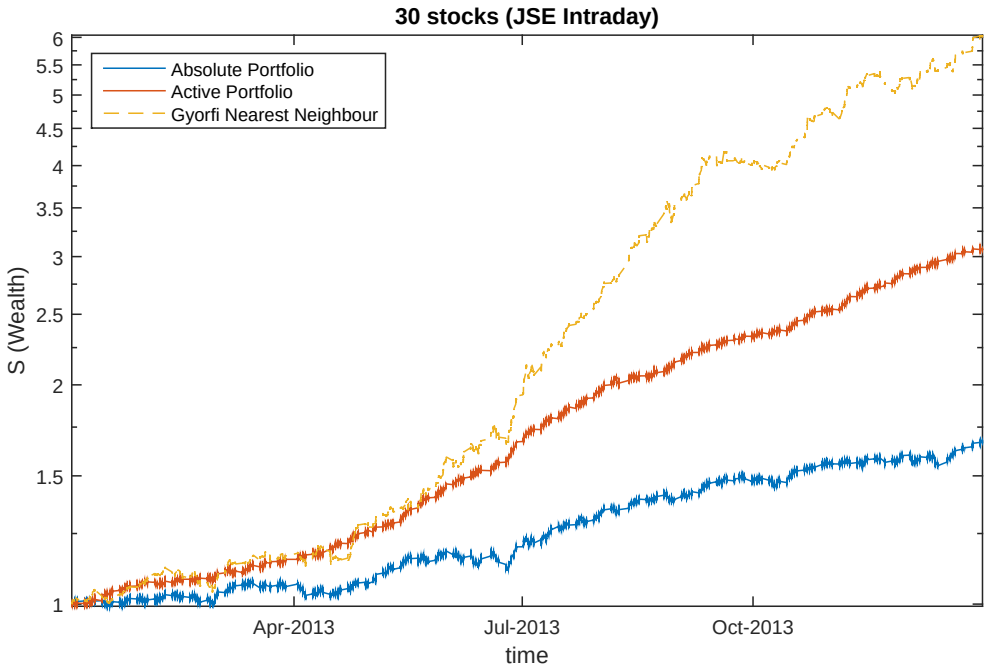


Figure 5.46: Wealth gained from 30 stocks for the different strategies.

5.6.8 20 Stocks Clustered

Investing in JSE Intraday Data with Clusters for Active Portfolios

	Resources	**Industrials**	**Financials**
Stocks	AGLJ	MTNJ	SBKJ
	BILJ	NPNJn	FSRJ
	SOLJ	SABJ	OMLJ
	IMPJ	CFRJ	SLMJ
	ANGJ	REMJ	NEDJ
	GFIJ	SHFJ	
	AMSJ	BVTJ	
		SHPJ	
Best Agent	9.1018	3.4470	3.6698
Total Wealth	4.6368		
Running Time	635243		

Table 5.47: Wealth achieved by the active portfolio for using clusters.

Investing in JSE Intraday Data with Clusters for Absolute Portfolios

	Resources	**Industrials**	**Financials**
Stocks	AGLJ	MTNJ	SBKJ
	BILJ	NPNJn	FSRJ
	SOLJ	SABJ	OMLJ
	IMPJ	CFRJ	SLMJ
	ANGJ	REMJ	NEDJ
	GFIJ	SHFJ	
	AMSJ	BVTJ	
		SHPJ	
Best Agent	5.6655	2.6341	2.6059
Total Wealth	2.2093		
Running Time	632121		

Table 5.48: Wealth achieved by the absolute portfolio for using clusters.

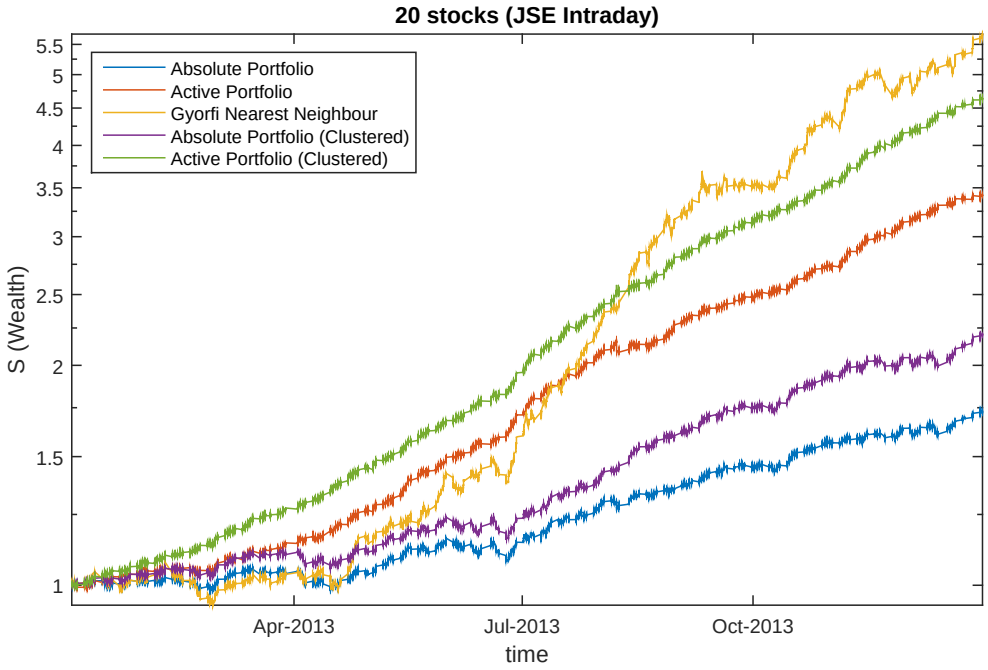


Figure 5.47: Wealth gained from 20 stocks when using clusters compared to the different strategies.

Chapter 6

Conclusion

6.1 Overall Summary

In this dissertation, an algorithm for portfolio selection that allows for long only portfolios (absolute portfolios) and zero cost portfolios (active portfolios) is presented. The results obtained from both the active and absolute portfolios are compared with strategies from the literature, namely the Universal Portfolios strategy by Thomas Cover [10] and the Nearest Neighbour method by Györfi *et al* [21]. It can be argued that the algorithm is analogous to Györfi's nearest neighbour algorithm when using the same choice of neighbours Györfi uses in [21], a long only portfolio and an optimisation step instead of the analytical approach. However it is important to note that the algorithm presented potentially allows for the algorithm to be implemented online, as well as the inclusion of the ability to cluster the stocks and partition the time series.

While clustering has been investigated, only the trivial form of the partitioning has been investigated. The results obtained from using clusters in section 5.6.8 show that clustering the stocks into resources, financials and industrials can be advantageous.

It can be concluded from the results that there are patterns that exist in both the New York Stock Exchange and the Johannesburg Stock Exchange and these patterns can potentially be exploited. The results of the algorithm for both the absolute and active portfolios compares favourably with the strategies from the literature and in most cases can beat the best stock. The results from the two stock combinations show that the wealth achieved from the absolute portfolio has a slight variance to the results achieved from Györfi's nearest neighbour strategy, this indicates that the quadratic approximation is highly accurate in comparison to the optimisation. This claim is further supported by the figures that show the wealth achieved from the different strategies for the two stock combinations. As the number of stocks are increased, it can be observed that the accuracy of the quadratic approximation decreases as the number of stocks increase. This is illustrated perfectly in Chapter 5, sections 5.5 and 5.6, for the JSE data as the number of stocks chosen increase from two stock combinations to 10, 20 and 30. However the loss of accuracy is the cost of achieving better computational performance.

