Universidade de Aveiro Departamento de Eletrónica, Telecomunicações e Informática
2017

Miguel Marreiros Inácio de Campos

Plataforma para Negociação FOREX

Universidade de Aveiro Departamento de Eletrónica,
Telecomunicações e Informática
2017

Miguel Marreiros
Inácio de Campos

Plataforma para Negociação FOREX

“”

Universidade de Aveiro
2017

Departamento de Eletrónica,
Telecomunicações e Informática

Miguel Marreiros
Inácio de Campos

Plataforma para Negociação FOREX

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Diogo Nuno Pereira Gomes, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Luis Seabra Lopes, Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais, por todo o apoio que me deram ao longe deste percurso.

agradecimentos /
acknowledgements

Agradeço o apoio de todos os que partilharam o percurso académico comigo.
Aos camaradas de CT, aos colegas de casa e à Mafalda.

Keywords

quantitative trading, algorithmic trading, machine learning, backtest, brokerage, time series analysis, discrete event simulation

Resumo

A crescente democratização dos mercados financeiros alimentada por novas tecnologias abriu a porta a novos investidores e investigadores. Os mercados mudaram, a negociação contínua tornou-se uma realidade e o número de ordens financeiras aumentou exponencialmente. Com o aumento da flexibilidade e acessibilidade aos mercados, a negociação algorítmica a titulo individual cresceu, com investidores a implementar seus próprios algoritmos nas estratégias de negociação. O uso de algoritmos de aprendizagem automática e análise de séries temporais tornou-se comum, aumentando a complexidade das estratégias de negociação.

Para criar e testar algoritmos lucrativos, existem regras que devem ser seguidas. Esta dissertação apresenta o desenvolvimento de uma nova geração de sistemas de investigação e negociação cujo objectivo é ajudar investigadores e investidores a aumentar a produtividade e eficiência. Foi desenvolvido como um sistema de testes baseado em eventos e um sistema de negociação em tempo-real com uma abordagem inovadora para compartilhar relatórios. Além disso, ao fundir a negociação baseada na análise técnica com novas técnicas e cumprindo com o paradigma de testes, o objetivo é proporcionar um ambiente mais rico aos usuários.

Keywords

quantitative trading, algorithmic trading, machine learning, backtest, brokerage, time series analysis, discrete event simulation

Abstract

The growing democratization of financial markets fueled by new technologies opened the door to new investors and researchers. Markets changed, continuously negotiation began and the number of financial orders rose exponentially. With the increase in flexibility and accessibility to markets, algorithmic trading grew at the retail level, with traders starting to implement their own algorithms in trading strategies. The use of machine learning algorithms and time series analysis became widely popular, adding complexity to trading strategies.

In order to create and test profitable algorithms there are rules that must be followed. This dissertation presents the development of a new generation of research and trading system that aims to help researchers and traders to be more productive and efficient. It was developed as an event-driven backtest and live trading system with an innovative approach to sharing backtest reports. Also, by merging the technical analysis based trading with new techniques, and complying with the backtest paradigm, the aim is to provide a richer environment to users.

Contents

List of Figures

List of Tables

Acronyms

ADF	Augmented Dickey Fuller	**LR**	Logistic Regression
ADX	Average Directional Index	**LSVC**	Linear Support Vector Classifier
API	Application Programming Interface	**MACD**	Moving Average Convergence Divergence
AT	Algorithmic Trading	**MOOC**	Massive Open Online Course
ATR	Average True Range	**OBV**	On Balance Volume
BB	Bollinger Bands	**OHLC**	Open-Low-High-Close
BC	Bagging Classifier	**OLS**	Ordinary Least Square
CADF	Cointegrated Augmented Dickey Fuller	**OS**	Operating System
CPU	Central Processing Unit	**PCA**	Principal Component Analysis
CSV	Comma-separated Values	**QDA**	Quadratic Discriminant Analysis
EMA	Exponential Moving Average	**RFC**	Random Forest Classifier
FCM	Futures Commission Merchants	**RFE**	Recursive Feature Elimination
FOREX	Foreign Exchange	**RMS**	Root Mean Square
GUI	Graphical User Interface	**RSI**	Relative Strength Index
IB	Interactive Brokers	**RSVM**	Radial Support Vector Machine
HE	Hurst Exponent	**SMA**	Simple Moving Average
HFT	High Frequency Trading	**STD**	Standard Deviation
HR	Hedge Ratio	**VC**	Voting Classifier
JSON	JavaScript Object Notation	**WET**	Western Europe Time
LDA	Linear Discriminant Analysis	**ZMQ**	ZeroMQ

Definitions

Security - A security is a negotiable financial instrument that holds some type of monetary value.
Long - A long (or long position) is the buying of a security such as a stock, commodity or currency with the expectation the asset will rise in value.
Short - A short, or short position, is a directional trading or investment strategy where the investor sells a borrowed security in the open market with the expectation the asset will drop in value.
Pip - A pip is the smallest price move that a given exchange rate makes based on market convention.
Currency Pair - A currency pair is the quotation and pricing structure of the currencies traded in the forex market; the value of a currency is a rate and is determined by its comparison to another currency.
Fundamental Analysis - Fundamental analysis is a method of evaluating a security in an attempt to measure its intrinsic value, by examining related economic, financial and other qualitative and quantitative factors.
Technical Analysis - Technical analysis is a trading tool employed to evaluate securities and attempt to forecast their future movement by analyzing statistics gathered from trading activity, such as price movement and volume.
Retail Trader - A retail trader is an individual investor who buys and sells securities for his/hers personal account, and not for another company or organization
Paper trading - A paper trade refers to using simulated trading to practice buying and selling securities without actual money being involved.
Trailing Stop - A stop order that can be set at a defined percentage away from a security's current market price.
Limit Order - A limit order is a take-profit order placed with a bank or brokerage to buy or sell a set amount of a financial instrument at a specified price or better.
Market Neutral - A market-neutral strategy is a type of investment strategy undertaken by an investor or an investment manager that seeks to profit from both increasing and decreasing prices in one or more markets, while attempting to completely avoid some specific form of market risk.
Trading Strategy - A trading strategy is a fixed plan that is designed to achieve a profitable return by going long or short in markets.
Volume - Volume is the number of shares or contracts traded in a security or an entire market during a given period of time.

CHAPTER 1

Introduction

This chapter presents this dissertation's context and purpose. It starts by introducing the reader to the underlying issues that led to the need for this work. Afterwards it proceeds to explain the goals it aims to achieve and finally it explains how the rest of the document is structured.

1.1 Contextualization

One aspect of financial markets that has intrigued investors for many years is whether there exist technical trading rules based on pattern in prices which can be relied on to make money. In order to uncover the pattern a technical analysis approach is needed. Technical analysis refers to the use of past prices, trading volume and other variables to forecast future price changes. However, Sanford J. Grossman and Joseph E. Stiglitz (Grossman and Stiglitz, 1980) showed that it is impossible for a market to be perfectly informationally efficient [1]. Because information is costly, prices cannot perfectly reflect the information which is available, since if it did, investors who spent resources on obtaining and analysing it would receive no compensation. LeRoy and Porter (1981) showed that stock markets exhibit 'excess volatility' and they reject market efficiency [2]. Despite academic skepticism, technical analysis continues to be widely used in practice. In earlier days, it was practiced with the aid of pencil and paper, more or less equated with "chartism" but in recent decades there was a transformation caused by the increasing use of computers [3].

Technology has revolutionized the way financial markets function and the way financial assets are traded. Two significant interrelated technological changes are investors using computers to automate their trading processes and markets reorganizing themselves so virtually all markets are now electronic limit order books [4]. As the markets evolved, the automation of trading processes became more popular, not only in the big banks and hedge funds but also in the retail trading world.

With this new trend of automation, created by the new wave of technology, trading evolved. One form particularly has changed the markets. Quantitative trading consists of trading strategies based on quantitative analysis, which rely on mathematical computations to generate trading opportunities [5]. Quantitative trading is a sophisticated area of quant finance. Not only that, but it requires extensive programming expertise, at the very least in a language such as MATLAB, R or Python [6].

1.2 MOTIVATION

Although sophisticated, Quantitative trading is becoming more and more democratized. Brokerage services are starting to offer Application Programming Interfaces (APIs) for their trading systems and new initiatives, such as Quantopian[1] are changing the norms by creating crowd-source hedge funds for quant freelancers. Also, one of the biggest factors for this democratization is the current state of the art in technologies like Python or R. In light of this change, the need for new platforms that allow research in quantitative trading arises.

The main purpose of this dissertation is to suggest and create a system for researchers to backtest tradings strategies, to perform statistical analysis on financial data and live trade their algorithms. Also, it is aimed to create and backtest example strategies with the new developed system.

It is within the objective of this work to offer all tools necessary to evaluate the true performance of a trading strategy according to the state of the art regarding quantitative trading, particularly algorithmic trading.

1.3 OUTLINE

- **Chapter 2** describes the state of the art surrounding the topic of quantitative trading including the current paradigm of trading strategies, a techonology review over pertinent solutions and the concept of broker;
- **Chapter 3** elaborates on the concept of backtesting and its pitfalls, how to prepared data and how benchmark strategies;
- **Chapter 4** describes the requirements that this system should fulfill, as well as how it was planned and conceptually designed;
- **Chapter 5** contains all the implementation done during the length of this work; planned and conceptually designed;
- **Chapter 6** presents the results for the system's performance, for the example strategies and other studies performed;
- **Chapter 7** discusses the obtained results, how they are relevant and future improvements for this work

[1]https://www.quantopian.com/

CHAPTER 2

State of the Art

This chapter describes the State of the Art surrounding the topic of Algorithmic trading.

2.1 Algorithmic trading

As a subset of quantitative trading, Algorithmic Trading is the use of an automated system for carrying out trades, which are executed in a predetermined manner via an algorithm without human intervention. For AT engines, speed of execution, availability of real-time market data and minimum latency have become key success factors as already milliseconds can make a difference [7]. Algorithmic Trading strategies are designed prior to the commencement of trading and can be executed without discretionary input from human traders. AT possesses numerous advantages over discretionary methods:

- With a fully automated system there is no need for an individual or team to be constantly monitoring the markets for price action or news input. This frees up time for the developer(s) of the trading strategy to carry out more research and thus, depending upon capital constraints, deploy more strategies into a portfolio.
- Furthermore by automating the risk management and position sizing process, with a set of systematic strategies, it is necessary to automatically adjust leverage and risk factors dynamically, directly responding to market dynamics in real-time. This is not possible in a discretionary world, as a trader is unable to continuously compute risk and must take occasional breaks from monitoring the market.Furthermore by automating the risk management and position sizing process, with a set of systematic strategies, it is necessary to automatically adjust leverage and risk factors dynamically, directly responding to market dynamics in real-time. This is not possible in a discretionary world, as a trader is unable to continuously compute risk and must take occasional breaks from monitoring the market.
- Another advantage of an automated trading system is that there is no need for subsequent discretionary input. Therefore errors caused by fear and greed that can be overwhelming

to the trader are avoided. This refers to modification of trades at the point of execution or while in a position motivated by the trader's emotions.

Strategies that operate at higher frequencies over many markets become possible in an automated setting. Indeed, some of the most profitable trading strategies operate at the ultra-high frequency domain on limit order book data. These strategies are simply impossible for a human to carry out.

The most important advantage in creating an automated strategy is that its performance can be ascertained on historical market data, which is to a certain degree, representative of future market data. This process is known as Backtesting. A backtest system can have different designs and architectures. It can be, for example, a vectorized system or an event-driven system. The latter uses a discrete event simulator that allows the (prior) statistical properties of the strategy to be determined, providing insight into whether a strategy is likely to be profitable in the future. A discrete-event simulation is a system in which the operations are modelled as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur; thus the simulation can directly jump in time from one event to the next [8].

But more than just performing due diligence, backtesting allows experimenting with variations of the original strategy, thereby refining and improving the strategy. Backtesting can also provide as output a set of metrics that can be used to benchmark different algorithmic trading strategies. Sharpe Ratio, Maximum Drawdown and Maximum Drawdown Duration are three main metrics for evaluating the performance of a strategy [9].

While the advantages of algorithmic trading are numerous there are some disadvantages. Obtaining data feeds for intraday quantitative strategies is not cheap for the retail trader. Depending upon the latency needs it may be necessary to co-locate a server in an exchange, which increases the monthly costs. For the interday retail trader this is not necessarily an issue, but it is worth considering. In order to mitigate these disadvantages a more robust internet connection and powerful (and thus expensive) desktop machines can be be purchased [9].

2.2 DISCRETE EVENT SIMULATORS

A discrete-event simulation is a system in which the operations are modelled as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system [8]. Between consecutive events, no change in the system is assumed to occur; thus the simulation can directly jump in time from one event to the next. This contrasts with continuous simulation in which the simulation continuously tracks the system dynamics over time. Time is broken up into small time slices and the system state is updated according to the set of activities happening in the time slice [10]. Because discrete-event simulations do not have to simulate every time slice, they can typically run much faster than the corresponding continuous simulation. A more recent method is the three-phased approach to discrete event simulation (Pidd, 1998) [11]. In this approach, the first phase is to jump to the next chronological event. The second phase is to execute all events that unconditionally occur at that time (these are called B-events). The third phase is to execute all events that conditionally occur at that time (these are called C-events). The three phase approach is a refinement of the event-based approach in which simultaneous events are ordered so as to make the most efficient use of computer resources. The three-phase approach is used by a number of commercial simulation software packages, but from the user's point of view, the specifics of the underlying simulation method are generally hidden.

Discrete event simulators are used widely in finance. One case of this use is the trading strategy backtester, in which the system models events based on historical data. This historical data, in the form of a time series, is used as the input, in which, each point is a discrete variable. On each period the system has a set of rules that are checked in order to produce the output. In this case the output is a set of metrics that show the profitability of a trading strategy.

2.2.1 OMNET++

OMNeT++ is an object-oriented discrete event simulator written in C++. It is primarily designed for simulation tasks especially for performance modeling of computer and data communication networks. It could be used to simulate different routing protocols, whether MANET or WSN, it has a lot of examples to illustrate how these protocols work in details [12]. OMNeT++ provides a GUI with an unique object inspector for zooming into components and to display the current state of each component at any time during the simulation. One advantage it has is the automatic generation of scenarios, as no proprietary binary formats are used to store models. Another advantage is the support for parallel and distributed simulations, running not only on multiprocessor system, but also in a network of distributed hosts. In OMNeT++, the most important elements besides events are modules. Being open source software it allows users to change the source code to suit their needs. It builds on small modules that can be reused and combined to more complex modules. Thus a hierarchy could be generated and different levels of abstraction can be realized. There are two different types of modules: simple modules and compound modules. Simple modules are implemented in C++ and represent the active components of OMNeT++ where events occur and model behaviours are defined. Compound modules are a composition of other simple or compound modules and are used as containers to structure a model [12].

2.2.2 SIMPY

SimPy[1] is a process-based discrete-event simulation framework based on Python language. It has an event dispatcher based in Python's generators. Processes in SimPy are simple Python generator functions and are used to model agents. This agents can be vehicles or customers depending on the use case. Simulations can be performed "as fast as possible", in real time (wall clock time) or by manually stepping through the events. Also, SimPy can be used for asynchronous networking or to implement multi-agent systems (with both, simulated and real communication). SimPy also provides various types of shared resources to model limited capacity congestion points (like servers, checkout counters and tunnels) [13].