Another observation can be made on the JSE daily data results in section 5.5 that the choice of price relative data set can have a significant impact on the wealth achieved. For instance the resources two stock combination (ANGJ and AGLJ) perform the best on the *open-to-close* data set, the financials two stock combination (SBKJ and FSRJ) performs well on the *open-to-open* and *close-to-close* data set and the industrials two stock combination (TBSJ and WHLJ) performs the best on the *open-to-open* data set. All of the two stock combinations and multiple stock combinations perform poorly on the *close-to-open* data set in comparison to the other price relative data sets, this could most likely be linked to the fluctuations in stock prices caused overnight between the closing price and opening price.

One of the criticisms of the results in this dissertation could be that trading costs, market impacts and stock liquidity are not taken into account. To address the concerns relating to stock liquidity, in majority of the stock combinations for the Johannesburg Stock Exchange only highly liquid stocks have been selected. However, it should be kept in mind that the aim of this dissertation is to confirm the existence of time-series patterns in the financial market data investigated and not to find a profitable trading strategy, even though the existence of a profitable trading strategy is a key measure of the existence of non-trivial patterns that have not yet been arbitraged out of the system.

6.2 Further Work

Possible areas of further work on this topic can include investigating different methods of partitioning the time series data as mentioned in Chapter 3, investigating the pattern matching aspect of portfolio selection and an investigation on the use of parallel computing to significantly decrease the computational running time of the agents.

Another extension would be to make the algorithm truly online [15, 40]. This can be achieved by creating a library of the matches offline, including an update rule, and having the algorithm search the library for the closest match at each time iteration.

In this dissertation, there is no consideration made to performance measures other than the wealth gained of the portfolios. There are however more effective measures of algorithm performance that are an important aspect of testing online algorithms, such as the approach suggested by Mohr and Schmidt [40] based on average competitive performance. These issues are not directly addressed in this dissertation, however it is an important consideration to take into account once the algorithm becomes truly online.

Bibliography

[1] Amit Agarwal, Elad Hazan, Satyen Kale, and Robert E Schapire. Algorithms for portfolio management based on the newton method. In *Proceedings of the 23rd international conference on Machine learning*, pages 9–16. ACM, 2006.

[2] Marco Aiolfi, C Favero, and Giorgio Primiceri. Recursive 'thick'modeling of excess returns and portfolio allocation. Technical report, IGIER Working Paper 197, IGIER, 2001.

[3] Marco Aiolfi and Carlo A Favero. Model uncertainty, thick modelling and the predictability of stock returns. 2003.

[4] Karhan Akcoglu, Petros Drineas, and Ming-Yang Kao. Fast universalization of investment strategies with provably good relative returns. In *International Colloquium on Automata, Languages, and Programming*, pages 888–900. Springer, 2002.

[5] Karhan Akcoglu, Petros Drineas, and Ming-Yang Kao. Fast universalization of investment strategies. *SIAM Journal on Computing*, 34(1):1–22, 2004.

[6] Paul H Algoet and Thomas M Cover. Asymptotic optimality and asymptotic equipartition properties of log-optimum investment. *The Annals of Probability*, pages 876–898, 1988.

[7] Avrim Blum and Adam Kalai. Universal portfolios with and without transaction costs. *Machine Learning*, 35(3):193–205, 1999.

[8] Allan Borodin, Ran El-Yaniv, and Vincent Gogan. Can we learn to beat the best stock. *J. Artif. Intell. Res.(JAIR)*, 21:579–594, 2004.

[9] Martin Casdagli, Stephen Eubank, J Doyne Farmer, and John Gibson. State space reconstruction in the presence of noise. *Physica D: Nonlinear Phenomena*, 51(1-3):52–98, 1991.

[10] Thomas M Cover. Universal portfolios. *Mathematical finance*, 1(1):1–29, 1991.

[11] Thomas M Cover and Erik Ordentlich. Universal portfolios with side information. *IEEE Transactions on Information Theory*, 42(2):348–363, 1996.

[12] Jason E Cross and Andrew R Barron. Efficient universal portfolios for past-dependent target classes. *Mathematical Finance*, 13(2):245–276, 2003.

[13] James P Crutchfield. Between order and chaos. *Nature Physics*, 8(1):17–24, 2012.

[14] Puja Das and Arindam Banerjee. Meta optimization and its application to portfolio selection. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1163–1171. ACM, 2011.

[15] Robert Dochow. *Online algorithms for the portfolio selection problem*. Springer, 2016.

[16] David Easley, Marcos M López de Prado, and Maureen O'Hara. The volume clock: Insights into the high-frequency paradigm. *The Journal of Portfolio Management*, 39(1):19–29, 2012.

[17] Alexei A Gaivoronski and Fabio Stella. Stochastic nonstationary optimization for finding universal portfolios. *Annals of Operations Research*, 100(1-4):165–188, 2000.

[18] Tim Gebbie and Fayyaaz Loonat. Learning zero-cost portfolio selection with pattern matching. *arXiv preprint arXiv:1605.04600*, 2016.

[19] G. Gelencsér and G. Ottucsák. Nyse data sets at the log-optimal portfolio homepage. `http://www.szit.bme.hu/~oti/portfolio`, 2006.

[20] László Györfi, Gábor Lugosi, and Frederic Udina. Nonparametric kernel-based sequential investment strategies. *Mathematical Finance*, 16(2):337–357, 2006.

[21] László Györfi, Frederic Udina, and Harro Walk. Experiments on universal portfolio selection using data from real markets, 2008.

[22] László Györfi, Frederic Udina, and Harro Walk. Nonparametric nearest neighbor based empirical portfolio selection strategies. *Statistics & Decisions International mathematical journal for stochastic methods and models*, 26(2):145–157, 2008.

[23] László Györfi, András Urbán, and István Vajda. Kernel-based semi-log-optimal empirical portfolio selection strategies. *International Journal of Theoretical and Applied Finance*, 10(03):505–516, 2007.

[24] Elad Hazan and Comandur Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 393–400. ACM, 2009.

[25] David P Helmbold, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8(4):325–347, 1998.

[26] Dieter Hendricks, Tim Gebbie, and Diane Wilcox. Detecting intraday financial market states using temporal clustering. *Quantitative Finance*, 16(11):1657–1678, 2016.

[27] Dieter Hendricks, Tim Gebbie, and Diane Wilcox. High-speed detection of emergent market clustering via an unsupervised parallel genetic algorithm. *South African Journal of Science*, 112(1-2):01–09, 2016.

[28] Robert Jarrow, Melvyn Teo, Yiu Kuen Tse, and Mitch Warachka. An improved test for statistical arbitrage. *Journal of Financial Markets*, 15(1):47–80, 2012.

[29] Narasimhan Jegadeesh. Evidence of predictable behavior of security returns. *The Journal of Finance*, 45(3):881–898, 1990.

[30] John L Kelly. A new interpretation of information rate. *Information Theory, IRE Transactions on*, 2(3):185–189, 1956.