The SimPy distribution contains tutorials, vast documentation, and examples. SimPy is released as open source software under the MIT License. The first version was released in December 2002.

2.2.3 DESMO-J

DESMO-J[2] is an acronym for Discrete-Event Simulation Modelling in Java. It allows for rapidly and flexibly building discrete event simulation models in Java, supporting both the event-oriented and process-oriented paradigms. DESMO-J offers an extensive set of readily usable Java classes for stochastic distributions, static model components (like queues or resource synchronization), time representation and scheduling, experiment conduction and reporting. Supported by this simulation infrastructure, the user is free to concentrate on specifying the model's behaviour in terms of events or processes [14]. DESMO-J has been developed at University of Hamburg's and it was first released in 1999 with its the environment being maintained and kept up to date. DESMO-J's predecessor was DESMO, a Modula-2-based simulation library, which in turn was inspired by DEMOS, a system for discrete event modelling on Simula [15].

Besides providing a hybrid discrete event simulation environment able to process event as well as process model descriptions, key features of DESMO-J include: A GUI for experiment conduction, 2D animation based on icons and symbols and 3D visualization based on Java3d. Also, an online tutorial is available on the project web page. Most real-world DESMO-J applications focus on manufacturing and logistics. DESMO-J is integrated into business process modelling tools like Borland Together or Intellivate IYOPRO, augmenting these tools with simulation functionality [15].

2.3 TRADING AND BACKTEST SYSTEMS

In finance, a trading platform is a computer software program that can be used to place orders for financial assets, over a network, using a broker. Some of these trading platform

[1]http://simpy.readthedocs.io
[2]http://desmoj.sourceforge.net/home.html

also include technical analysis with automate charting and price generated indicators. Others include algorithmic trading. There are also the backtest systems, which allow assessing the performance of an automated strategy using historical price data.

There are systems that only offer the purchase of financial products, these are normally web applications owned by brokerage firms, such as Plus500[3] [16]. In addition, in the opposite field, there are several open-source software libraries for finance. Usually this type of software is utilized by researchers and developers, in contrast to the previously described web applications that target retail traders. QuantLib, for example, offers an extensive framework for quantitative finance. Zipline is the engine used by Quantopian to backtest and execute trades, it comes with statistical and machine learning features [17][18]. TA-Lib is widely used to generate indicators with price data and to perform technical analysis [19]. Lastly there are the more complex systems that target both institutional traders and retail traders, such as MetaTrader, ProTrader or TradeStation. These systems include algorithmic trading, technical analysis indicators and backtesting [20][21][22].

In this work there was a particular interest in two solutions. Metatrader because it includes live trading with indicators, automated trading and backtesting, in addition to its popularity in the FOREX trading world. Quantopian because it offers the tools for quantitative analysts to develop, test and trade as freelancers to a crowd-sourced hedge fund.

2.3.1 METATRADER

Metatrader 4 is an electronic trading platform widely used by online retail foreign exchange speculative traders. The first solution was developed by MetaQuotes Software and it was released in 2002. MetaTrader is licensed to foreign exchange brokers who provide the software to their clients [20]. Metatrader has a server side which is run by the broker and a client software provided to the broker's customers, who use it to see live streaming prices and charts, to place orders, and to manage their accounts. The client also provides a discrete event simulator in the form of a backtester to the user. This architecture can be seen in figure 2.1.

The client component is a Microsoft Windows application with many functionalities. It includes a built-in editor and compiler with access to a user contributed free library of software, articles and help. One of the reasons for MetaTrader's popularity was its support of algorithmic trading.

[3]https://www.plus500.pt/

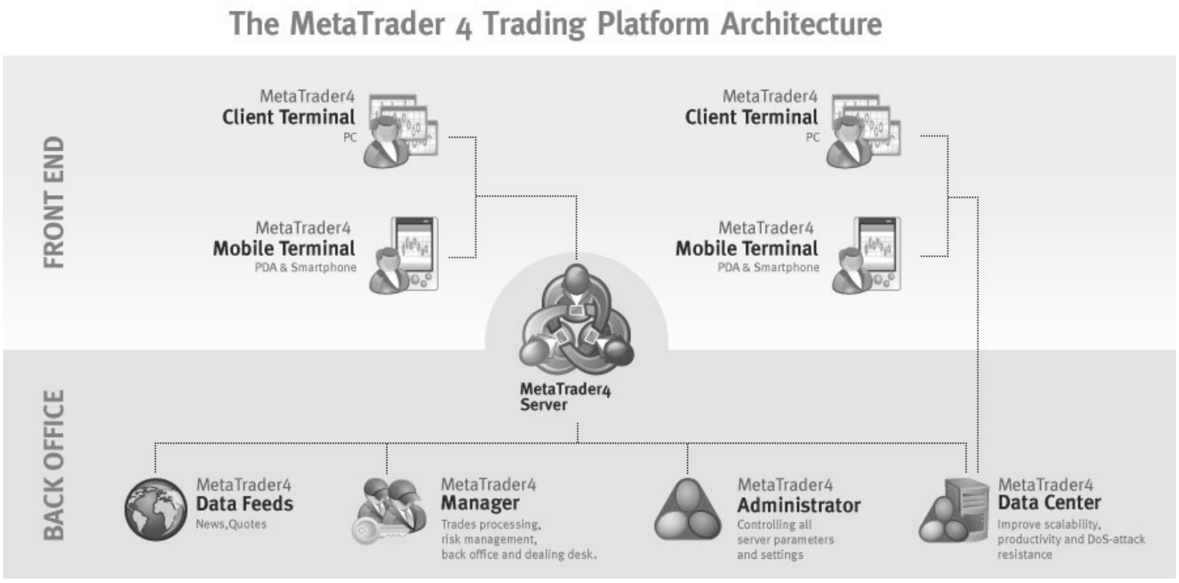
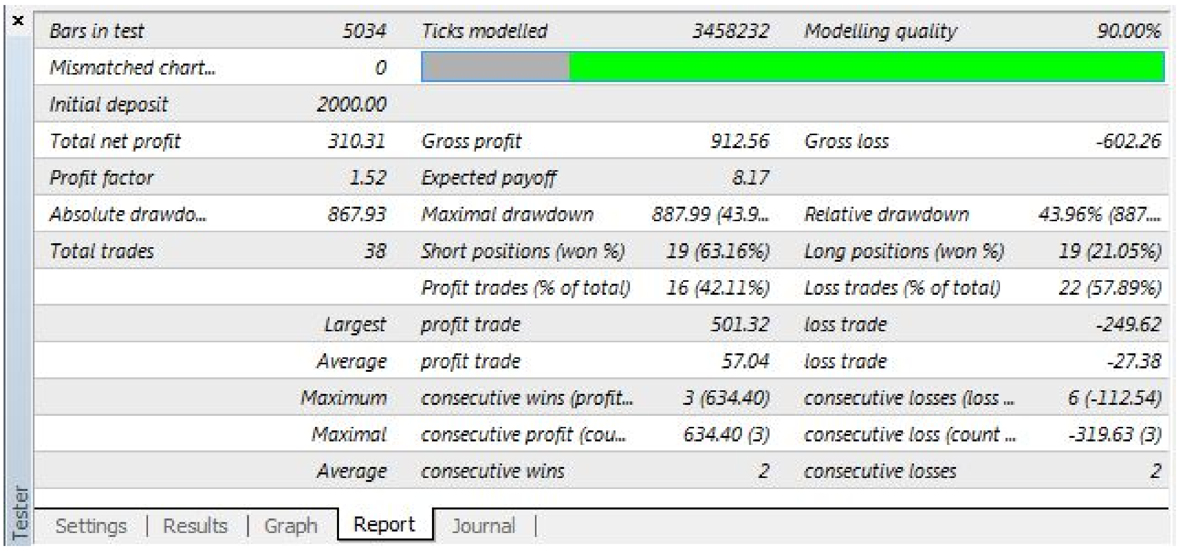

The MetaTrader 4 Trading Platform Architecture
FRONT END
BACK OFFICE
MetaTrader4 Client Terminal PC
MetaTrader4 Mobile Terminal PDA & Smartphone
MetaTrader4 Client Terminal PC
MetaTrader4 Mobile Terminal PDA & Smartphone
MetaTrader4 Server
MetaTrader4 Data Feeds News,Quotes
MetaTrader4 Manager Trades processing, risk management, back office and dealing desk.
MetaTrader4 Administrator Controlling all server parameters and settings
MetaTrader4 Data Center Improve scalability, productivity and DoS-attack resistance

Figure 2.1: Metatrader 4 architecture[20]

Metatrader calls trading strategies "Expert Advisors", the scripts are written in MQL4 language and compiled to .ex4. Compiled Expert Advisors can be backtested using the built-in backtester, as shown in figure 2.2. The input parameters are: instrument, price, period, spread and historical data length. The backtest outputs a full report with the number of iterations, returns, number of trades, drawdown, spread, wins and losses. A graph representing the changes in equity is also outputted. After testing, the Expert Advisor can be attached to a chart and manage trades automatically without any discretionary input by the user.



Bars in test	5034	Ticks modelled	3458232	Modelling quality	90.00%
Mismatched chart...	0				
Initial deposit	2000.00				
Total net profit	310.31	Gross profit	912.56	Gross loss	-602.26
Profit factor	1.52	Expected payoff	8.17		
Absolute drawdo...	867.93	Maximal drawdown	887.99 (43.9...	Relative drawdown	43.96% (887....
Total trades	38	Short positions (won %)	19 (63.16%)	Long positions (won %)	19 (21.05%)
		Profit trades (% of total)	16 (42.11%)	Loss trades (% of total)	22 (57.89%)
	Largest	profit trade	501.32	loss trade	-249.62
	Average	profit trade	57.04	loss trade	-27.38
	Maximum	consecutive wins (profit...	3 (634.40)	consecutive losses (loss ...	6 (-112.54)
	Maximal	consecutive profit (cou...	634.40 (3)	consecutive loss (count ...	-319.63 (3)
	Average	consecutive wins	2	consecutive losses	2

Figure 2.2: Example of Backtest Report on Metatrader 4 platform

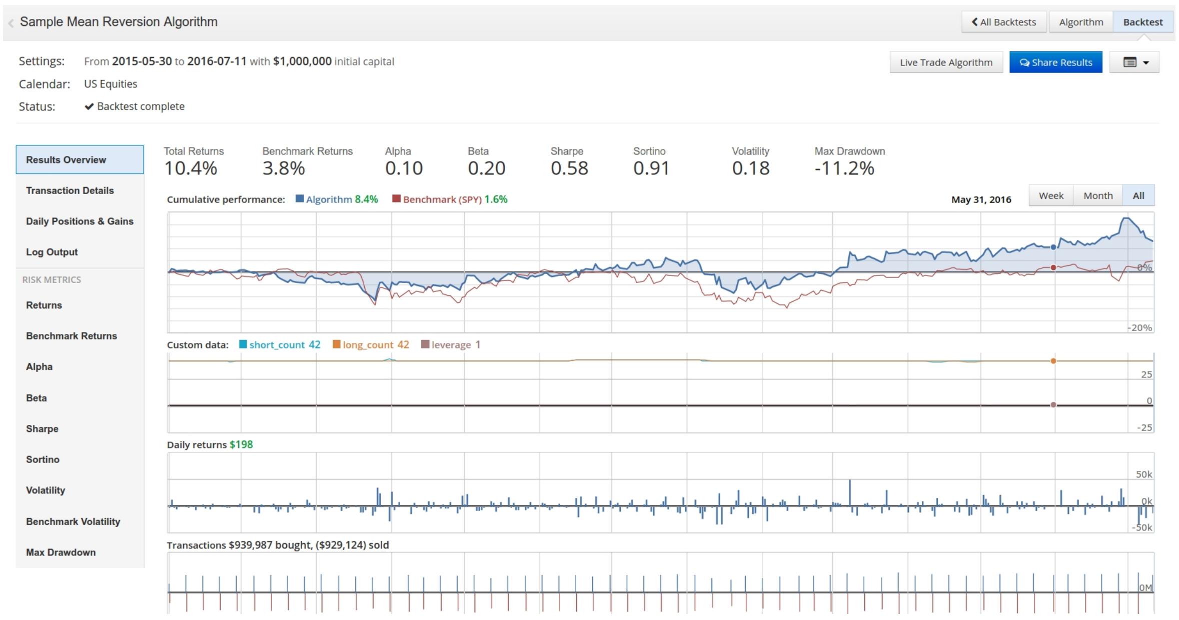
Metatrader 5, the most recent Metatrader version, brought many changes. A native support for 64-bit architecture was granted and the server architecture changed with MT5 implementing a distributed architecture. Also, the number of instruments, orders database size and the number of account groups became unlimited. Table 2.1 shows some of the bigger changes between version 4 and 5 [23].

Feature	MetaTrader 5	MetaTrader 4
Reports	Charts(HTML5) + tables	Tables only
Manager API	C++, ASP.Net, PHP	C++
Gateway API	Yes	No
Technical Indicators	38	30
Timeframes	21	9
Economic Calendar	Yes	No
Strategy Tester	Multi-threaded + Multi-currency + Real ticks	Single thread
Supported Markets	Forex / CFDs / Futures /Options / Stocks / Bonds	Forex / CFDs
Order Fill Policy	Fill or Kill / Immediate or Cancel / Return	Fill or Kill

Table 2.1: Features offered by MetaTrader 5 vs features in MetaTrader 4

2.3.2 QUANTOPIAN

Quantopian is a platform that allows users to learn, design trading algorithms and backtest them. Quantopian's web-based product is written in Python and relies on the open source backtesting engine Zipline. Live trading is guaranteed by coupling the trading algorithm's buy/sell orders to a brokerage account at Interactive Brokers [24][25]. The uploaded algorithms of users remain the trade secrets of the individual (unless the person chooses to publish them). Quantopian's backtest offers users an extensive report on strategy performance, shown in figure 2.3. On the Results Overview tab the user can analyze five graphs: Cumulative performance, custom data, daily returns and transactions. Transactions are described in the Transaction Details section, where the user can see when each transaction happen, which security was traded, the amount transactioned or the order type. There is also a section for analyzing daily positions and gains in the portfolio. Tables for analyzing metrics in the one month, three month, six month and twelve month period are also showed. The offered metrics are: Returns, Benchmark Returns, Alpha, Beta, Sharpe, Sortino, Volatility, Benchmark Volatility and Maximum Drawdown.


Sample Mean Reversion Algorithm

Settings: From 2015-05-30 to 2016-07-11 with $1,000,000 initial capital

Calendar: US Equities

Status: ✓ Backtest complete

Total Returns	Benchmark Returns	Alpha	Beta	Sharpe	Sortino	Volatility	Max Drawdown
10.4%	3.8%	0.10	0.20	0.58	0.91	0.18	-11.2%

Cumulative performance: Algorithm 8.4% Benchmark (SPY) 1.6%

Custom data: short_count 42 long_count 42 leverage 1

Daily returns $198

Transactions $939,987 bought, ($929,124) sold

Figure 2.3: Screenshot of backtest result on Quantopian platform

In terms of technologies, Quantopian's web application is written in Python and uses the engine Zipline [18]. Zipline, an event-driven system, supports both backtesting and live-trading. In order to provide statistical and machine learning features it uses popular python packages like statsmodels, scipy or sklearn. It also uses pandas dataframes to give structure to data.

Quantopian is also a crowd-funding hedge fund that organizes competitions to find the best trading algorithms and a MOOC-like platform for higher education [26].

2.4 FOREX

There are several global financial markets but one stands out for being by far the largest market in the world in terms of trading volume. This market is known as the Foreign Exchange and is the decentralized global market where currencies are traded 24/7. This includes all aspects of buying, selling and exchanging currencies at current or determined prices. Foreign exchange transactions encompass everything from the conversion of currencies by a traveler at an airport kiosk to billion-dollar payments made by corporations, financial institutions and governments. Transactions range from imports and exports to speculative positions with no underlying goods or services. Some of the more popular ways that traders participate in the forex market is through the spot market, futures, options, and exchange-traded funds [27].

Forex trading is the simultaneous buying of one currency and selling another. Currencies are traded through a broker or dealer, and are traded in pairs. Since currencies are always traded in pairs, the foreign exchange market does not set a currency's absolute value but rather determines its relative value by setting the market price of one currency if paid for with another. For example the USD/EUR pair represents the relationship between the US Dollar and the Euro quotes. In this case the base currency is the USD and the quote currency is the EUR. Also, in this example if the investor predicts a weaker Dollar vs a stronger EUR, then the pair should be sold. If the investor thinks the Dollar is rising and EUR is weaker, then the pair should be bougth in order to generate profit [27].

Increasing globalization has led to a massive increase in the number of foreign exchange transactions in recent decades. The largest trading centers are London, New York, Singapore and Tokyo. The global foreign exchange market, being the largest financial market in the world, has an average daily volume in the trillions of dollars. Foreign exchange transactions can be done for spot or forward delivery [28].

2.5 BROKERAGE FIRMS

The world of trading is divided between retail and institutional traders. Contrary to institutional traders, retail traders don't have direct access to markets, they must use the service of a third party. These third parties, usually called Brokers, act as an agent for a costumer and charges commission for its services. In most markets brokers are intermediaries who match buyers and sellers, however in the case of the FOREX market, the concept of broker changes. Investors deal directly with the market maker or dealer. This has the the upside of lowering the transaction costs and the downside of limiting the trader to the quotes offered by the broker. These entities have limited use because the broker can trade with only a single market maker, acting as a price taker, while entities like Interactive Brokers, are the counterparty to all trades. In the retail FOREX market, an IB or Futures Commission Merchants representative fills the role of the broker in the pure sense of the word. The existence of two types of brokers, regular brokers and broker-resellers is one of the major flaws of the current Forex trading mechanism. Regular brokers generally are considered more notable than broker-resellers [29][30].

When choosing the right brokerage service each case is different but there are some aspects that need to be contemplated. Trade execution fees are important, but there are other brokerage fees to consider, as well. Here are some additional costs to consider

- **Minimums and Margins** - Most brokers have a minimum balances for starting a brokerage account. Margin accounts, accounts in which the broker lends the customer cash to purchase securities, usually have higher minimum balance requirements than standard brokerage accounts. It's also important to take a look at the interest rates that broker charges when making a trade on margin.
- **Withdrawal** - Sometimes it can be hard to get money out from the brokerage account. That occurs because brokers sometimes charge fees to make a withdrawal, or they won't let you take any money out if it will drop your balance below the minimum.
- **Complicated Fee Structures** - Some brokers have complex fee structures that make it harder to sort out hidden fees. This is particularly common among broker-resellers who may use fee structure as a selling point to entice clients. If the rates seem too good

to be true, a careful reading of the account agreement and fee summaries may show hidden additional fees [29].

2.6 TECHONOLOGY REVIEW

In the previous sections, the concepts of Algorithmic trading and Backtesting were introduced. The technical requirements for systems that offer both were also discussed in sections 2.2 and 2.3. This section aims to evaluate and explain technologies that can be utilized to build a system with such requirements. It will focus in programming languages and libraries.

2.6.1 PYTHON LANGUAGE

Python is a high-level interpreted programming language and has a design philosophy which emphasizes on code readability. Python syntax allows programmers to express concepts in fewer lines of code than in other lower level languages. It has an extensive array of libraries and a large community. In the particular case of finance applications, libraries such as Numpy(vectorised operations), scikit-learn(machine learning algorithms), matplotlib(data visualization), statsmodel(statistical modelling) or pandas(time series analysis) provide a very powerful stack for developers. Bokeh[4] is a Python interactive visualization library, based on matplotlib it offers high performance with large datasets leveraging from WebGL[5]. An example of a Bokeh graph can be seen in figure 2.4.

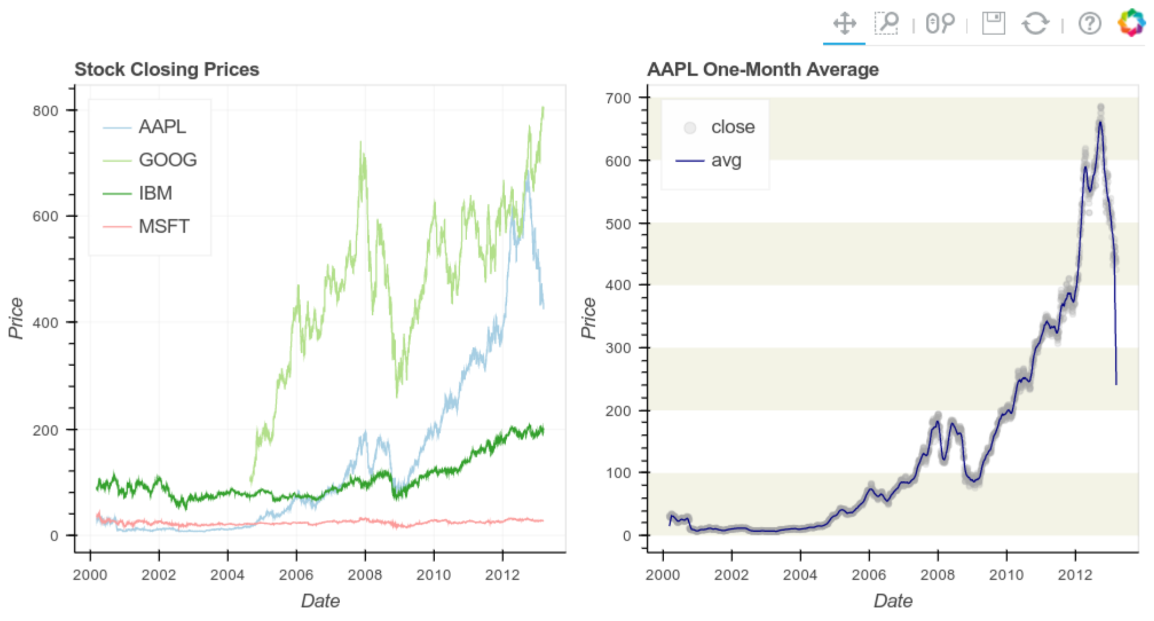
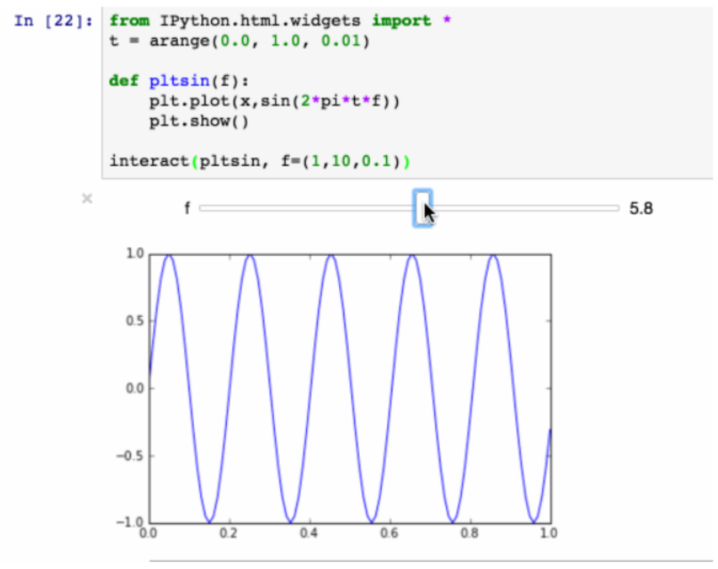


Figure 2.4: Example of data visualization using Bokeh[31]

[4]http://bokeh.pydata.org
[5]https://www.khronos.org/

A big advantage in using Python is IPython and Jupyter Notebooks[6]. IPython[7] is a command shell for interactive computing that provides support for interactive data visualization and use of GUI toolkits [32]. IPython provides a Python kernel to Jupyter, which is an open-source web application that allows creating and sharing documents containing live code, equations or visualizations. Jupyter notebooks are widely used in Data Science, their sharing capabilities provide interactive reports for researchers [33]. This, in a team environment, can improve productivity, as well as, produce reports of higher quality. In the context of finance, these notebooks are of high value because they can be used, by quants and researchers, to produce backtest reports tailored to their strategies. These reports are essential in the process of improving a trading algorithm, in part because of the need for good data visualization and flexibility.



```
In [22]: from IPython.html.widgets import *
         t = arange(0.0, 1.0, 0.01)

         def pltsin(f):
             plt.plot(x,sin(2*pi*t*f))
             plt.show()

         interact(pltsin, f=(1,10,0.1))
```

Figure 2.5: Example of interactive data visualization using IPython with Jupyter Notebooks[34]

Learning Python is also easier when compared to languages such as C++ and excels at development speed, hence improving productivity. As for speed of execution this language is not as fast as compiled languages, such as C, but by using heavily optimised methods(Numpy for example uses vectorisation) good results can be obtained. Also solutions like Cython[8] can be used to improve speed, with the downside of having an increase in code complexity. The open source and cross-platform nature of Python is also a plus for developers. This characteristics make Python the perfect language to create backtesting and execution environment for algorithmic trading. There is however the case of Ultra-High Frequency trading, in which python can't compete with C because of the speed requirements.

[6]https://jupyter.org/
[7]http://ipython.org/
[8]http://cython.org/

2.6.2 R LANGUAGE

R is a language and environment for statistical computing and graphics. It is similar to the S language and is considered as a different implementation. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Another is it's large community [35]. R has a big library of statistical packages for specialized statistical work. It's packages cover everything from Psychometrics and Genetics to Finance. Also, R is better for data visualization, packages like ggplot2 make plotting easier and more customizable. R builds in data analysis functionality by default, whereas Python relies on packages [36]. Because Python is a general purpose language, most data analysis functionality is available through packages like NumPy and pandas. However, R was built with statistics and data analysis in mind, so many tools that have been added to Python through packages are built into base R. This is harder to integrate to a production workflow and has a steeper learning curve compared to languages like Python.

2.6.3 C/C++ LANGUAGE

C is a general-purpose, structured language. C++ is a general-purpose object-oriented programming language and is an extension of the C language. It is therefore possible to code C++ in a "C style" or "object-oriented style." Contrary to Python, C and C++ are middle-level languages. This types of languages don't provide all the build-in functions found in high level languages, but provide all building blocks that are needed to produce the desired result.

If ultimate execution speed is desired then C++ (or C) is likely to be the best choice. It offers the most flexibility for managing memory and optimising execution speed [37]. C++ could be the best choice for creating an event-driven backtester that needs extremely rapid execution speed, such as for HFT.

2.7 TRADING STRATEGIES

A trading strategy can be regarded as a signal processing element that uses external information and past prices as inputs and incorporates them into future prices [38]. As stated before in section 2.1, strategies are designed prior to the commencement of trading and can be executed without discretionary input from human traders. There are several types of strategies, in this section it will be outlined the most common type of strategies in financial markets. The two most common types are Trend Following and Mean Reverting. There are also the Momentum strategies that rely on Trend Following or Mean Reverting in addition of momentum indicators. Pairs trading is also a very common way to explore correlations between financial instruments [39]. Finally learning algorithms will be considered because strategies that rely on forecasting provided by machine learning are becoming a new reality in the financial markets [40].

2.7.1 TREND FOLLOWING

Trend following is a trading strategy based on the technical analysis of market prices, rather than on the fundamental analysis. In financial markets, traders and investors using a trend following strategy believe that prices tend to move upwards or downwards during periods of time. They try to take advantage of these market trends by observing the current direction and using this to decide whether to buy or sell. Trend-following investing involves going long in markets that have been rising and going short in markets that have been falling, betting that those trends continue [41]. The most basic trend-following strategy is time series momentum — going long markets with recent positive returns and shorting those with recent negative returns.

The direction of the trend depends upon trading period being analyzed. Real instruments fluctuate minute-to-minute, day-to-day and year-to-year. There is, therefore, an enormous supply of historical points to use to determine trend. As such, it can be determined as many instances of trend as pleased, in any direction [42].

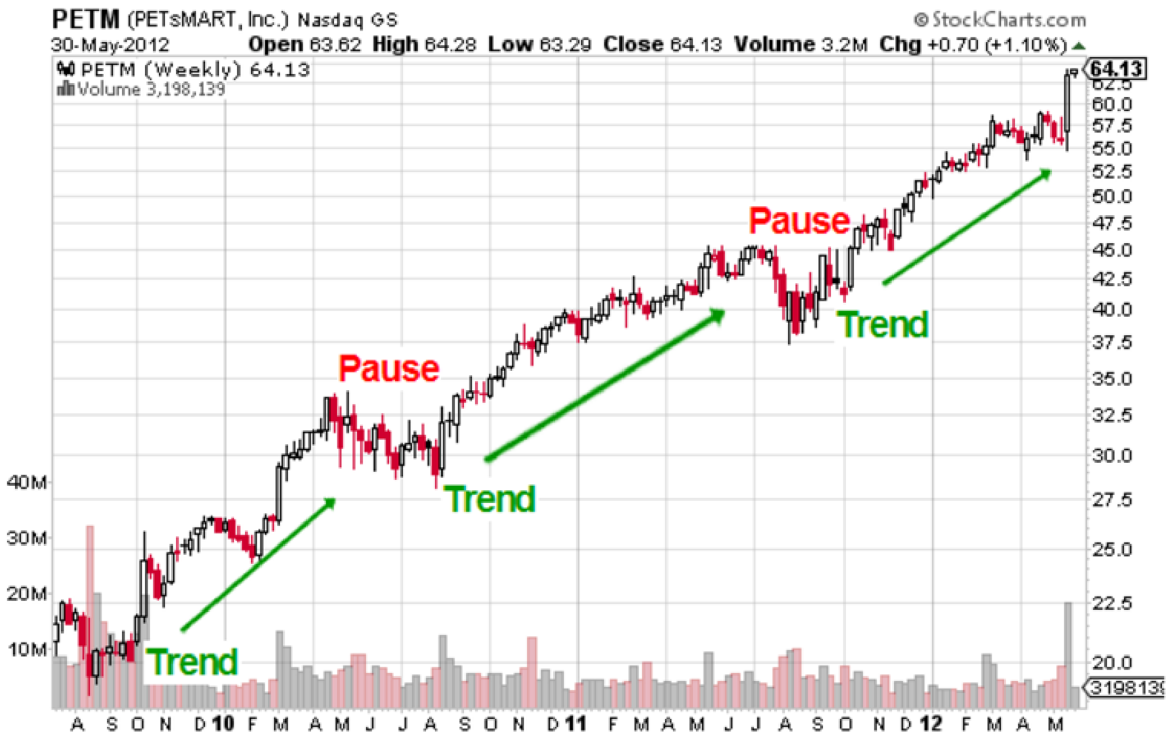


Figure 2.6: Example of Up Trend[43]

2.7.2 MEAN REVERTING

Mean reverting time series can be found everywhere from nature (ex. Water level of the Nile) to social sciences like the famous example Sports Illustrated jinx, which is the claim that "an athlete whose picture appears on the cover of the magazine is doomed to perform poorly the following season" (Kahneman, 2011) [44]. The scientific reason is that an athlete's performance can be thought of as randomly distributed around a mean, so an exceptionally good performance one year (which puts the athlete on the cover of Sports Illustrated) is very likely to be followed by performances that are closer to the average [45].