[31] Wai Lee. *Theory and Methodology of Tactical Asset Allocation*, volume 65. John Wiley & Sons, 2000.

[32] Bin Li and Steven CH Hoi. On-line portfolio selection with moving average reversion. 2012.

[33] Bin Li and Steven CH Hoi. Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)*, 46(3):35, 2014.

[34] Bin Li, Steven CH Hoi, and Vivekanand Gopalkrishnan. Corn: Correlation-driven nonparametric learning approach for portfolio selection. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):21, 2011.

[35] Bin Li, Steven CH Hoi, Peilin Zhao, and Vivekanand Gopalkrishnan. Confidence weighted mean reversion strategy for online portfolio selection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(1):4, 2013.

[36] Bin Li, Peilin Zhao, Steven CH Hoi, and Vivekanand Gopalkrishnan. Pamr: Passive aggressive mean reversion strategy for portfolio selection. *Machine learning*, 87(2):221–258, 2012.

[37] Andrew W Lo, Harry Mamaysky, and Jiang Wang. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The journal of finance*, 55(4):1705–1770, 2000.

[38] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[39] Robert C Merton and Paul Anthony Samuelson. Continuous-time finance. 1990.

[40] Esther Mohr, Iftikhar Ahmad, and Günter Schmidt. Online algorithms for conversion problems: A survey. *Surveys in Operations Research and Management Science*, 19(2):87–104, 2014.

[41] Esther Mohr and Günter Schmidt. Empirical analysis of an online algorithm for multiple trading problems. In *Modelling, Computation and Optimization in Information Systems and Management Sciences*, pages 293–302. Springer, 2008.

[42] György Ottucsák and István Vajda. An asymptotic analysis of the mean-variance portfolio selection. *Statistics & Decisions*, 25(1/2007):63–86, 2007.

[43] Norman H Packard, James P Crutchfield, J Doyne Farmer, and Robert S Shaw. Geometry from a time series. *Physical review letters*, 45(9):712, 1980.

[44] Marcelo Perlin. Ms_regress-the matlab package for markov regime switching models. 2015.

[45] Andre F Perold and William F Sharpe. Dynamic strategies for asset allocation. *Financial Analysts Journal*, 44(1):16–27, 1988.

[46] M Hashem Pesaran and Allan Timmermann. Predictability of stock returns: Robustness and economic significance. *The Journal of Finance*, 50(4):1201–1228, 1995.

[47] Thomson Reuters. Thomson reuters tick history. https://tickhistory.thomsonreuters.com, 2015.

[48] Tim Sauer, James A Yorke, and Martin Casdagli. Embedology. *Journal of statistical Physics*, 65(3):579–616, 1991.

[49] Cosma Rohilla Shalizi and James P Crutchfield. Computational mechanics: Pattern and prediction, structure and simplicity. *Journal of statistical physics*, 104(3):817–879, 2001.

[50] Cosma Rohilla Shalizi, Kristina Lisa Shalizi, and James P Crutchfield. Pattern discovery in time series, part i: Theory, algorithm, analysis, and convergence. *Journal of Machine Learning Research*, pages 02–10, 2002.

[51] Peter Spellucci. Donlp2 short users guide. *Technische Universitt Darmstadt*, 1999.

[52] István Vajda. Analysis of semi-log-optimal investment strategies. In *Prague Stochastics*, pages 719–727, 2006.

[53] Volodimir G Vovk. Aggregating strategies. In *Proc. Third Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.

[54] Volodya Vovk and Chris Watkins. Universal portfolio selection. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 12–23. ACM, 1998.

[55] Diane Wilcox and Tim Gebbie. Serial correlation, periodicity and scaling of eigenmodes in an emerging market. *International journal of theoretical and applied finance*, 11(07):739–760, 2008.

[56] Diane Wilcox and Tim Gebbie. Hierarchical causality in financial economics. 2014.

[57] Ting-Kam Leonard Wong. Universal portfolios in stochastic portfolio theory. *arXiv preprint arXiv:1510.02808*, 2015.

Appendix A

MATLAB Code

A.1 Pattern Class

The algorithms have been implemented using a MATLAB class called pattern[1], as discussed in section 5.1.4. The first input parameter is the price relative data. Below is a guide on how to import and prepare the data for the pattern class.

```
%% Load data
stock1 = csvread('stock1.csv');
stock2 = csvread('stock2.csv');

% x is the return vector of the stocks
x = [stock1(:,2) stock2(:,2)];

% size of x
[m,n]=size(x);
% reshape x to [Stocks,Features,Times]
x = reshape(x',n,1,m);
```

The second input parameter is the k values and the third input parameter is the l values. The forth parameter defines any clustering of the stocks and and the fifth parameter states the portfolio strategy that should be used. See below for examples on how the method is run for the different portfolios.

1. Active Portfolio

   ```
   %% Active Porfolio

   p = pattern(x,1:5,1:10);
   ```

2. Absolute Portfolio

[1] The pattern code is available at https://github.com/FayyaazL/Pattern

```
%% Absolute Portfolio

p = pattern(x,1:5,1:10,[],'absolute');
```

3. Györfi's Nearest Neighbour

```
%% Gyorfi Nearest Neighbour

p = pattern(x,1:5,1:10,[],'gyorfi_opt');
```

The above examples do not take into account running the code with clusters, below is an example of how to do so.

```
%% Clustered stocks
resources = [1 1 0 1 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0];
industrials = [0 0 1 0 1 1 0 1 0 0 0 0 0 0 1 0 1 1 0 1];
financials = [0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1 0];
clusters = [resources; industrials; financials];
clusters = logical(clusters);

p = pattern(x,1:5,1:10,clusters);
```