Reversion to the mean, also called regression to the mean, is the statistical phenomenon stating that the greater the deviation of a random variate from its mean, the greater the probability that the next measured variate will deviate less far. In other words, an extreme event is likely to be followed by a less extreme event [46]. In finance, mean reverting can be understand as the change of the price series in the next period being proportional to the difference between the mean price and the current price [45]. The question is, can mean reversion be constantly observed in a financial price series? Academic research has indicated that stock prices are on average very close to random walking. However, this does not mean that under certain special conditions, they cannot exhibit some degree of mean reversion or trending behavior. Furthermore, at any given time, prices can be both mean reverting and trending depending on the time horizon [9]. Strategies for mean reverting time series are the amongst the simplest non-trivial algorithms. For example, the well known Bollinger Bands strategies in which buy/sell signals are generated if the price stands 2 standard deviations above or below the mean, expressed by a moving average, relies on mean reversion. In order to determine if a time series is mean reverting the ADF test, the Hurst exponent and the Variance Ratio Test can be performed.

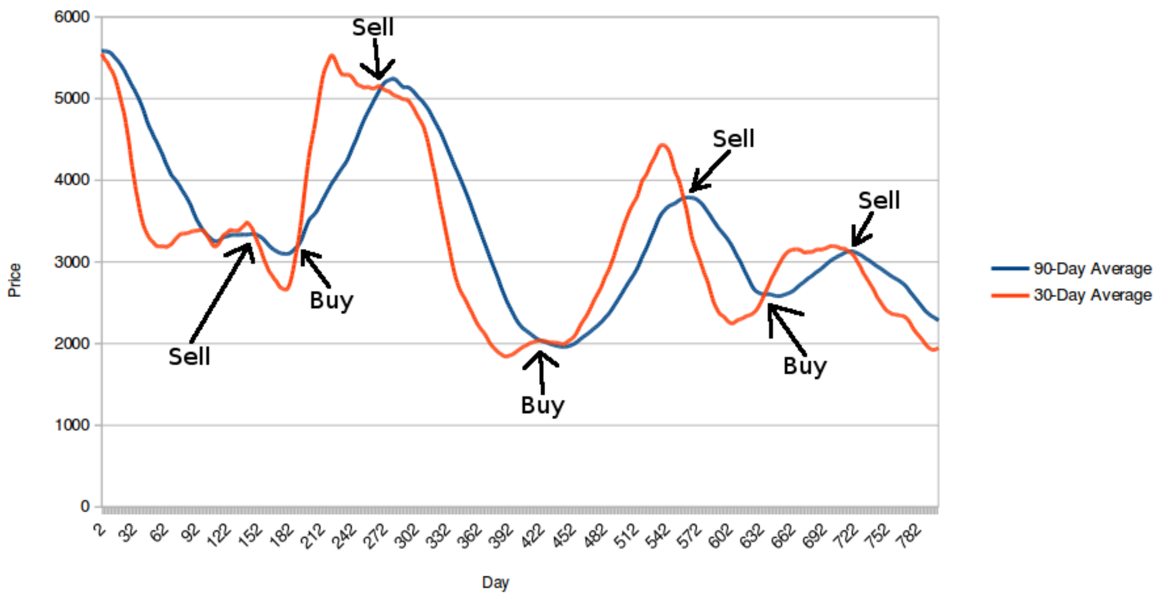


Figure 2.7: Example of Mean Reverting Strategy based on Moving Averages[47]

2.7.3 MOMENTUM STRATEGIES

Constructing a trading strategy is essentially a matter of determining if the prices under certain conditions and for a certain time horizon will be mean reverting or trending, and what the initial reference price should be at any given time.

When the prices are trending, they are also said to have "momentum," and thus the corresponding trading strategy is often called a momentum strategy [9]. Momentum can be generated by the slow diffusion of information—as more people become aware of certain news, more people decide to buy or sell a stock, thereby driving the price in the same direction. Momentum can also be generated by the herdlike behavior of investors: investors interpret the (possibly random and meaningless) buying or selling decisions of others as the sole justifications of their own trading decisions [9]. In an article in the New York Times(Schiller, 2008), Robert Schiller said nobody has all the necessary information to make fully informed financial decisions so one has to rely on the judgment of others [48]. Moskowitz and Grinblatt (Moskowitz & Grinblatt, 1999) also argue that momentum in individual stock returns is driven by momentum in industry returns [49]. Established the nature and concept of Momentum, the question that remains is how to trade the Momentum.

We generally distinguish between two types of momentum strategy:

- The trend following strategy, which consists of buying (or selling) an asset if the estimated price trend is positive (or negative) [50].
- The contrarian (or mean-reverting) strategy, which consists of selling (or buying) an asset if the estimated price trend is positive (or negative) [50].

Strategies that use both momentum properties and trend following need a trend detection method to establish the trend slope and price generated indicators, such as the Moving Average Convergence Divergence or the Average Directional Index, to produce entry/exit signals. Strategies that rely on mean reversion should have one or more indicators to forecast reversions. For example, Simple Moving Averages with different periods or Bollinger Bands.

2.7.4 PAIRS TRADING

Pairs trading is a trading strategy that matches a long position with a short position in a pair of highly correlated instruments. This type of strategies offers the opportunity to profit with both increasing and decreasing prices in one or more markets, this is called a market-neutral strategy.

Pairs traders wait for weakness in the correlation, and then go long on the under-performer while simultaneously going short on the over-performer, closing the positions as the relationship returns to its statistical norm [39]. The strategy's profit is derived from the difference in price change between the two assets, rather than from the direction in which each moves. A residuals series is the result of this difference between prices and its stationarity is of the utmost importance because it has direct impact on the accuracy of price change prediction. A stationary process is one whose statistical properties do not change over time [51], showing no trends or seasonality. So, the more stationary a time series is, more predictable its behavior is. Usually a hedge ratio is calculated in order to both improve stationarity in the residual series and determine position sizing. Michael Halls-Moore and Chan(2013) suggest performing a linear regression against the two selected time series and using the slope value as the hedge ratio. Then the Cointegrated Augmented Dickey-Fuller (CADF) test is used to test the stationarity of the generated residuals series [45][37]. Therefore, a profit can be realized if the long position goes up more than the short, or the short position goes down more than the long (in a perfect situation, the long position will rise and the short position will fall, but this is not a requirement for making a profit). It is possible for pairs traders to profit during a variety of market conditions, including periods when the market goes up, down or sideways, and during periods of either low or high volatility.

The same process used to test and improve stationarity in the residuals time series can be used to select assets to pair in a Pairs trading strategy that are cointegrated. Cointegration measures whether or not the price distance between two or more assets remain stable over time [52]. If we can find a stationary linear combination of several nonstationary price series, then these price series are called cointegrated [45]. Therefore if the residuals series of a pair of assets performs well in the CADF test, showing a stationary behavior, the two assets are cointegrated.

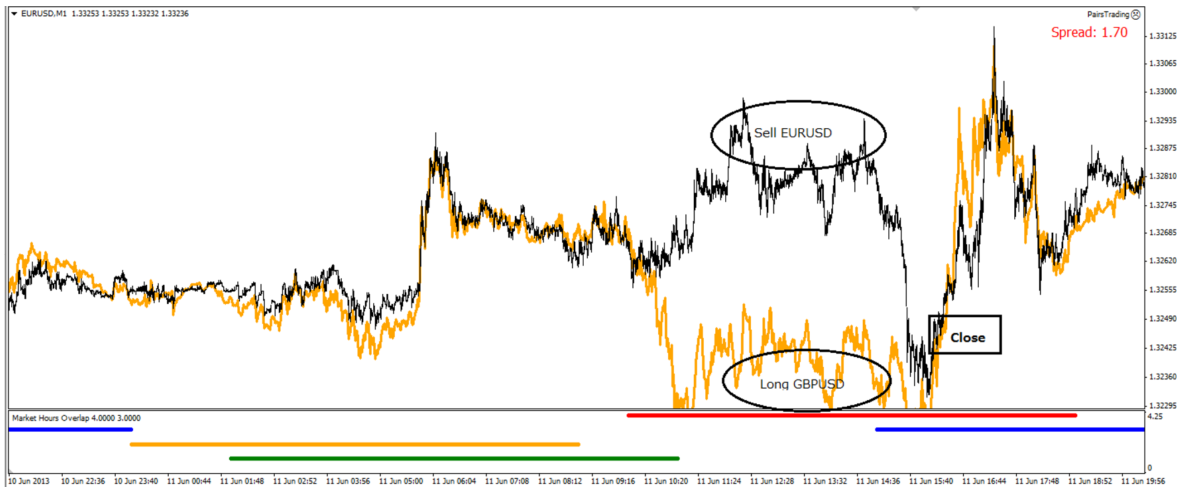


Figure 2.8: Example of Pairs Trading Strategy of two FOREX currency pairs[53]

2.7.5 LEARNING ALGORITHMS

Machine learning has been widely used in Finance in the past years, from predictive analysis for credit scores and bad loans to fraud detection. In the context of trading algorithms, supervised learning models are often used to predict future market returns which can be used to create and validate market entry/exit signals. Predicting asset price trend by interpreting the seemly chaotic market data has always been an attractive topic to both investors and researchers [54]. Among those popular methods that have been employed, Machine Learning techniques are very popular due to the capacity of identifying asset trend from massive amounts of data that capture the underlying asset price dynamics [40].

The difficulty of this problem is mainly related to the dynamic, complex, evolutive and chaotic nature of the markets [55][56][57]. These characteristics of the financial markets clearly showed the limitations of classical statistical methods for stock price time-series predictions, and asked for more powerful methods to accomplish the task. In particular, when dealing with market trends, methods should be able to deal with large amounts of noisy and non-linear data, with a high degree of uncertainty and a random nature [58][59][60].

Price time series prediction techniques are classified into two main categories: techniques that try to predict the actual value of the rate exchange or the actual returns value and techniques that try to predict the trend direction (uptrend, downtrend and sideways) [61]. Then there are several the different types of Machine Learning models that can be used. Neural Networks and Support Vector Machines became a popular choice, but there are many other models used for forecasting such as Bayesian Networks, k-nearest neighbors or genetic algorithms [62].

There are however some challenges in using Machine Learning for trading. For instance, the mere act of attempting to select training and testing sets introduces a significant amount of bias (a data selection bias) that creates a problem. Then, there is the feature selection and the dimensionality problem. Principal Component Analysis is used to decompose a multivariate dataset in a set of successive orthogonal components that explain a maximum amount of the variance. This dimensionality reduction technique is very useful when dealing with an high-dimensional data set. Recursive Feature Elimination helps identify relevant features in the spectra, given an external estimator that assigns weights to features. This feature ranking technique recursively considers smaller and smaller sets of features. First, the estimator is trained on the initial set of features and weights are assigned to each one of them. Afterwards, features whose absolute weights are the smallest are pruned from the current set features [63].

Measuring algorithm success is also a challenge. Inevitably the machine learning algorithms used in trading strategies should be measured in merit by their ability to generate positive returns and high sharpe ratios but before returns are considered Hit Rates should be high enough. If the Hit Rate of a model is close to 50% its predictions may have a random nature, which make them not suitable and even detrimental to a trading strategy. Also, even correct predictions do not necessarily equal profitable trading, other variables such as transaction costs, position sizing or the entry/exit signals timing may have direct impact on returns.

CHAPTER 3

Backtesting and benchmarking

In this chapter it is elaborated the concept of backtesting and its pitfalls. In addition it is explained how to prepare data for a backtest and how to assess a learning algorithm prior to the backtest. Also, there is a section explaining how to benchmark trading strategies, with focus in explaining some metrics that can be used to achieve this goal.

3.1 Data processing

When backtesting there is the need for high quality historical pricing data, in other words data without gaps and wrong values. The storage method is also relevant, historical data should be stored in a format that allows high degrees of access and good performance. Commonly historical pricing data from vendors is prone to many forms of error:

- **Corporate Actions:** - Incorrect handling of stock splits and dividend adjustments.
- **Spikes:** - Pricing points that greatly exceed certain historical volatility levels.
- **Missing Data:** - Lack of trades in a particular time, trading holidays or errors in the exchange system.

In order to mitigate the existence of Spikes in a dataset it is necessary to set a threshold in which a data point may deviate from the mean. In order to calculate this threshold the Standard Deviation may be used. Missing data errors can be fixed by padding or interpolating data points. Padding, which means filling missing data points with the previous value, tends to be the better choice because it doesn't incur in a Lookahead bias, contrary to interpolation [64].

For the retail algorithmic trader or small quantitative fund the most common data sets are end-of-day and intraday historical pricing for equities, indices, futures and foreign exchange [37]. Depending on the market, the data model may change. In this cases the most common data model is the Open-Low-High-Close that usually has 7 fields, including also Volume:

- **Day:** - Day of period
- **Time:** - Time at the beginning of each period

- **Open:** - First value in each period
- **Low:** - Lowest value in each period
- **High:** - Highest value in each period
- **Close:** - Last value in each period
- **Volume:** - Market volume in each period

In a OHLC dataset each row represents the price behaviour during a time period. For intraday trading the most common periods are 1 minute, 5 minutes, 15 minutes, 30 minutes, 1 hour and 4 hours. For interday trading, periods such as 1 day, 1 week and 1 month are of common use. Candle charts are used to represent all OHLC values, while the common line charts usually represent the closing prices. In markets like FOREX, Volume tends to be disregarded for not representing the true market Volume, it simply implies the number of units that price moved in that given time frame.

Tick data models are also of common use. Tick can be a measure of the minimum upward or downward movement in the price of security or the change in price of a financial asset from trade to trade. For example the minimum tick size for stocks trading above 1$ is 1 cent [65]. Tick data can have different shapes. In the FOREX market, for example, data includes 3 values, the Bid value, the Ask Value and the respective timestamp. The bid value represents the value at which the investor can buy units of an asset at a given time. The ask value represents the value at which the investor can sell units of an asset at a given time. This type of data is usually offered by brokers and is venue dependent. The amount by which the ask price exceeds the bid price for an asset in the market is called the bid-ask spread [66]. Depending on the spread offered, the same asset price may have different Bid and Ask value in different brokers and even clients.

As for storage formats there are 3 main ways of storing financial data [37]. Flat files(for example CSV files), NoSQL databases(Document oriented databases), Relational databases(Make use of the relational model to store data) CSV files offer easy access to data, NoSQL databases allow different schemes of data and Relational databases provide good performances in large datasets.