The pattern class is shown below.

```
classdef pattern
    % PATTERN Pattern matching and learning class
    %
    % The class implements both online and offline pattern matching and
    % learning over k-tuples for M objects and N features as specified in
    % an MxNxP data matrix X. The algorithm searches for L nearest-neighbour
    % K-tuple matches in the provided partition of data. A qaudratic
    % approximation is used to find the log-optimal portfolio using T+1
    % expected return subsequent to the pattern matching times T for each
    % k and ell to find H(K,L,CI;T) and SH(K,L,CI;T) for each K-tuple
    % L value and cluster CI. The controls are then aggregate using machine
    % learning to provide the controls B and realised accumulated
    % performance S. If the K matching patterns are predefined then K is
    % the index over the matching patterns. The default is and empty
    % matching pattern and to use the last k-tuple as the matching pattern.
    %
    % See Also: PATTERN/MATCH, PATTERN/OFFLINE, PATTERN/LEARN, PATTERN/ONLINE, HFTS

    %
    % A. OHLC patterns
    % B. Fundamental model patterns
    % C. (side information partitioning)
    %
    % 1. *Data*
    %   2.1 M   (date-times)
    %   2.2 N  entities (objects e.g. stocks)
    %   2.3 P  features (OHLC)
    % 2. *Pattern* (k-tuple) [historic or user provided]
```

```
%   2.1 k-tuple (k free parameter)
%       2.2.1. k=1...K
%       2.2.2. k=1,...,N*K multiples of N (DSI)
%   2.2 k-tuple ConSet=[A,b] A*x>=b active/absolute (0,1)
%       2.2.1. active (many stocks,stock+cash)
%       2.2.2. absolute (single stock, long only portfolio)
% 3. *Segment (s) and Partition (p)*
%   3.1. [111...1] : full partition search for L nearest neighbours
%   3.2. [10..0],[010...0],...,[0...01] : L partitions for single best fit in each
%        partition
%   3.3. [001],[011],[111] for partitions weight to current time
%   3.4. ball radius (1-sigma ball relative to average distance)
% 4. *Distance*
%   4.1. surface norm
%   4.2. vector norm
% 5. *Predictors*
%   5.1. dependent (correlated) matching times j(n,ell)
%   5.2. independent matching times j(1...n,ell)
%   5.3. different matching times ???
%   5.4. [equi-probable] view is geometric average E[r]_t = exp(r_1+...+r_L)
% 6. *Agents*
%   6.1. Active/Absolute mean-variance
%   6.2. Canonical Agents:
%       6.2.1. controls H (NxMxT) M objects, N agents, T times)
%       6.2.2. performance SH (arithmetic) (TxN for T times and N agents)
%       6.2.3. horizon parameter (t to include delay)
%   6.3. allow mixture of long-only with cash neutral.
% 7. *Algorithm*
%   7.1. Online/Offline
%   7.2. Parallel computing
%   7.3. Interface agents (h,SH)
% 8. *Machine Learning* (EG,EW,EWMA,UNIV + abs/act + part/comb)
%   8.1. Weighted Arithmetic Average over all agents to create optimal predictors
%       8.1.1. Performance weighted averaging (using arithmetic averaging)
%           B(T) = SUM(K,L) (SH(T-1|K,L) H(T|K,L)) / SUM(K,L) SH(T-1|K,L)
%           this is probability wieghted where the probability is
%           proportional to the returns.
%       8.1.2. Exponentially weight more recent performance data
%   8.2. parameters: window W_L, forgetting factor Lamba_L
%   8.3. either fully invested or active.
% 9. *Sectors and States* (clusters)
%   9.1. CI PxN for P clusters and N objects

% Authors: Tim Gebbie, Fayyaaz Loonat (2016,2017)

%% public properties
properties
    b = []; % aggregated controls    (T,M) Time x Objects
    S = []; % aggregated performance (T,1) Time x 1
    h = []; % agent controls      (N,M,T)  Agents x Objects x Time
    SH = []; % agents performance (T,N)    Time x Agents
end
%% private properties
properties (Access = private)
    x = []; % data as price relatives (M,F,T) Objects x Features x Time
    k = []; % k-tuple size
    ell = []; % partition size
```

```
    p = []; % current partition
    s = []; % segment specification
    ci = []; % clusters  (C,M) Cluster x Objects
    mnp = []; % dimensionality (M,F,T)
    ntype = 'active'; % agent normalisation (estimation)
    lntype = 'absolute'; % agent relative (learning) normalisation
    ltype = 'univlearning';
    lparam = []; % learning parameters
    ptype = 'trivial';
    qH = []; % agents controls (T,N) Time x Agents
    xnk = {}; % predefined patterns
    horz = 1;
end
%% methods
methods
    %% constructor
    function p = pattern(varargin)
        % P = PATTERN/PATTERN constructor
        %
        % P = PATTERN(X,K,ELL,CI,NTYPE,LNTYPE,LTYPE) For data X (price relatives)
        %   and is MxNxP dimensional data for M objects, N features and
        %   P date-times. dim(X) is dim(Price)-1. A typical object is
        %   a stocks, e.g. AGL, a typical feature is a factors such as
        %   OPEN, HIGH, LOW, CLOSE (OHLC), and date-times are the time
        %   stamps for the data.
        %
        % Example 1:
        % >> x = 1+0.2*randn(10,1,1000);
        % >> p = offline(p);
        % >> p.S
        %
        %  P = PATTERN(X,XNK,ELL,CI,NTYPE,LNTYPE,LTYPE) For matching
        %   pattern XNK instead of K-tuple range.
        %
        %  P = PATTERN(X,XNK,ELL,CI,NTYPE,LNTYPE,LTYPE,TREND)
        %   LONGTERM is TRUE to use long-term relative views. This is
        %   by default false.
        %
        % The data X is homogenize in time. K is the set of tuple
        % sizes if there is no matching pattern e.g. [1:3], if there
        % is a matching pattern it is the number of matching patterns.
        % The number of matching times is ELL this is typically 10
        % per partition. The cluster definitions is CI, this is by
        % default the trivial cluster (all the objects) when CI is
        % empty. CI is KxM for K clusters of M objects. NTYPE is the
        % strategy normalisation and can be either 'active', 'absolute'
        % or 'gyorfi_opt'. LNTYPE is the agent normalisations and can
        % be either 'active' or 'absolute'.
        % The LTYPE is the learning type this is by default 'univ'.
        %
        % Note: X are price relatives (P(t)/P(t-1)). These can be
        % conveniently computed using EXP(DIFF(LOG(P))).
        %
        % See Also PATTERN/DISPLAY, PATTERN/SUBSREF

        if nargin==0
            % null constructor
```

```
else
    %% Compute price relatives as returns
    x0 = varargin{1}; % (dim = n-1);
    % size of the data (M=OBJECTS, N=FEATURES, P=DATATIMES)
    mnp0 = size(x0);
    % catch the case of M,N,1 (data sliced at a single time)
    if length(mnp0)==2
        mnp0(1:2) = mnp0;
        mnp0(3) = 1;
    end
    xnk0 = varargin{2};
    % trivial cluster
    ci0 = true(1,mnp0(1));
    % set defaults for optional inputs
    optargs = {x0 xnk0 10 ci0 'active' ...
        'absolute' 'univlearning' 'trivial' false};
    % now put these defaults into the valuesToUse cell array,
    optargs(1:nargin) = varargin;
    % number of k's
    p.k=size(xnk0);
    % class test
    switch class(xnk0)
        case 'cell'
            % Case 1: multiple (k0) user defined patterns
            % dimensions of the possible matching pattern input
            xnlmnp = size(xnk0{1});
        case 'double'
            % Case 2: single user defined pattern or k-tuples
            % dimensions of the possible matching pattern input
            xnlmnp = size(xnk0(1));
    end
    % check for either tuple or pattern
    if (xnlmnp(1)==mnp0(1)) && (xnlmnp(2)==mnp0(2))
        % user defined patterns
        [p.x, p.xnk, p.ell, p.ci, p.ntype, p.lntype, p.ltype, p.ptype]...
            = optargs{:};
    else
        % Place optional args in memorable variable names
        [p.x, p.k, p.ell, p.ci, p.ntype, p.lntype, p.ltype, p.ptype]...
            = optargs{:};
    end
    switch p.ltype
        case 'eglearning'
            p.lparam = 0.01;
        case 'univlearning'
            p.lparam = Inf;
        case 'ewlearning'
            p.lparam = 0.99;
    end
    % ensure the inputs are feasible
    if isempty(p.ci) || strcmp(p.ci,':'), p.ci = ci0; end;
    % update the size property
    p.mnp = mnp0;
    % control parameters
    L = size(p.ell,2); % partitions (param 1)
    W = size(p.ci,1); % cluster
    K = size(p.k,2); % tuples (param 2)
```

```
        % initialise state variables and pre-allocate memory
        p.b = nan(p.mnp(3)+1,p.mnp(1)); % Time x Objects
        p.h = nan(L*W*K,p.mnp(1),p.mnp(3)+1); % Agents x Objects x Time
        p.qH = nan(p.mnp(3)+1,L*W*K); % Time x Agents
        p.S = ones(p.mnp(3)+1,1); % Time x 1
        p.SH = ones(p.mnp(3)+1,L*K*W); % Time x  Agents