Survivorship bias should also be considered during the process of assessing data. Chan(2013) argues that in the absence of such survivorship bias–free data, backtesting should be limited to only the most recent, say, three years of historical data to reduce the damage [45].

All data to be used in backtesting should be divided in training and test datasets. Training data, also called in sample data, should be 60%/70% of all data, test data or out of sample data should be the rest of the data [45].

3.2 ASSESSING A LEARNING ALGORITHM

It has became common for trading strategies to include learning algorithms to perform a number of predictions. Before using a learning algorithm or any kind of ensemble of algorithms there is the need to assess their accuracy. How to assess a learning algorithm depends on its type and objective. In this work the focus will be on analysing Supervised Learning algorithms, which are the most commonly used for predicting financial prices or spreads.

The first step in this process is aggregating all historical data to be used for feeding the algorithms and slice it in a number of chunks. This process is called Cross Validation. The number of chunks generated would depend on the size of the original dataset, but the goal is to have at least two slices, one with in-sample data for training and other with out-of-sample data for testing. If, for example, there were 10 chunks of data, 3 could be selected as test data and the other 7 as training data in order to run a trial. Running more trials would only imply changing the chunks being used as training and test data. This allows performing several trials over the original dataset of historical data. However in the case of financial data, data is constrained by time. Testing with data from January 2017 after training with data from February 2017 would imply peeking into the future. So with financial data, instead of just using Cross Validation, Roll Forward Cross Validation should be implemented. Roll Forward means all training chunks are always from previous time periods than testing data chunks.

Root Mean Square (RMS) error helps measure the error between a model and the input data points. The more RMS error a model has, the less accurate it is.

$$RMS = \sqrt{\frac{\sum (Ytest - Ypredict)^2}{N}} \tag{3.1}$$

where *Ytest* represents one input data point, *Ypredict* is the respective prediction point and *N* is the number of data points. If data is divided in training and test slices it is expected that in-sample data will have a lower RMS than the out-of-sample data [64]. Another way of studying the error in a model is to check the correlation between predicted values and the real values. Plotting a scatter plot with both datasets in different axis helps visualizing the problem.

In Machine Learning if the knowledge and data available is not sufficient there is the risk of Overfitting. Overfitting can be understood by decomposing the generalization error(out of sample error), into bias and variance [67]. Bias is the learner's tendency to consistently learn the same wrong thing and variance is the tendency to learn random things irrespective of the real signal. Depending on the model and data, overfitting can have different causes [68]. In Machine Learning for Trading course offered by Georgia Tech, Balch argues that in the case of parameterized polynomial models one could check for overfitting by analysing in sample error and out of sample error [64]. By calculating the two errors successively with an increasing polynomial degree it is possible to detect overfitting when both errors start to diverge, for example if in-sample error continues dropping and out-of-sample error starts increasing [64].

The analysis of the confusion matrix(also known as error matrix) could also be useful, in a supervised learning algorithm(In the case of Unsupervised learning it is usually called a matching matrix). For instance, in a binary classifier context, in which the aim is only to predict if the market is trending up or down, if the matrix shows high error in predicting up trends but low error on the down trend prediction, the model could be considered only to confirm down trends. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class. Confusion matrix in a binary context:

$$\begin{bmatrix} TruePositives & FalseNegatives \\ FalsePositives & TrueNegatives \end{bmatrix} \tag{3.2}$$

Finally there is the concept of precision and recall. Precision is the fraction of well predicted instances among all predicted instances. Recall is the fraction of all predicted instances over the total of relevant instances. Both are very useful in the task of assessing a model's performance. In the context of Binary Classification, the precision given by the prediction score can show if the classifier is making random predictions or if in fact has any forecasting power.

All the notions given in this section can be implemented easily in Python with the scikit-learn package. The matplotlib library can be also used to plot the graphs needed for analysis.

3.3 PITFALS OF BACKTESTING

The performance of a strategy is usually very sensitive to details, and small changes on these details can result in substantial improvements. For instance in a Mean Reverting strategy, changing the period of the moving average change dramatically the backtest output. Backtesting any strategy is prone to errors, some types of errors are very common and are generally applicable to all markets.

Look-ahead Bias is an example of a common mistake made in the backtesting process. It happens when tomorrow's prices are used to determine today's trading signals. The solution is to use lagged historical data for calculating trading signals, meaning that all quantities are based on data up to the close of the previous trading period only [37].

Survivorship Bias happens when backtesting a stock-trading model if the historical data does not include delisted stocks, meaning they contain only stocks that survive until today. This bias can make backtest results look excellent. Buying a stock when it was beaten down badly, but subsequently survived, could result in good fictional results, although when live-trading the strategy this could not have been predicted.

Data-Snooping Bias is another common error. This error occurs when backtest performance is inflated relative to the future performance of a strategy because of the existence of overfitting parameters based on transient noise in the historical data. In order to mitigate this error tests should use out-of-sample data. Out-of-sample data, as stated before, means the original dataset is divided in two parts, training data and test data. Strategy parameters should be optimized using the training data and only the output of tests using test data should be taken into account. In case of a bad performance result, the strategy should be re-written and not optimized on the test data until overfitting occurs again [9].

Different financial brokers tend to offer different spreads, commissions and slippage. There is no rule that says a trade executed at one venue has to be at the best bid or ask across all the different venues. As a result, the transaction costs are also highly venue dependent and need to be taken into account in a backtest [45].

3.4 BENCHMARKING STRATEGIES

Benchmarking is used to measure performance using a set of specific indicators. In the case of trading algorithms/strategies these indicators are obtained by backtesting. The backtesting process in trading should follow the "scientific method". It should start with a hypothesis about an arbitrage opportunity, maybe based on the intuition about the market or from some published research. Then, confirm or refute this hypothesis by a backtest. If the results aren't good enough, the hypothesis should be modified and the process repeated [45].

Chan(2009) argues that Sharpe Ratio and drawdowns are the two most important indicators to determine performance [9]. Sharpe Ratio should be high, maximum drawdown and drawdown duration should be low. He also argues that, by his experience, when backtesting with one-minute data the dataset should contain at least 7 months worth of data [9].

Drawdown is the peak-to-troug decline during a specific recorded period of returns. It is usually quoted as the percentage between the peak and the subsequent trough [69].

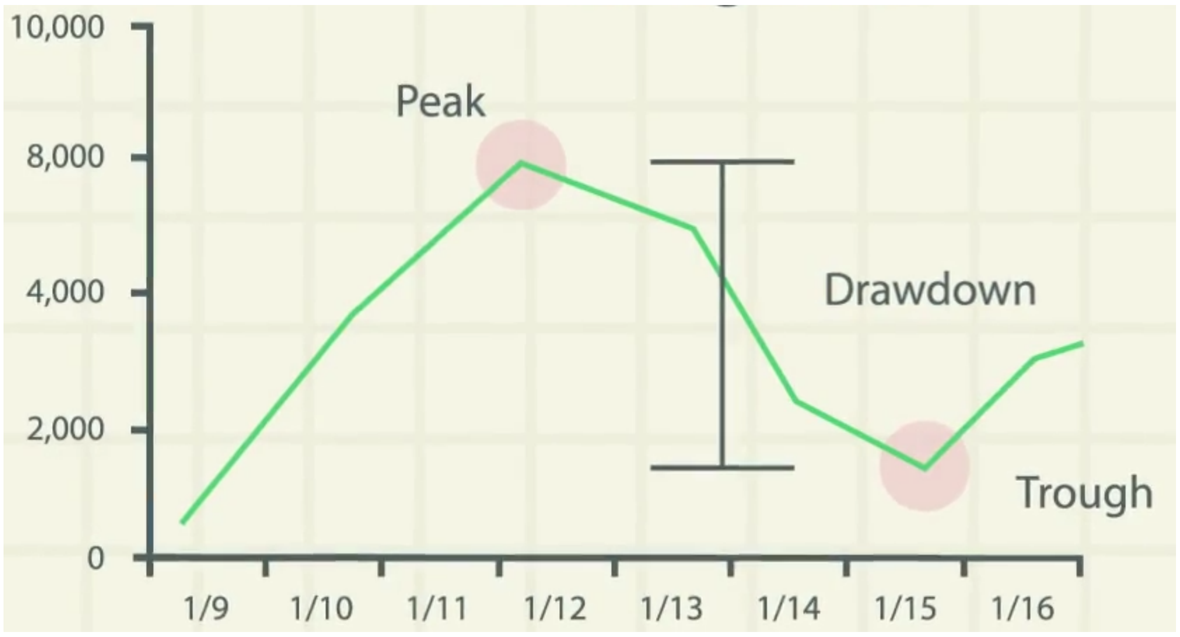


Figure 3.1: Visualization of drawdown[69]

The Drawdown Duration is the length of any peak to peak period and the Maximum Drawdown Duration is the worst amount of time between peaks, as shown in figure 3.2.

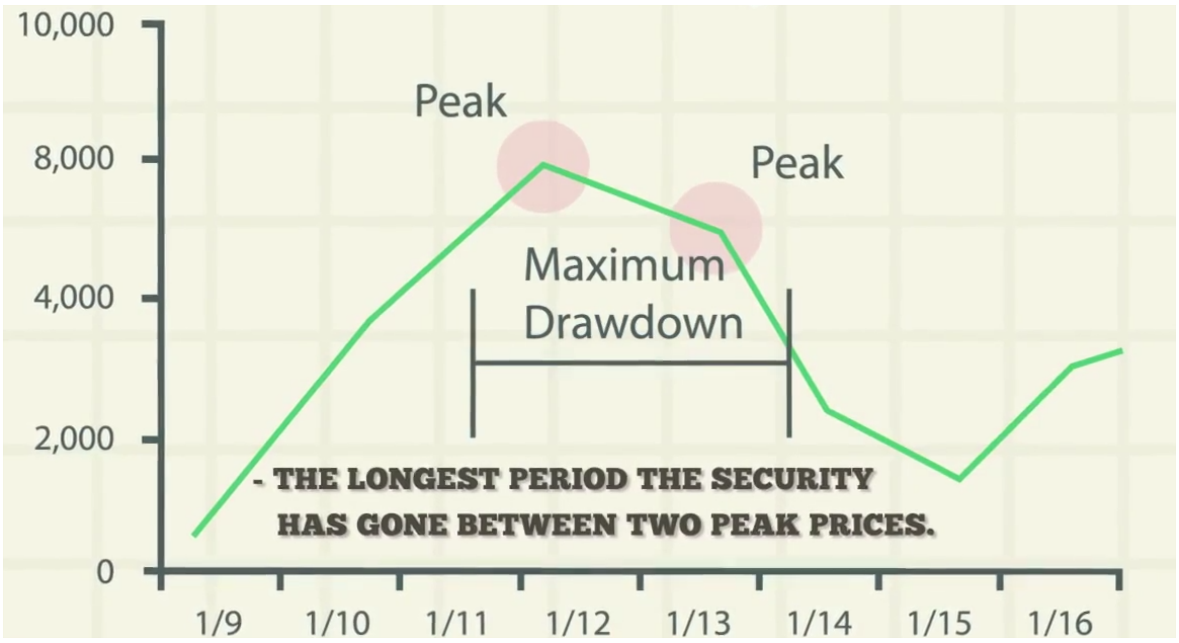


Figure 3.2: Visualization of maximum drawdown[69]

The Sharpe ratio heuristically characterises the reward/risk ratio of the strategy. It quantifies how much return can be achieved for the level of volatility endured by the equity curve [70]. In the case of two strategies producing the same returns, the sharpe ratio would help assess which one contains more risk. Sharpe Ratio S $<$ 1 should not often be considered. However, there are exceptions to this, particularly in the trend-following space. Quantitative funds tend to ignore any strategies that possess Sharpe ratios S $<$ 2. The Sharpe ratio will often increase with trading frequency. These strategies rarely suffer from great losses and thus minimise their volatility of returns, which leads to such high Sharpe ratios. However high-frequency strategies such as these can simply cease to function very suddenly, which is another aspect of risk not fully reflected in the Sharpe ratio [37].

The Sharpe ratio formula is:

$$S = \frac{R_p - R_f}{\sigma_p} \tag{3.3}$$

where R_p is the averaged returns, R_f the risk-free rate and σ_p the standard deviation of returns.

For market-neutral strategies, there is a particular complication regarding whether to make use of the risk-free rate or zero as the benchmark. The market index itself should not be utilised as the strategy is, by design, market-neutral. The correct choice for a market-neutral portfolio is not to subtract the risk-free rate because it is self-financing . Since a credit interest, R_f, is gained from holding a margin, the actual calculation for returns is:

$$(R_p + R_f) - R_f = R_p \tag{3.4}$$

Hence there is no actual subtraction of the risk-free rate for a currency neutral strategies.

In light of this, the formula used to calculate the annualised Sharpe Ratio was:

$$S_A = \sqrt{nperiods} \times \frac{R_p}{\sigma_p} \tag{3.5}$$

where *nperiods* represents the number of trading periods in one year, for example, it can be 252 for daily periods.

CHAPTER 4

System Description And Architecture

This chapter introduces the system description, architecture and how its development was conceptualized.

4.1 DESCRIPTION AND REQUIREMENTS

This section provides an in depth description about the backtest and live trading system. The main objective of this dissertation is to create a system that allows both investors and researchers to perform performance analysis over a range of trading strategies. Moreover it was also important to create a system that allowed live trading. In order to be able to fully create, improve and test trading strategies, both capabilities must exist. Backtesting only goes so far because, as explained before, transaction costs are venue dependent. Only by live trading strategies in a particular venue one can truly assess their performance. A system with this characteristics has to guarantee a considerable set of requirements:

- The system must allow both basic conditional trading algorithms, but also hardware demanding code. Machine learning and other forms of statistical analysis may be used to produce trading signals.
- Besides having algorithms that produce buy/sell signals, money management should be taken into account in any trading strategy. The system must then grant portfolio management capabilities.
- A toolkit of financial technical indicators is mandatory in a trading system.
- Non trivial trading strategies may require the previous study of historical price series. The system is required to offer tools like the Hurst exponent or the ADF test for the study of price series.
- Prior to building a strategy that leverages on learning algorithms there is a search for the right model and respective hyper-parameters that achieves a reasonable forecasting accuracy. This search should be contemplated in the system.

- Several data storage formats should be considered. Accessibility to data is an important requirement.
- The OHLC and the bid/ask data model have to be contemplated in the system in order to provide compatibility with the maximum number of data sources.
- Trading strategies must have access to different quantities of historical data of different instruments.
- The system should provide an interactive visual backtest report to allow further analysis and improvement.
- Truly independence from brokerage services is impossible, but the system must be able to function with different venues with minimal changes.
- Depending on technology constraints, execution time is an issue. The backtest system should be optimized in order to reduce execution time.
- In order to benchmark different trading strategies, performance metrics should be calculated, stored and compared.

4.2 EVENT-DRIVEN ARCHITECTURE

In order to comply with the requirements, during the design phase, a modular event-driven architecture was chosen. Event-driven systems are widely used in software engineering, commonly for handling GUI input, they are also ideal for algorithmic trading [37]. Considering a situation where the automated trading strategy is connected to a real-time market feed. New market information will be sent to the system, which triggers an event to generate new trading signals and consequently an execution event. Therefore such a system is in a continuous loop waiting to receive events and handle them appropriately.