        %% partition the data
        p=partition(p);
    end
end
%% offline (to force offline estimation)
function p=offline(p)
    % PATTERN/OFFLINE Offline estimation
    %
    % P = OFFLINE(P) to estimate (H,SH) for T=K*L:T0 for K and L.
    %
    % See Also: PATTERN/ONLINE

    tmin0 = 3*max(p.ell);
    t0 = p.mnp(3);
    if t0<tmin0
        error('pattern:offline','Not enough Data L*K>T');
    end
    % Find matching times j for pattern [offline loop]
    for t=tmin0:t0 % time loop
        p = online(p,t);
    end

end
%% online (to force online estimation)
function p=online(varargin)
    % PATTERN/ONLINE Online estimation
    %
    % P = ONLINE(P) to estimate (H,SH,B,S) at T for the range of
    %   K and L over the specified clusters CI. This requires the
    %   online structure for learning to have been initialised. T
    %   is taken to be the last time in the object.
    %
    % P = ONLINE(P,T) to estimate online values at time T using the
    %   data from times 1 to T (1:T).
    %
    % See Also: PATTERN/ONLINE, PATTERN/MATCH, PATTERN/MATCH

    %% input parameters
    p = varargin{1}; % (dim = n-1);
    % set defaults for optional inputs
    optargs = {p p.mnp(3) {}};
    % now put these defaults into the valuesToUse cell array,
    optargs(1:nargin) = varargin;
    % Place optional args in memorable variable names
    [p, t, s0] = optargs{:};
    %% control parameters
    L = size(p.ell,2); % partitions (param 1)
    W = size(p.ci,1); % cluster
    K = size(p.k,2); % tuples (param 2)
    %% Model Identification Loop
    % initial controls
```

```
% t+1 rows as the last row is for an unrealised return
ht = nan(W*K*L,p.mnp(1)); % agents controls
xt = p.x(:,:,1:t);
% exit if there is not enough data
if 3*max(p.ell)>t
    error('pattern:online','Not enough data K*L>T');
end
if (t>p.mnp(3))
    error('pattern:online','Not enough data T> dim(X,T)');
end
% partition the data
p = partition(p,t);
% partition
p0 = p.p;
% matching loop
for w0=1:W % clusters (groups)
    ci0=p.ci(w0,:);
    % the cluster referenced data block with fixed partition
    xtw0 = xt(ci0,:,:);
    for ell0=1:L, % parameter 1 -> passed to match (ell neighbours)
        ell1 = p.ell(ell0);
        for k0=1:K  % parameter 2 -> passed to match (k-tuple)
            % expert index KLrow(w0,k0,ell0)
            KLrow = sub2ind([W*K,L],K*w0-k0+1,ell0);
            % (k,ell)-agents matched pattern for cluster ci(w0)
            % --- 1. select the pattern -------------------------
            if isempty(p.xnk)
                % k index tuple size
                xnk0 = xtw0(:,:,end-p.k(k0)+1:end)-1; % r= R - 1
            else
                % k index of the matching pattern
                xnk0 = p.xnk{p.k(k0)};
            end
            % --- 2. pattern matching ---------------------------
            [hklt,s0]=match(xtw0,p0,xnk0,ell1,p.ntype,p.horz,s0,L); % online
            % -------------------------------------------------
            % expert controls per cluster mapping
            ht(KLrow,ci0) = transpose(hklt);
        end % k
    end % ell
end % w
% initialise h,SH if it is the trivial object
% compute the update performance for the prior agent step
if any(isnan(p.h(:,:,t)))
    dSH = ones(size(p.SH(t,:)));
else
    dSH = transpose((p.h(:,:,t) * (p.x(:,1,t)-1))+1); % was exp
end
% remove NaN
ht(isnan(ht))=0;
% update the agents
p.h(:,:,t+1) = ht;
% update the agent accumulate performance (geometric returns)
p.SH(t+1,:) =  p.SH(t,:) .* dSH;
% update options
p.s = s0;
%% online update the learning
```

```
    p = learn(p,t);
end
%% learning
function p=learn(varargin)
    % PATTERN/LEARN Machine Learning based on performance
    %
    % P = LEARN(P) The updates the aggregated agents and agent
    %   performance (B,S) using the specified learning type.
    %
    % P = LEARN(P,TYPE) will reset the learning
    %
    % Table 1: Learning types
    % +----------------+---------------------------------------+
    % | TYPE           | Description                           |
    % +----------------+---------------------------------------+
    % | 'univlearning' | Universal learning (log-optimal)      |
    % |                | [PARAM=[NONE]                         |
    % | 'eglearning'   | Exponentiated Gradient                |
    % |                | [PARAM=ETA in [0,1] typ. [0,0.2]      |
    % | 'ewlearning'   | EWMA in control based learning        |
    % |                | [PARAM=LAMBDA in [0,1] typ. [0.9,0.99]|
    % +----------------+---------------------------------------+
    %
    % References:
    % [1] Cover, T., M. (1991) Universal Portfolios
    % [2] Gyorfi, L., Udina, F., Walk, H., (2008) Experiments on universal
    %           portfolio selection using data from real markets
    % [3] Cover, T. M., (1996), Universal Portfolios with Side Information
    % [4] Algoet, P. H., Cover, T. M., (1980) Asymptotic optimality and
    %           symptotic equipartition properties of log-optimum investments
    % [5] Helmbold, D., P., Schapire, R., E., Singer, Y., Warmuth, M.,
    %           K.,(1998) On-line portfolio selection using multiplicative updates
    %
    % See Also: PATTERN/MATCH

    % SHX = exp(diff(log(SH)) for price path SH!

    %% input parameters
    p = varargin{1}; % (dim(x) = t) price t+1;
    % set defaults for optional inputs
    optargs = {p p.mnp(3)};
    % now put these defaults into the valuesToUse cell array,
    optargs(1:nargin) = varargin;
    % Place optional args in memorable variable names
    [p, t] = optargs{:};

    %% Machine Learning
    if (size(p.h,1)==1)
        % only single agent
        % ---- ONLINE update ----
        b0 = p.h(1,:,t+1);
        qH0 = 1;
        % -----------------------
    else
        % multiple agents
        switch p.ltype
            case 'univlearning'
```

```
        % compute weights
        qH0 = p.SH(t+1,:);
    case 'eglearning'
        eta = p.lparam; % learning parameters
        % update the weights using the exponentiated gradient
        % algorithm
        if all(p.qH(t,:)==0) || all(isnan(p.qH(t,:)))
            qH0 = p.SH(t+1,:); % initialise
        else
            qH0 = p.qH(t,:) .* exp(eta .* ...
                p.SH(t+1,:) ./ (p.qH(t,:) * p.SH(t+1,:)'));
        end
    case 'ewlearning'
        lambda = p.lparam; % learning parameters
        % update term
        if all(p.qH(t,:)==0) || all(isnan(p.qH(t,:)))
            qH0 = p.SH(t+1,:);
        else
            Z = (p.qH(t,:) .* p.SH(t+1,:))./ (p.qH(t,:) * p.SH(t+1,:)');
            % update the weights using the exponentiated gradient algorithm
            qH0 = lambda * p.qH(t,:) + (1-lambda) * Z;
        end
end
%% renormalise the weights
switch p.lntype
    case 'active'
        qH0 = (qH0 - mean(qH0));
        norm0 = sum(abs(qH0));
        if norm0>eps
            qH0 = qH0 ./ norm0;
        else
            qH0 = zeros(size(qH0));
        end
    case 'absolute'
        qH0 = qH0 ./ sum(qH0);
end
%% create the performance weighted combination of experts.
% ONLINE
% -------------------------------------------------------
b0 = qH0 * p.h(:,:,t+1);
% -------------------------------------------------------
%% compute normalization abs(long) + abs(short)
tb = nansum(abs(b0));
% renormalize controls (leverage=1) [FIXME LEV]
if tb==1
elseif tb>eps
    b0 = (1/tb) * b0;
    % update the agent mixture weights for leverage
    qH0 = (1/tb) * qH0;
else
    switch p.lntype
        case 'absolute'
            % update the agent mixture weights for leverage
            qH0 = zeros(size(qH0)); % FIXME (should be equally weighted)
            % zero weights
            b0 = zeros(size(b0));
    end
```

```
        end
    end % only 1 agent
    %% compute the leverage corrected output price relative
    % compute the updated returns
    dSH = p.SH(t+1,:)./p.SH(t,:);
    % uses the inputed online structure reference.
    if all(isnan(p.qH(t,:)))
        dS = 1;
    else
        dS = ((dSH-1) * transpose(p.qH(t,:)))+1; %  LINRET was exp
    end
    % update the properties
    p.qH(t+1,:) = qH0;
    p.b(t+1,:) = b0;
    p.S(t+1,:) = p.S(t,:) * dS;
end
%% partition
function p=partition(varargin)
    % PATTERN/PARTITION Partition the data
    %
    % P = PARTITION(P) single partition: [111...1] for the
    %   partition type 'trivial'. NP=ELL as the number of partitions.
    %   The type of partition as TYPE.
    %
    % Table 1: Partition Types
    % +-------------+-----------------------------------------+
    % | TYPE        | Description                             |