Four modules were projected for the system. They are :

- **Portfolio** - Responsible for managing risk by assessing trading signals and tracking all open positions
- **Strategy** - Produces trading signals using historical data
- **Data Handler** - Aggregates and processes data, providing it to the Strategy
- **Execution Handler** - Executes trading orders, calculates transactions costs and slippage

To be able to work as a discrete event simulator, the system needed an engine to process all the events generated by the modules, this engine was called Backtest. This structure was used as groundwork for the backtest and live trading system. In order to allow live trading, more modules were added. Table 4.1 describes their usage.

Module	Usage
Porftolio	Backtest and Live Trading
Data Handler	Backtest and Live Trading
Data Gatherer	Live Trading
Strategy	Backtest and Live Trading
Execution Handler	Backtest and Live Trading
Data Storage	Live Trading

Table 4.1: Usage of modules

In figure 4.1 is presented the high-level projected architecture of the system, with all the components referred throughout this chapter. It shows the interaction between modules and external services.

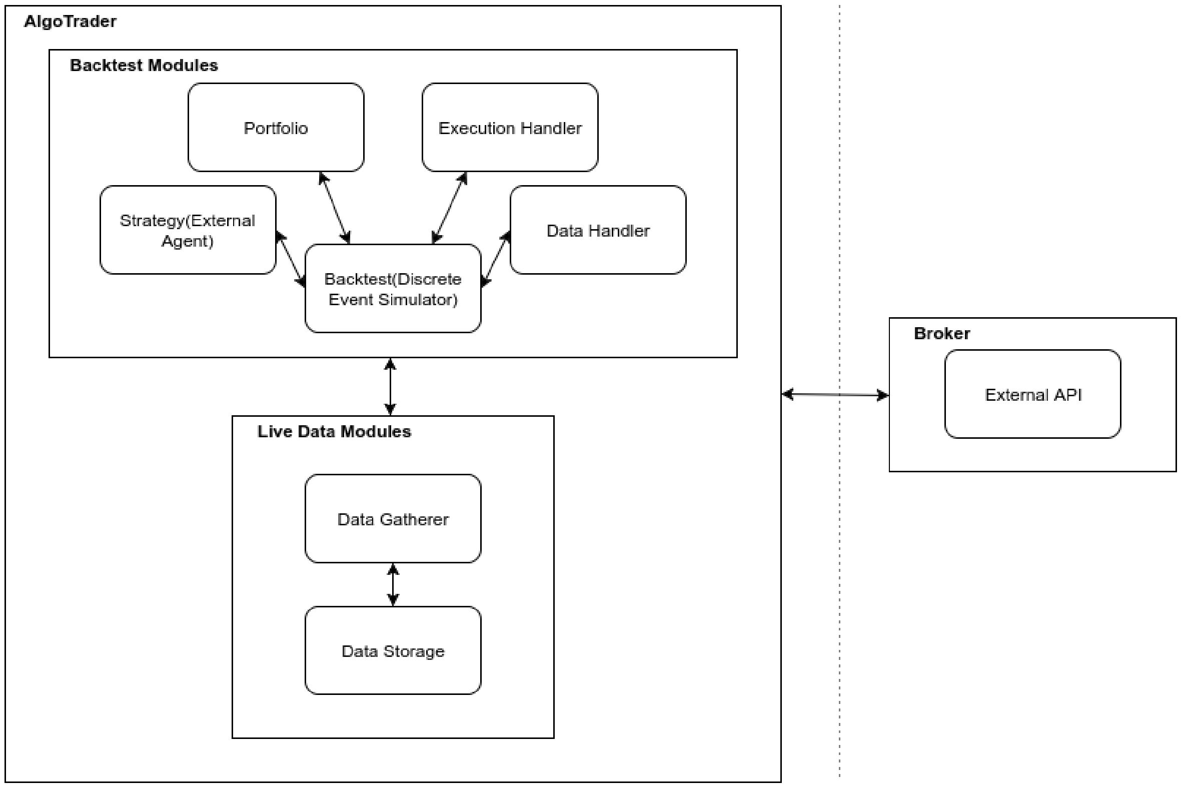


Figure 4.1: High-Level System Architecture

The modular nature of the Strategy module also allows the existence of more than one strategies producing signals for the Portfolio to analyze. Having this possibility together with the signal strength value, present in every Signal Event object, creates a new dimension for backtesting. Not only individual strategies can be backtested, but ensembles of strategies can be tested for their capacity to produce profits in a sustainable manner.

A modular event-driven system has two main advantages, the first is code reuse. By design the system allows backtesting with live data or historical data. By changing two modules, the Data Handler, from a historical data handler to a live data handler, and the Execution Handler, from simulated execution handler to a live execution handler, the system can be used for live trading. Another advantage is the prevention of Lookahead Bias. This bias, as stated before, occurs when tomorrow's data is used to determine today's trading signals. Using an event to model the data tick arrival allows the event-driven system to operate only inside the time constraints and avoid this bias.

As this type of system helps mitigate pitfalls associated with backtesting it also has its disadvantages. An event-driven solution is slower in contrast with a vectorized solution and it can be more demanding for the processing unit.

4.2.1 EVENTS

The event-driven nature of the system demands the use of different type of events in order to simulate the backtest process. There are four types of events to consider, in figure 4.2 events and their respective generator modules are shown.

First is the Market event that can be triggered once every heartbeat or when the Data-Handler object receives a new update of market data. The heartbeat is the period during which the system waits before requesting new data from the Data Handler. Each Market Event triggers the Strategy object responsible for generating trading signals in the system. Its object only contains the event identification as no further information is needed.

The Strategy object returns trading signals using Signal Events, where information about the signal is stored. A Signal Event contains the id of the strategy responsible for its creation, signal strength, signal direction, ticker symbol and a timestamp for when it was generated.

The third type is the Order Event that is generated by the Portfolio. Taking into account portfolio metrics such as available cash or risk and the signal strength, the Portfolio object decides if an Order Event should be created. This event can activate the creation of a real order by the Execution Handler object in a live trading context. This object receives the quantity to order, order direction, order timestamp and ticker symbol, all contained in the Order Event.

The fourth type of event is the Fill Event. The goal of the Fill Event is to help the system take into account brokerage costs and slippage. When the Execution Handler receives an Order Event waits until the order is transacted and creates a Fill Event. The newly created event has a timestamp, a ticker symbol, broker id, quantity transacted, slippage, type of transaction and direction of transaction.

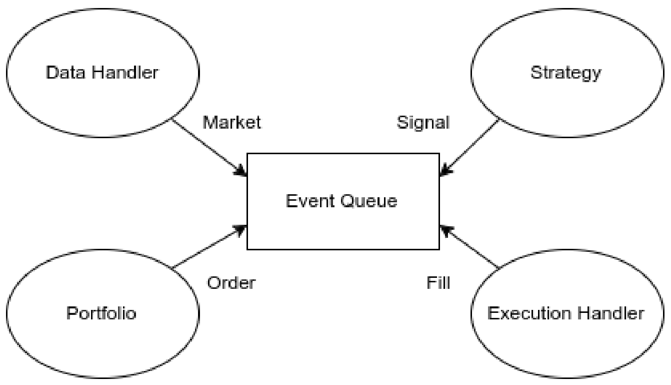


Figure 4.2: Event creation modules

4.2.2 MODULES

The DataHandler as a key module of the event-driven system must provide price data to the Strategy object in order to generate trading signals. The data origin can be a live stream from the Data Gatherer module, a historical data CSV file, an external API(ex. Quandl) or a TXT file. Independent to the data origin, all the Data Handler objects have a fixed set of methods for data manipulation to allow a certain degree of flexibility to the Strategy object. For example, in a Historical Data context, after loading all the data, the Data Handler object must be able to return the latest x rows of data, the latest x values of a chosen price variable or any time indexes.

In a live data context the Data Handler object must communicate with the Data Gatherer module each heartbeat, requesting the latest price values. All the prices received from the Data Gatherer must be stored, processed and made available for later use.

In finance terminology portfolio represents the grouping of financial assets being managed by investor or funds. In order to decide which assets to include in the Portfolio and the quantity they should be traded, risk tolerance and investing objectives should be taken into account. Current market positions and their current market values are called holdings. Depending on investment objectives and current holdings different rules can be set to prevent the increase of risk, this practice is called risk management. This is why a Portfolio module is essential in the backtest system, the module tracks and manages open positions in financial markets and takes use of methods, such as the Kelly Criterion, to tackle risk management and position sizing. In the case of position sizing the strategy type is also taken into account, strategies like Pairs trading may need the hedge ratio value taken into account in the sizing decision in order to be profitable. The Portfolio object receives Signal Events and Fill Events. Each signal is processed and may generate an Order Event depending on the current holdings and established rules. Fill Events are used by the object to update holdings and keep track of all transactions.

As previously stated the Portfolio object is responsible for creating Order Events which the Execution Handler object will use to create Fill Events and in the case of live trading to generate trade requests to the Broker API. Before this happens Signal Events must be generated, sending information to the Portfolio about the asset to be traded, the strength of the signal and the direction to trade.

The Strategy module is responsible for generating this Signal Events. It requests market data, for each asset to be traded, every heartbeat and makes calculations over it. This calculations will result in indicators that will be taken into account in the process of generating trading signals. In the case of high frequency trading these calculations must be performed in the smallest amount of time possible, adding a high a degree of dependence to hardware. Every Strategy object has its own set of custom indicators and based in their values it may generate Signal Events. It also has rules that aim to improve profitability for instance, in the context of a simulation, a "stop loss" can be directly implemented in the strategy logic in order to prevent further losses on a losing trade. The concept of a limit order can also be materialized inside the Strategy object without increasing complexity of the system. The modular nature of the Strategy module also allows the existence of more than one Strategies producing signals for the Portfolio to analyze. Having this possibility together with the signal strength value, present in every Signal Event object, creates a new dimension for backtesting. Not only individual strategies can be backtested, but ensembles of strategies can be tested for their capacity to produce profits in a sustainable manner.

An investor can have different trading accounts, for different purposes, in different brokerage companies. The brokerage companies are, in some situations, the gateway for the financial markets and in other cases they create the market themselves . In a trading system it is necessary to have an entity responsible for converting trading orders to brokerage API trade requests. The goal of the Execution handler object is to serve as middleman between the trading system and the brokerage service. In a live trading context, upon the arrival of Order Events, this module is responsible for generating the appropriate requests to the desired brokerage account. Also, it provides methods to calculate transaction costs, slippage, account balance and open trades. Finally it must create a Fill Event describing the filled order so the system can keep track of open positions and be able to update the holdings and cash available. In the case of a backtest simulation, the Execution Handler only has to create the Fill Event. Transaction costs such as commissions and spread can be introduced in order to provide a more accurate backtesting.

Live trading is both a necessity in the process of assessing the profitability of a strategy and the objective when profitability is assured. Therefore it was of the utmost importance to implement a live trading solution in this work.

Due to the event-driven and modular nature of the backtest system Live Trading can be easily added. The first step is to create a Data Handler module capable of handling live tick data. This data contains a Bid value, an Ask value and a timestamp. Next is the Execution Handler. It is necessary to have an Execution module that can serve as middleware between the backtest engine and Broker APIs. Besides creating alternative modules for the Backtest Engine it is necessary to create mechanisms for aggregating and storing data. Hence the Data Gatherer and Data Storage modules creation.

The Data Gatherer module is only used in live trading, its goal is to provide a stream of live data to the Data Handler and store data for further research. The module is then responsible for connecting three entities: the Data handler object, the Data Storage object and the brokerage account.

There are several entities that tap into live data sources, store and clean the data in order to sell it later to researchers. It is only logical that the backtest system includes a module for storing live data obtained via Data Gatherer.

4.3 OANDA FOREX BROKER

When building the backtest system the goal was to make a system independent from any particular broker, however without collaboration from brokers it is impossible to achieve full independence from the brokerage service. Metatrader[1], for example, achieves this by offering a server solution to brokers and a client solution to investors. The server serves requests for live prices, historical prices and trading orders.

In order to tackle the communication with the broker issue, it was created a middleware module, the Execution Handler, that transform system orders into Broker trade requests. Broker requests are generated by Wrapper classes, each Broker has its own wrapper for its respective API. As shown in figure 4.3, both the Live Trading modules and the Backtest Modules can request data through the Wrapper class. For example, it can be useful to request market information so the Portfolio can better manage risk. Also, of course, there is request of the price data needed by the Data Handler.

There are only some brokerage companies that offer API's, the Oanda Brokerage service is one of them and it was chosen as the use case. This choice was made taking into account the:

- Accessibility and flexibility of service
- Live Stream of data through HTTP Stream
- Well documented API with a wide range of endpoints
- Offer of "paper trading"
- Offer of macroeconomic and market information, for example the economic calendar

[1]https://www.metaquotes.net/

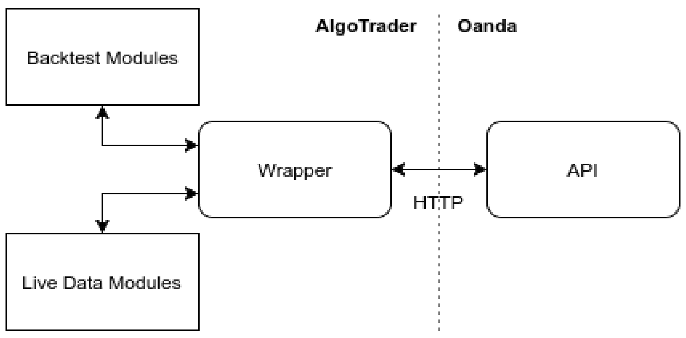


Figure 4.3: Diagram describing interactions with the Oanda Brokerage Service

CHAPTER 5

Implementation

In this chapter it will be detailed how the proposed system was implemented, how each of the system's requirements were met and how each component was orchestrated to communicate with each other.

5.1 BACKTEST ENGINE

In this section it will be given an overview of how technologies were used to implement the backtest engine. Also it will be explained how the different modules that make the system interact with each other.

Due to the event-driven nature of the backtest engine, a loop to handle events throughout test execution was created and can be seen below in algorithm 1. This algorithm was based on the work of Ernest Chan and Halls-Moore [9][45][37]. In line 3 it's possible to see how data is feeded to the system. Either in a simulation context in which data is drop fed or in a live trading context in which data arrival can be periodic or aperiodic. When data is updated in the Data Handler, Market Events are generated and added to the event queue, which is shared by all modules. If there are events in the event queue, depending on their type different modules will act. If it's a Market Event the Strategy Module may produce Signal Events and the Portfolio updates all the records with the latest price data timestamp, re-calculating all Portfolio metrics. In the case of a Signal Event, the Portfolio decides if an Order Event should be produced. When an Order Event is issued, the Execution Handler is responsible for either executing it to a broker and generating a Fill Event, or in a simulated context, by only generating a Fill Event for the simulated order. With the arrival of a Fill Event the Portfolio updates all holdings. If there are no events in the event queue the system sleeps for a period set by the heartbeat. The heartbeat is used to achieve periodic trading.