    % +-------------+-----------------------------------------+
    % | 'trivial'   | [111...1] (single partition)            |
    % | 'overlap'   | [0...001][0...011][0...111]...[1...111] |
    % | 'exclusive' | [100...0][010...0]...[000...1]          |
    % | 'sideinfo'  | partition using side-information        |
    % +-------------+-----------------------------------------+
    %
    % P = PARTITION(P,T) Relative to time T.
    %
    % Note 1: Side Information base learning will partition the data based
    % on the state of the side information. The required state will
    % then be used to determine which partition to use at time T
    % for the model estimation and learning.
    %
    % Note 2: Removes all days with the same returns as these are
    %   considered incorrect data days.
    %
    % See Also

    p = varargin{1}; % (dim = n-1);
    % set defaults for optional inputs
    optargs = {p p.mnp(3)};
    % now put these defaults into the valuesToUse cell array,
    optargs(1:nargin) = varargin;
    % Place optional args in memorable variable names
    [p, t] = optargs{:};
    % create the partitions
    switch p.ptype
        case 'trivial'
            % trivial (default) partition [111...1]
```

```
            p.p=true(1,t);
        case 'exclusive'
            if any(size(p.ell)>1) && (floor(t/p.ell)<1),
                error('pattern:partition',...
                    'Only single ELL allowed for type EXCLUSIVE');
            end
            % ELL exclusive partitions
            p.p=false(p.ell,t);
            % number of partitions
            ni = 0:floor(t/p.ell):t;
            if ((t-ni(end))>0)
                ni = ni(1:end-1); % drop the last partition
                ni = [ni t]; % extend the last partition
            end
            for j=1:p.ell
                p.p(j,ni(j)+1:ni(j+1))=true;
            end
        case 'overlap'
            % ELL overlapping partitions
            if any(size(p.ell)>1) && (floor(t/p.ell)<1),
                error('pattern:partition',...
                    'Only single ELL allowed for type EXCLUSIVE');
            end
            % ELL exclusive partitions
            p.p=false(p.ell,t);
            % number of partitions
            ni = 0:floor(t/p.ell):t;
            for j=1:p.ell
                p.p(j,ni(j)+1:t)=true;
            end
        case 'sideinfo'
            % ELL references the side-information factor
            %
            % 1. use one of the factors to partition based on
            %   side-information.
            % 2. The side-information factor will be excluded
            %   from the nearest-neighbour calculation
            error('pattern:partition','Side Information unsupported');
    end
end
%% sanity check data
function p = sanitycheck(p)
    % SANITYCHECK Remove insane data days (all zero, nan, 1 @T)
end
function p = commutecheck(p)
    % COMMUTECHECK Check that the commutation of controls holds
    %
    % Figure 1: Commutation of the weight b, and agents H using
    %           agents weights q and the returns of the stocks r
    % +-----+         +--------+      +-----------------------+
    % |(r,H)|   q ->  | b= q H | N -> | tilde b (1/lambda) b  |
    % +-----+         +--------+      +-----------------------+
    %    |r               | r                       |r
    % +--------+     +-----------+      +----------------------+
    % |SH = H r|q -> | S = q' SH | N -> | S = (1/lambda) q' SH |
    % +--------+     | S = b r   |      | S = tilde b r        |
    %                +-----------+      +----------------------+
```

```
    %
    % See Also:

    % r,H -> b = q H -> S = b r

    % S = H r <-> S = br
end
%% subscript assignment
function p = subsasgn(p,s,b)
    % PATTERN/SUBSASGN Subscript assigment
    %
    % P.<VarName> = Value
    %
    % See Also: PATTERN/SUBSREF

    varName = s(1).subs;
    switch varName
        case {'learntype'}
            p.ltype = b;
            if ~isempty(p.SH) && ~all(all(p.SH==1))
                switch p.ltype
                    case 'univlearning'
                        p.lparam = Inf;
                    case 'eglearning'
                        p.lparam = 0.01;
                    case 'ewlearning'
                        p.lparam = 0.99;
                    otherwise
                        error('pattern:subsasgn:learntype',...
                            'Unrecognized Learning');
                end
                % update the learning
                tmin0 = 3*max(p.ell);
                t0 = p.mnp(3);
                if t0<tmin0
                    error('pattern:subsasgn:learntype','Not enough Data L*K>T');
                end
                % Find matching times j for pattern [offline loop]
                for t=tmin0:t0 % time loop
                    p = learn(p,t);
                end
            end
        case {'learnparam'}
            p.lparam = b;
            if ~isempty(p.SH) && ~all(all(p.SH==1))
                % update the learning
                tmin0 = 3*max(p.ell);
                t0 = p.mnp(3);
                if t0<tmin0
                    error('pattern:subsasgn:learnparam','Not enough Data L*K>T');
                end
                % Find matching times j for pattern [offline loop]
                for t=tmin0:t0 % time loop
                    p = learn(p,t);
                end
            end
        case {'learnnorm'}
```

```
            p.lntype = b;
            if ~isempty(p.SH) && ~all(all(p.SH==1))
                % update the learning
                tmin0 = 3*max(p.ell);
                t0 = p.mnp(3);
                if t0<tmin0
                    error('pattern:subsasgn:learnparam','Not enough Data L*K>T');
                end
                % Find matching times j for pattern [offline loop]
                for t=tmin0:t0 % time loop
                    p = learn(p,t);
                end
            end
        otherwise
            error('pattern:subsref','Incorrect reference');
    end
end
%% subscript reference by time
function p = subsref(p,s)
    % PATTERN/SUBSREF Sybscript reference
    %
    % P = P(T) subscript reference out the pattern object over time
    %   range T.
    %
    % P.<VarName> to reference out the required properties such as
    %   H the agent controls, SH the accumulated geometric agent
    %   performance, B the aggregated controls, and the aggregated
    %   agent performance.
    %
    % Examples: p.h(:,:,end)
    %
    % See Also: PATTERN/SUBSASGN

    switch s(1).type
        case '()'
            if size(s.subs,2)==1
                % subscript reference object by time
                % (M,N,P) -> (:,:,P0)
                t0=s(1).subs{1};
                p.x = p.x(:,:,t0);
                p.mnp = size(p.x);
                % retain the internal state history
                t1 = [t0 t0(end)+1]; % for online functionality
                p.h = p.h(:,:,t1); %
                p.SH = p.SH(t1,:);
                p.b = p.b(t1,:);
                p.S = p.S(t1);
                p.qH = p.qH(t1,:);
                % re-partition
                p = partition(p);
            elseif size(s.subs,2)==2
                % subscript reference object by time and clusters
                % (M,N,P) -> (M0,:,P0)
                error('pattern:subsref','CI subsref not supported');
            end
        case '.'
            % A reference to a variable or a property.  Could be any sort of
```

```
                    % subscript following that.  Row names for () and {} subscripting
                    % on variables are inherited from the dataset.
                    varName = s(1).subs;
                    switch varName
                        case {'h','SH','S','b'}
                            p = p.(varName);
                            if size(s,2)==2
                                p=subsref(p,s(2));
                            end
                        case {'ci'}
                            p = p.ci;
                        case {'cin'}
                            % number of clusters
                            p = size(p.ci,1);
                        otherwise
                            error('pattern:subsref','Incorrect reference');
                    end
                otherwise
                    error('pattern:subsref','Incorrect reference');
            end
        end
        function display(p)
            % PATTERN/DISPLAY Display the Pattern object
            %
            % See Also PATTERN

            disp(p);
            if ~isempty(p.x)