Algorithm 1 Main loop in backtest engine

```
1: while True do
2:     if There is new data then
3:         Data Handler updates data
4:     else
5:         Break
6:     end if
7:     while True do
8:         if Event queue is empty then
9:             Break
10:        else
11:            if Market Event then
12:                Strategy calculates signals
13:                Portfolio updates market value
14:            else if Signal Event then
15:                Portfolio analyzes signal
16:            else if Order Event then
17:                Execution Handler executes order
18:            else if Fill Event then
19:                Portfolio updates holdings
20:            end if
21:        end if
22:    end while
23:    Sleep(Heartbeat)
24: end while
```

5.1.1 ARCHITECTURE

The implemented architecture can be seen in figure 5.1. Although beyond the scope of this work, communication with an instance of the Metatrader 4 platform was established. An expert advisor was created to receive orders from the Execution Handler and provide price data via Data Gatherer. All the communication with the platform was also achieved with the ZMQ[1] library with a Publish/Subscribe communication pattern.

[1]http://zeromq.org/

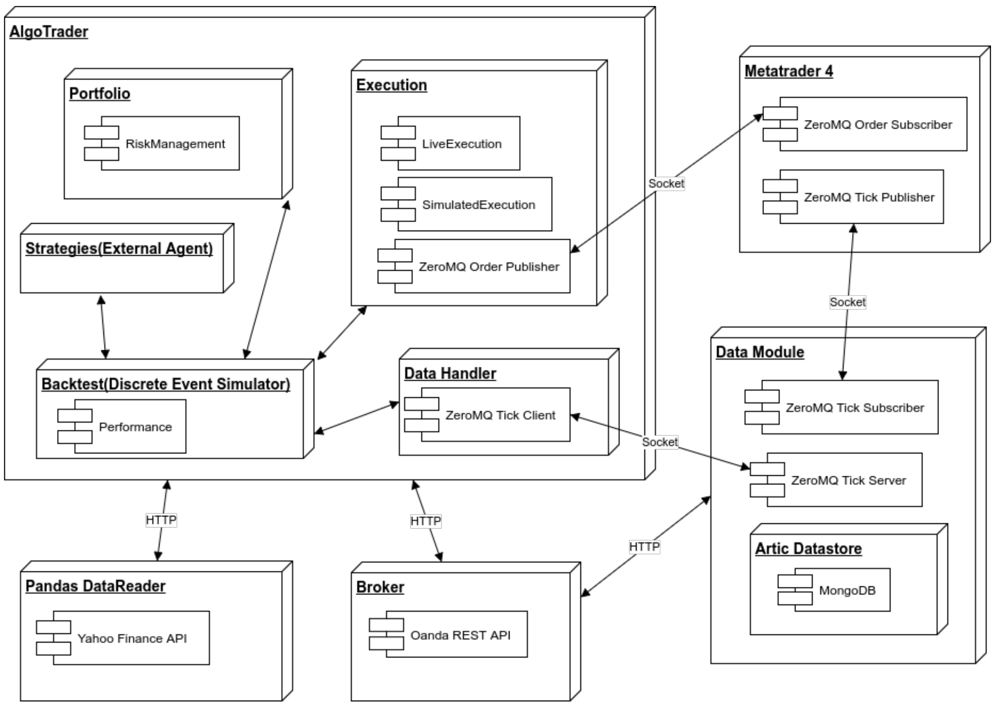


Figure 5.1: Implemented architecture

The developed code was made available at Github on the domain https://github.com/mglcampos/algotrader. There it is possible to find the backtest and live trading system. Also, all the strategies developed and methods for statistical analysis and machine learning are available.

5.1.2 MODULES

The backtest engine modules were implemented using Python's object oriented programming. Each module is implemented as an abstract base class, providing significant flexibility.

The Data Handler base class was implemented with the following methods:

Code 1 Data Handler main methods

```
def get_latest_bar(symbol):
"""
Returns the last bar updated.
"""
def get_latest_bars(symbol, N=1):
"""
Returns the last N bars updated.
"""
def get_latest_bar_datetime(symbol):
"""
Returns a Python datetime object for the last bar.
"""
def get_latest_bar_value(symbol, val_type):
"""
Returns one of the Open, High, Low, Close, Volume or OI
from the last bar.
"""
def get_latest_bars_values(symbol, val_type, N=1):
"""
Returns the last N bar values from the
latest_symbol list, or N-k if less available.
"""
def update_bars():
"""
Pushes the latest bars to the bars_queue for each symbol
in a tuple OHLC format: (datetime, open, high, low,
close).
"""
```

When a Data Handler object is instantiated data is loaded to a pandas dataframe from a data source. Depending on the data source several subclasses were created to serve the needs. The following data handling subclasses were created:

- **CSVDataHandler** - Handles data stored in CSV files
- **YahooDataHandler** - Handles data from the Yahoo API
- **MongoDataHandler** - For data stored in a MongoDB instance.
- **LiveDataHandler** - Handles data requested from the Data Gatherer module

The CSVDataHandler used pandas read_csv() to read CSV files, YahooDataHandler worked using the pandas-datareader package that includes connection to Yahoo Finance API, MongoDataHandler used pymongo to connect and fetch data from the MongoDB instance and the LiveDataHandler used ZMQ to request data from the Data Gatherer module.[71][72][73]

In a Historical Data context, the whole data set is loaded to a pandas dataframe called "symbol data". In order to simulate a live feed of data the object appends one row of data to a second dataframe called "latest bars" each time a Market Event occurs. The DataHandler object can then return the latest x rows of data or the latest row datetime with the getter methods in code 1. Using the "latest bars" dataframe as data source for the Strategy object prevents one of the major pitfalls in backtesting, the Lookahead Bias.[37]

In a live data context the Data Handler object functions as a client in a client-server architecture requesting the latest price values from the server, the Data Gatherer module, each heartbeat. Code 2 shows how the ZMQ client was implemented. All the prices received from the server are stored in a dataframe accessible to the Strategy object via the set of data manipulation methods.

Code 2 ZeroMQ client implementation on the LiveDataHandler

```
import zmq

def startListener():
        context = zmq.Context()
        self.socket = context.socket(zmq.REQ)
        self.socket.connect("tcp://127.0.0.1:5558")

def update_symbol_data(symbol_list):
        if self.socket is not None:
                for symbol in self.symbol_list:
                        request = 'tick'
                        message = "{0} {1}".format(request, symbol)
                        self.socket.send(message)
                        response = self.socket.recv()
```

All Strategy objects share the same structure and have at least one method in common, calculate_signals() which produces Signal Events. Also, Execution Handler objects must implement an execute_order method that processes Order Events.

5.2 LIVE TRADING

After the implementation of the backtest engine, live trading was added to the system. This was accomplished by creating the Data Gatherer and Data Storage modules.

In terms of technologies, two more were added to allow live trading: ZeroMQ and the Artic Datastore[2]. To make the best use out of ZMQ capabilities it was necessary to study its patterns, requirements and work the best possible configuration to fit the systems needs. ZMQ offers four types of communication patterns: PAIR, Client/Server, Publish/Subscribe and Push/Pull.

5.2.1 DATA GATHERER

In the use case of the Oanda Broker, data is provided with an HTTP stream.

The communication between the Data Handler and this module is guaranteed, as stated before, by the ZMQ asynchronous communication library. As shown in figure 5.2, Client/Server was selected as the communication pattern. This was to enable the Data Handler to set a period, upon which it requests new tick data. This period should be equal or lower to the

[2]https://github.com/manahl/arctic

system heartbeat if possible. In the case of a need to feed the system with aperiodic data, Publish/Subscribe can be used as the communication pattern, the Data Handler is configured as the Subscriber and the Data Gatherer as Publisher.

When the Data Gatherer starts running it launches two threads, one to establish connection with the Oanda's HTTP Stream the other to keep the ZMQ server listening for requests. The ZMQ server thread, upon the arrival of a message from the client, sends a message with the latest tick, its timestamp and correspondent asset ticker symbol. The HTTP stream thread receives data for the selected assets and stores them in a buffer. This buffer, with variable size depending on the heartbeat of the system, hardware capabilities and the Python language constraints, is emptied each time it gets full sending all its content to the database via Data Storage.

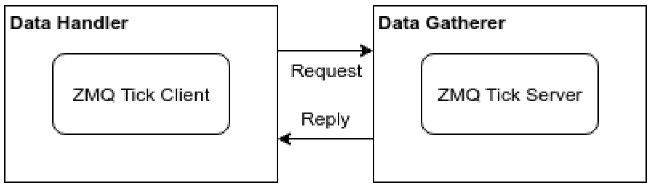


Figure 5.2: Diagram describing the communication pattern between the Data Handler and the Data Gatherer

5.2.2 DATA STORAGE

Live trading demands the use of live data in order to create trading orders. This live data in the form of a Tick, meaning it contains a timestamp, a Bid value and a Ask value, has significant value for later research and should be stored.

In order to do this, three technologies were used: a MongoDB database, the Artic timeseries and Tick store and Python's library Pandas. The Data Storage module uses the Artic datastore to save data in the form of Pandas Dataframes and Pandas Series in MongoDB collections. Artic includes three types of storage engines[74]:

- **VersionStore** - a key-value versioned TimeSeries store which supports pandas data types, multiple versions of each data item and a wide range of data frequencies.
- **TickStore** - Column oriented tick database, designed for large continously ticking data.
- **Chuckstore** - A storage type that allows data to be stored in customizable chunk sizes that aren't versioned and can be appended to.

The default storage engine is the VersionStore, and it was the engine used in this work. The choice was made taking into a account the development nature of this work and availability of storage space. In a production scenario, if there was no buffer, the HTTP stream could be directly connected to the TickStore. And if there is a large quantity of incoming data, Chunckstore offers the most "economic" option. In addition to allowing the storage of data, using pandas data formats, this datastore provides high performance for querying numeric data in a MongoDB instance.

5.3 TIME SERIES ANALYSIS AND MACHINE LEARNING

There are many shapes of trading strategies. The classic paradigm is divided in two parts, one that focus on fundamental analysis, on “which macroeconomic factors make the price drop” and in strategies that focus on technical analysis, on “which indicators allow a price drop prediction”. Still within this paradigm there are those who perform statistical analysis of price time series prior to the process of testing the hypothesis. Knowing the nature of an asset price series may help create the appropriate type of strategy for that particular asset. For example, if the price series of an asset is strongly stationary than a Mean Reverting strategy could result in reasonable profits. To help researchers study price series two types of studies were created and will be described in detail in this section. The first study uses the Hurst exponent and the ADF test to evaluate stationarity in a particular price series. There is also the study of cointegration between two or more price series.

Besides statistical analysis there are other forms of improving a Strategy performance. Machine learning became a popular tool to improve the quality of generated signals in a strategy. For example, a Mean Reverting strategy could benefit from a binary classifier that confirms trend reversal. Creating such a model, that receives historical prices and price generated indicators as features and predicts if the market is going up or down, with a reasonable degree of accuracy, is not a trivial task. It requires extensive study of techniques such as dimensionality reduction or feature extraction. In order to allow this, the system offers a set of different classifiers and meta-classifiers with a wide range of features.

5.3.1 STATISTICAL ANALYSIS

Statistical analysis is the science of collecting and interpret quantitative data to discover its underlying causes, patterns, relationships and trends[75]. In this work the importance of applying statistics to price data was recognised. There are several useful statistical studies than can be applied on price data, in this case the focus was on stationarity and cointegration. As stated in 2.8 stationarity is the degree upon which its statistical properties change over time. Also, cointegration measures whether or not the price distance between two or more assets remain stable over time.

To study stationarity and cointegration in price time series, two studies were created. One that applied the Hurst Exponent and the Augmented Dickey Fuller test on a time series to evaluate its stationarity. And a second one that calculates a residuals series between two assets and then evaluates its stationarity, which may or may not indicate cointegration between the two assets. The method implemented to calculate the HE, based on Tom Starke’s implementation published in the Quantopian Community Forum[76], is shown in code 3.

Code 3 Hurst Exponent implementation

```python
import numpy as np

def hurst(p):
        tau = []; lagvec = []
        #  step through the different lags
        for lag in range(2,20):
        #  produce price difference with lag
        pp = np.subtract(p[lag:],p[:-lag])
        #  write the different lags into a vector
        lagvec.append(lag)
        #  calculate the variance of the difference vector
        tau.append(np.sqrt(np.std(pp)))
        # linear fit to double-log graph (gives power)
        m = np.polyfit(np.log10(lagvec),np.log10(tau),1)
        # calculate hurst
        hurst = m[0]*2
        return hurst
```

The CADF test was implemented using statsmodels package.[3] In order to obtain the residuals time series, the same process explained in subsection 5.4.4 was performed. Then, the code shown in 4 was used to calculate the CADF result.

Code 4 Cointegrated Augmented Dickey Fuller test implementation

```python
import statsmodels.tsa.stattools as ts

def cadf(res):
        cadf = ts.adfuller(res)
        return cadf
```

5.3.2 LEARNING ALGORITHMS

To apply machine learning forecasting in trading successfully requires an extensive research over learning algorithms and statistical analysis. Besides it implicates a scientific approach for preparing data and selecting the right datasets. As explained in section 2.7.5, in Machine Learning there are different types of learning algorithms and each type has a number of models. Each type can be used for different tasks, Unsupervised Learning, for example, could be used to find structure in price data of different assets and Supervised Learning for forecasting up and down trends.

In this case the scope was to help tackle research in Supervised Learning, namelly in Binary Classification. In order to implement all machine learning models, feature selection and decomposition methods scikit-learn package was used. Scikit offers several classifiers and meta-classifiers. In this work the following models were implemented:

- **Linear Discriminant Analysis** - A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.

[3]http://www.statsmodels.org/devel/generated/statsmodels.tsa.stattools.adfuller.html

- **Quadratic Discriminant Analysis** - A classifier with a quadratic decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.
- **Logistic Regression** - Measures the relationship between a binary categorical dependent variable and multiple independent continuous variables.
- **Radial Support Vector Machine** - C-Support vector classification using a radial basis function as kernel.
- **Linear Support Vector Classifier** - Linear c-support vector classification.
- **Voting Classifier** - Soft Voting/Majority Rule classifier for unfitted estimators.
- **Bagging Classifier** - Ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions.
- **Random Forest Classifier** - Meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset

The RSVM, RFC, VC and BC were implemented as follows:

Code 5 RSVM, RFC, VC and BC implementation

```
import sklearn

svc = SVC(C=1000000.0, cache_size=200, class_weight=None,
        coef0=0.0, degree=3, gamma=0.0001, kernel='rbf',
        max_iter = -1, probability = False, random_state = None,
        shrinking = True, tol = 0.001, verbose = False)

rfc = RandomForestClassifier( n_estimators=1000, criterion='gini',
        max_depth = None, min_samples_split = 2,
        min_samples_leaf = 1, max_features ='auto',
        bootstrap = True, oob_score = False, n_jobs = 1,
        random_state = None, verbose = 0)

vc = VotingClassifier(estimators=[('lr', lr), ('rfc', rfc), ('svc', svc),
        ('qda', qda), ('lsvc', lsvc)], voting='hard')

bagging = BaggingClassifier(svc)
```

In order to create features, TA-Lib was essential in providing technical analysis indicators. Also, statsmodels offered statistical computations and pykalman the kalman filter.

The first step was data preparation, methods were created to import OHLC data, remove gaps with padding and create several lagged series. Depending on the number of lags desired there were created lagged series with the objective of giving "past prices" to the models. By lagging data, a machine learning model can have access to the previous n periods in each time slot, being n the number of lags. Moreover, all price data and indicators were lagged at least once. The reason for this was that the OHLC data model is a end-of-period model in which only at the end of each period one can retrieve the respective Open, High, Low and Close values without peeking into the future. So in a Supervised Learning context, Y is paired with a lagged X.