                fprintf('\tParameters\n');
                fprintf('\t----------------------\n');
                fprintf('\tk-tuples       : %s\n',num2str(p.k));
                fprintf('\tell neighbours : %s\n',num2str(p.ell));
                fprintf('\t#clusters      : %d\n',size(p.ci,1));
                fprintf('\tlearn param.   : %3.2f\n',p.lparam);
                fprintf('\thorizon        : %d\n',p.horz);
                fprintf('\n');
                fprintf('\tData x(M,N,P)\n');
                fprintf('\t----------------------\n');
                fprintf('\tobjects (N) : %d\n',p.mnp(1));
                fprintf('\tfactors (M) : %d\n',p.mnp(2));
                fprintf('\tstates  (P) : %d\n',p.mnp(3));
                fprintf('\tpartitions  : %d\n',size(p.p,1));
                fprintf('\n');
                fprintf('\tAlgorithm Constraints\n');
                fprintf('\t----------------------\n');
                fprintf('\tmatching cons. : %s\n',p.ntype);
                fprintf('\tlearning       : %s\n',p.ltype);
                fprintf('\tlearning cons. : %s\n',p.lntype);
                fprintf('\tpartition      : %s\n',p.ptype);
            end
        end
    end % end methods
end
%% helper functions
function [w,s]=quadbet(varargin)
% QUADBET Quadratic optimal bet
%
```

```
% W = QUADBET(f,H,gamma) Solve the optimal tactical bet. The expected
%   view F and covariance matrix H are required as is the risk aversion
%   GAMMA.
%
% W(1) is the benchmark bet (fully invested minimum variance
%   portfplio).
% W(2) is the tactical bet based on the long-term eqiulibrium
%   view F. This is by default zero.
%
% use speudo inverse pinv(omega) is possible is conditioning is an issue

%% initialise the inputs
f = varargin{1};
H = varargin{2};
s = [];
% initialise inputs
optargs = {f H 1 s};
% now put these defaults into the valuesToUse cell array,
optargs(1:nargin) = varargin;
% Place optional args in memorable variable names
[f, H, gamma, s] = optargs{:};

%% compute the optimal portfolio
% get the size of the covariance matrix
[m,n] = size(H);
% diagonals of one
I = ones(m,1);
% invert the covariance matrix
invHI = H\I;
invH = inv(H);
% benchmark weights
w(:,1) = (I' * invHI) \ (invHI);
% active weights
w(:,2) = (1/gamma) * (I' * invHI) \ (invH * ( f(:) * I' - I * f(:)') * invH ) * I;

end
function [hkl,s] = match(varargin)
% PATTERN/MATCH Pattern Match for a given K-tuple and matching data set.
%
% [HKL]=MATCH(X0,P0,XNK,L0,NTYPE,TREND,HORZ) X0 are the factor relatives. P0 the
%   partition. K0 is the k-tuple size over the current data set, it is
%   computed from the user definied pattern (k-tupel) XNK. XNK is a MxNxK0
%   double for X0 a MxNxP size double for T>>K0. L0 is the number of
%   neighbours to include from the partitioning P0 of the data. NTYPE is
%   the normalisation type. This excludes the time loop over the the
%   matching horizon. This is the online version of the pattern matching
%   and learning algorithm. HKL is a Mx1 control vector that satifies the
%   normalisation type NTYPE.
%
% See Also: PATTERN/ONLINE, PATTERN/OFFLINE, QUADBET

% Author: Tim Gebbie

%% initialise the input variables
x0 = varargin{1};
p0 = varargin{2};
xnk = varargin{3};
```

```
optargs = {x0,p0,xnk,20,'active',1,{}};
% now put these defaults into the valuesToUse cell array,
optargs(1:nargin) = varargin;
% Place optional args in memorable variable names
[x0,p0,xnk,ell0,ntype,horz,s,L] = optargs{:};

%% Loop parameters wrt to partition
q0 = size(p0,1); % number of partitions
m0 = size(x0,1); % number of objects

%% Portfolio agent constraints
switch ntype
    case 'absolute'
        hkl = (1/m0)*ones(1,m0);
    case 'active'
        hkl = zeros(1,m0);
    case 'gyorfi_opt'
        hkl = (1/m0)*ones(1,m0);
end
% the pattern as returns computed from price relatives
k0 = size(xnk,3);
% check consistency
if any(size(k0)>1), error('portchoice:pattern:match','Incorrect K'); end
if any(size(ell0)>1), error('portchoice:pattern:match','Incorrect L'); end

%% Find matching times j for pattern
for q=q0 % maximum of ell partitions [P(ell)][temporal]
    % the partition
    psi = p0(q,:);
    % for agent h(k,ell) for cluster ci, and partition psi
    snk = x0(:,:,psi)-1;
    % find the times (this allows for inhomogenous partitions)
    jell = find(psi);
    % reset the distance measure for partition
    ed = Inf(size(snk,3),m0);
    %% get the test tuples by looping over partition snk
    for j=k0:size(snk,3) - horz % only allow jell+1 in partition
        % distance (element-wise difference)
        edi = xnk - snk(:,:,j-k0+1:j);

        if (k0==1)

            ed(j,1:m0) = norm(edi);
        else
            % reshape the distance by objects and factors
            edi = reshape(edi,size(edi,1),size(edi,2)*size(edi,3));
            ed(j,1:m0) = norm(edi);
        end
    end % j segment loop
    %% sort the matching times

    pl = 0.02 + 0.5*((ell0-1)/(L-1));
    pell0 = floor(pl*j);


    if (q0==1)
        % ell matches in a single partition
```

```
        [~,nj]=sort(ed,'ascend'); % ~ -> pat.ed not required

        % Take into account ties of the norm
        if pell0>0
            while pell0<length(nj) && (ed(nj(pell0))==ed(nj(pell0+1)))
                pell0 = pell0 + 1;
            end
        end

        % update the matching times
        njell = jell(nj(1:pell0,:));
        % update the norm
        ed = ed(nj(1:pell0,:));

        % update the distances
        % pat.ed = pat.ed(nj(pat.ell0,:));
        % find the ell matching times
        % --- LOOK AHEAD RULE ------------
        jnk = njell+horz; % 1-step look ahead
        % --------------------------------
    else
        % find the single closest match in the ell partitions
        [~,nj]=min(ed); % ~ -> pat.ed not required
        % find the matching time
        njell = jell(nj);
        % the uncertainty as the norm
        ed(q) = ed(nj);
        % update the matching times
        % --- LOOK AHEAD RULE ---------------
        jnk(q) = njell+horz; % 1-step look ahead
        % -----------------------------------
    end
end % end partition loop
%% find the predictions
E = eye(m0,m0);
hatx = zeros(pell0,m0);
% initialise prediction vector
for mj = 1:m0 % loop over objects
    % select first factor (price relative) @JNK matching times
    hatx(:,mj) = x0(mj,1,jnk(:,mj))-1; % r = R - 1
    % matching error (diag covariance matrix)
    E(mj,mj) = mean(ed(:,mj).^2);
end % loop over objects/stocks
%% find the agents using mean-variance approximation

% --------------------------------
f = mean(hatx,1); % expected view : r = R-1

% --------------------------------
% E = (1/sum(diag(E)))*E; % relative distance error
% T = mean(range(hatx,2)) * eye(size(E)); % entropy
% construct the experts for the k,ell choice
if size(hatx,1) > 2
    H = cov(hatx); % r = R-1
    if any(diag(H)==0)
        % remove zero covariance
        zi = (diag(H)==0);
```

```
        H(zi,zi) = nanmean(diag(H))*eye(sum(zi));
    elseif all(H==0)
        % all are zero use diagonal matrix
        H=eye(m0);
    end
else
    H = eye(m0);
end
if rcond(H)<1e-2,
    % warning('portchoice:newbcrp','Bad H conditioning using diag(H)');
    H = diag(diag(H));
    if rcond(H)<1e-2,
        H=eye(m0);
    end
end;
% combine intrinsic uncertainty with estimation uncertainty
% ------------------------------
% H = H + E + T; % intrinsic + estimation + entropy
% H = H + E; % intrinsic + estimation
% H = H; % intrinsic uncertainty
% ------------------------------
% approximate log optimal k,l-th expert using quadratic approx.
% ----------------------------
[hkl0,s] = quadbet(f,H);
% ----------------------------
switch ntype
    case 'absolute'
        % fully invested
        hkl = sum(hkl0,2);
        if sum(abs(hkl))>1+eps
            % find the biggest short-sold asset
            [~,i0] = min(hkl);
            % aggressiveness factor
            n0 = abs(hkl0(i0,1) / hkl0(i0,end));
            % rescale the tactical portfolio
            hkl = hkl0(:,1) + n0 * hkl0(:,end);
        end
    case 'active'
        % cash neutral
        hkl = hkl0(:,end);
        % leverage unity
        hkl = hkl/sum(abs(hkl));
    case 'gyorfi_opt'
        optfun = inline('-log(prod(x*transpose(b)))','b','x');
        xx = hatx +1;
        [~, m] = size(xx);
        b0 = (1/m)*ones(1,m);
        [hkl2, ~] = fmincon(@(b) optfun(b,xx),b0,[-eye(m);eye(m)],...
            [zeros(1,m)';ones(1,m)'],ones(1,m),1,[],[],[],...
            optimset('Algorithm','sqp','Display','off'));
        hkl = hkl2';
end
end
```

A.2 Universal Portfolios

The following MATLAB function was used to recover the results of Cover's Universal Portfolio [10] algorithm[2]. The function takes in the price relative data and returns the portfolio wealth, the wealth of the stocks and the fraction allocated to the first stock.

```
function [universalwealth, stock1wealth, stock2wealth, bm] = cover_up_rec(x)
% COVER_UP_REC - Implementation of Thomas M. Cover's Universal Portfolio
% for two stocks
%
% Syntax: [universalwealth, stock1wealth, stock2wealth, bm] = cover_up_rec(x)
%
% Inputs:
%    x - Price Relative Data of the two stocks
%
% Outputs:
%    universalwealth - Wealth acheived by the portfolio
%    stock1wealth - Wealth acheived by stock 1
%    stock2wealth - Wealth acheived by stock 2
%    bm - Fraction of wealth allocated to stock 1

%% STEP 1: Initialise Variables

[n, ~] = size(x);
stock1wealth=zeros(n,1);
stock2wealth=zeros(n,1);
universalwealth=zeros(n,1);
bm = zeros(n,1);
bm(1) = 0.5;

% Sn is the wealth accumulated from the constant rebalanced portfolios
Sn = zeros(n,21);

%% STEP 2 Update the Portfolio
for i = 1:n
    % Update the wealth of stock 1 and stock 2
    stock1wealth(i) = prod(x(1:i,1));
    stock2wealth(i) = prod(x(1:i,2));

    % Update the wealth achieved by the universal portfolio
    Sn(i,:) = (0:0.05:1)*x(i,1) +(1:-0.05:0)*x(i,2);
    universalwealth(i) = (sum(prod(Sn(1:i,:))))/21;

    % Update wealth assigned to stock 1
    if i >1
        b = sum((0:0.05:1)*(prod(Sn(1:i,:))'))/(sum(prod(Sn(1:i,:))));
        bm(i) = b;
    end
end
end
```

The function is called up as follows:

[2]The code is available at https://github.com/FayyaazL/Pattern

```
%% Load data
stock1 = csvread('stock1.csv');
stock2 = csvread('stock2.csv');

% x is the return vector of the stocks
x = [stock1(:,2) stock2(:,2)];

[U_S, S1, S2, b] = cover_up_rec(x);
```