After obtaining lagged OHLC prices to use as features, several more were created, being the majority technical indicators from TA-Lib that used the non-lagged price data. This library offers Overlap Studies Indicators(Including indicators such as Bollinger Bands), Momentum Indicators(The RSI indicator for example), Volume Indicators(On Balance Volume for example) and Volatility Indicators(For example the Average True Range). Each technical indicator requires a time period in which to it will apply a computation, so based on the type of indicator several time periods were included. Technical indicators that included moving averages were also used to create additional features, calculated by dividing the price by its respective averaged indicator and subtracting one. These new features were used to reflect the distance between indicator and price as an alternative to just giving averaged prices to models. In addition, the Kalman Filter was used to produce a filtered price signal.

How many and which features to use in the models is a challenge for researchers. To tackle this issue RFE and PCA were implemented. RFE is used for feature selection by recursively considering smaller and smaller sets of features and PCA is a tool for dimensionality reduction.

In order to choose the right classifier for the Binary Classification problem in the Forecasting Strategy, three classifiers were further compared. They were VC, BC and RSVM. EUR/USD one minute OHLC price data was used to generate features, as described above. Also two types of indicators were considered, the Kalman Filter and the Bollinger Bands derivative calculated in 5.4.

The available data consisted in 6 years, ranging from 2012 to 2016. Each year represented a sample of 366000 data rows and each month 30500 rows. The data used was in OHLC format, having four columns. Tests were performed with one month of data and one year. Both tests complied with the rule of 3/4 of sample for training and 1/4 for testing.

The VC was chosen because it produces results by applying a voting system to predictions from diferent models. In this case it uses as estimators instances of LR, RFC, RSVM, QDA and LSVC. Also, BC was used because it fits instances of RSVM, each on random subsets of the original dataset, and then aggregates their individual predictions. Finally, the results from VC and BC were compared with predictions from one base RSVM. The results for this comparison can be seen in section 6.2.1.

5.4 STRATEGIES

Having a backtest engine, a live trading solution and historical data, for a group of selected financial assets, the next step would be to start designing and implementing a trading strategy. Several strategies were created to test the system, covering some of the most common types of strategies. In this section it will be explained how Trend Following strategies were implemented in subsection 5.4.2, Mean Reverting in subsection 5.4.3, Pairs Trading in subsection 5.4.4 and Forecasting with Machine Learning in subsection 5.4.5.

As explained before in section 2.7 , the objective of a strategy is to produce trading signals for the backtest engine to process. As the use case for this work was the FOREX market, all the strategies implemented open and exit signals with buy or sell direction. Besides all strategies share the same structure.

In order to backtest all the Strategies a few foreign exchange currency pairs were selected. EUR/USD, one of the most traded pairs in the FOREX market was chosen because it reflects the two largest economies in the planet. EUR/AUD was selected due to Australia's economy dependence on commodities. EUR/NZD was selected because of the cointegration between EUR/NZD and EUR/AUD price series, as shown by the statistical analysis results in section 6.2.2. Also, tests with EUR/CAD and EUR/GBP were performed.

5.4.1 STRATEGY STRUCTURE

The structure used for strategies has 6 phases and can be seen in figure 5.3. The first phase, a), shown in the diagram, is to request price data from the data handler. This data can be in OHLC or bid/ask form. The length of data requested is defined by the researcher and depends on the strategy requirements. The next phase, b), is used to create stop losses and limit orders. Stop Losses are implemented to prevent price changes that are larger than a certain threshold. So in phase b) trade prices (price of the instrument when the position was opened) are compared with current prices and, based on the difference, Close signals can be created in order to issue a stop loss. The same concept applies to trailing stops. In the case of Limit Orders the logic is different. If in a previous iteration the strategy's flag for limit orders was activated and the price change between iterations exceeded a certain value then a Open signal is generated. The signal direction depends on the price change being positive or negative. In any of this cases, Stop Loss, Trailing Stop or Limit Order, the algorithm skips to phase f), the creation of the trading signals. Otherwise it continues to phase c), when all computations involving the price will be performed. In this phase new price data can be used to train a learning algorithm to predict future price change. This prediction can be used as an indicator. Also, technical analysis indicators, such as Bollinger Bands or RSI can be computed. When Pairs trading this phase is used to calculate the hedge ratio and produce the residuals series. After generating all indicators required for the strategy, it's time to process them and decide if a trading signal should be generated. This happens in phase d). If the decision is to create signals, the algorithm goes to phase f) creates the signals with the desired direction(Buy or Sell) and type(Open or Close) and ends. Contrarily, it ends without producing any signal, although flags for limit orders may have been activated.

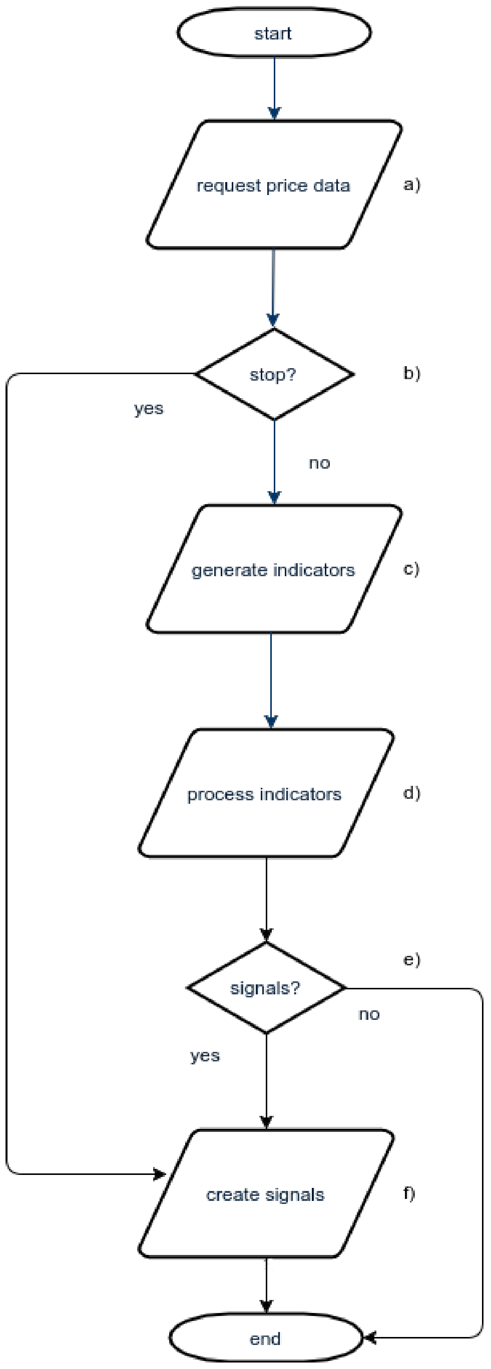


Figure 5.3: Diagram describing a trading algorithm

5.4.2 TREND FOLLOWING STRATEGY

Trend following strategies are among the most used by traders worldwide. As explained in section 2.6, these strategies rely on betting on trend continuation. If the price is trending up than the investor should go long, if the market is going down than the investor should short the trade.

To further understand this type of strategies, trend following strategies were implemented.

One example of a trading strategy is to use Momentum to establish trend strength and moving averages to interpret the trend. A fast and a slow Exponential Moving Averages (EMAs) were created with a rolling window of 5 and 12 periods. In order to track Momentum, the Relative Strength Index indicator was used with period of 21 time units. These periods were chosen based on the suggestions of the researcher Daniel Fernandez [77]. The equations for both indicators are shown in 5.1 and 5.2

$$RSI = 100 - \frac{100}{1 + RS} \tag{5.1}$$

where RS is the relative strength factor, a moving average. This moving average can be exponential moving average or an equally-weighted mean.

$$EMA_n = P_n \frac{2}{T+1} + EMA_{n-1}(1 - \frac{2}{T+1}) \tag{5.2}$$

where P is the price and T is the time period.

The algorithm implemented starts by requesting 21 time units of historical data(Time unit is equal to price data granularity). It computes the desired indicators using the TA-Lib[4] library on the available price data. Having the price generated indicators the next step is to assess their most recent values. This strategy is designed to only allow long trades, if most recent value of the Relative Strength Index (RSI) is more than 50, and short trades, if the the value is below 50. This momentum condition means that a long position will only be opened when more than 50% of the change of the last 21 bars of data has been towards the upside and vice versa. To find the right moment to enter the marker is necessary to check the slow and fast moving averages current values. If the fast EMA is higher than the slow EMA and the momentum condition is met than a open-buy signal is created. If the fast EMA is lower than the slow EMA and the momentum condition is met than a open-sell signal is generated. Also, the algorithm only allows 1 opened position at each time(Simplifies the strategy but prohibits Scaling In or Scaling Out). In order to exit the market the algorithm implements two conditions. The first is valid when the RSI value is higher or equal than 75 and closes long trades, The second is valid when the RSI value is lower or equal than 25 and closes short trades. In addition it includes one stoploss if the price is changed by more than 20 pips(0.0001 or 0.001 depending on the foreign exchange currency pair) and the change in price in the two most recent time periods is less than 4 pips. The for 4 pip threshold is used to avoid closing the trade immediately after a big movement in price, movements which tend to revert in the most cases providing a better closing position and reducing losses. In the foreign exchange market, most brokerage services don't offer trading on weekends. Thus, for preventing positions from staying opened through the weekend a system was put in place to prevent trades from opening on fridays after 3pm WET.

[4]http://www.ta-lib.org/

Algorithm 2 Trend Following Strategy algorithm

```
1: procedure CALCULATE SIGNALS
2:     for symbol in symbol list do
3:         Request price data for symbol
4:         if Current daytime is beyond trading period then
5:             end procedure
6:         else if Price activates stoploss then
7:             Generate signals
8:         else
9:             Calculate RSI21, EMA5, EMA12
10:            Assess values from indicators & check for open positions
11:            if Conditions applied to indicators are met then
12:                Generate signals
13:            end if
14:        end if
15:    end for
16: end procedure
```

5.4.3 BOLLINGER BANDS STRATEGY

Mean reverting strategies, explained in section 2.7, are strategies that leverage from the reversal of price to its mean. One of the most used indicators in mean reverting is Bollinger Bands. This indicator created by John Bollinger is consists in a moving average, an upper band and a lower band, as shown in figure 5.4. These bands are usually proportional to the standard deviation.

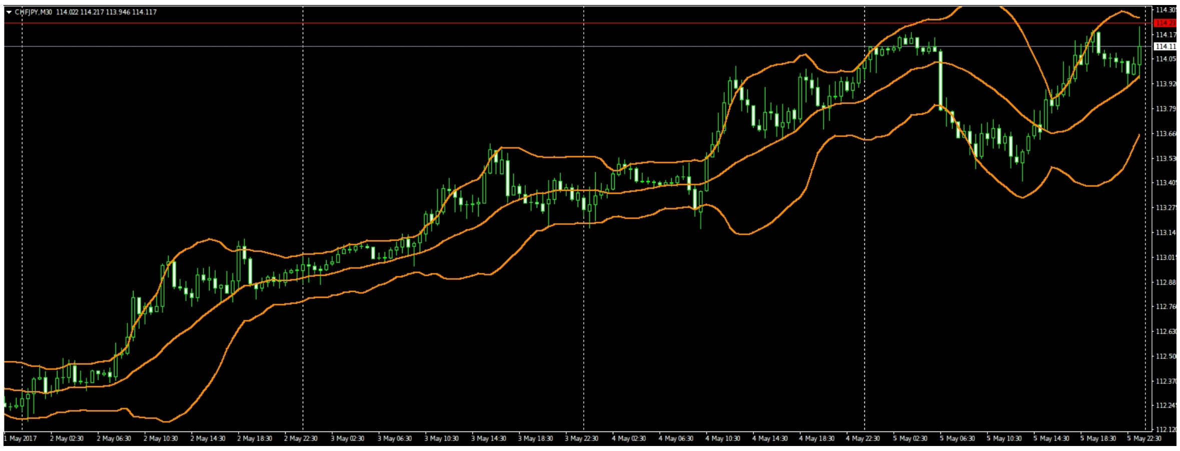


Figure 5.4: Screenshot of example chart with BB in Metatrader 4 platform

BB , as a tool, enables several types of trading. For example, the distance between bands can be used to assess volatility using the equation 5.3.

$$Bandwidth = \frac{(upperBB - lowerBB)}{middleBB} \tag{5.3}$$

In this case BB is used to detect instances of reversion. To achieve this, the following equation is used:

$$BBderivative_n = \frac{(price_n - SMA_n)}{2 \times RSTD_n} \tag{5.4}$$

where *RSTD* is a rolling STD with the same period of the *SMA*. This derived indicator from the BB is used to evaluate price position in relation to the bands.

The algorithm implemented starts by requesting 20 bars of historical data and using them to calculate the SMA and rolling STD. Then, the last value of each and the current price are used to calculate 5.4. Depending on the value of this derivative of BB positions are opened or closed. If this value is less than -1 than the price is two Standard Deviations (STDs) below the average and open-buy signal should be generated. Otherwise, if this value is more than 1, than the price is two STDs above the average and a open-sell signal should be generated. This means the price is expected to return to the mean after distancing itself two STDs from it. In addition, a stoploss similar to the one described in subsection 5.4.2 and the condition prohibiting open positions during weekends were implemented. Also, only one open position per instrument was allowed.

Algorithm 3 Bollinger Bands Strategy algorithm

```
procedure Calculate Signals
    for symbol in symbol list do
        Request price data for symbol
        if Current daytime is beyond trading period then
            end procedure
        else if Price activates stoploss then
            Generate signals
        else
            Calculate Bollinger Bands
            Assess BB derivative values
            if Conditions for BB derivative are met then
                Generate signals
            end if
        end if
    end for
end procedure
```

5.4.4 INTRADAY OLSHR STRATEGY

Pairs trading, addressed in section 2.8, involves matching long and short positions in a pair of correlated assets. In this case, criteria for selecting the assets was cointegration. In subsection 5.3.1 it is explained how the methods to determine cointegration were implemented. Also, in subsection 6.2.2 it is possible to analyze which of the availables currency pairs showed stronger cointegration. Based on this, EUR/AUD and EUR/NZD instruments were selected to test this strategy.

This strategy is called Intraday OLSHR and is based in the pairs trading strategy suggested by Ernest Chan[45]. Intraday because it was designed to be most profitable for short term trades, particularly trades that don't last more than a day. OLSHR because this strategy uses a Hedge Ratio obtained by computing a Ordinary Least Square regression and extracting the beta coefficient.

The algorithm that implements this strategy starts by requesting 100 bars of historical price data for the pair of assets. If the data is in acsohlc shape, only the Close values will be regarded, if it is tick data both bid or ask can be used. The next step is to calculate a HR to improve the stationarity of the residuals time series. In order to calculate it OLS regression was implemented, using the statsmodels[5] python library. The resulting beta coefficient(regression slope) is then used in the following equation:

$$spread_n = priceA_n - (HR \times priceB_n) \tag{5.5}$$

where *priceA* is the first instrument price series and *priceB* the second. The obtained spread is normalized by being subtracted by its own mean and divided by its STD, as seen in 5.6.

$$score = \frac{spread_k - Mean(spread)}{STD(spread)} \tag{5.6}$$

where the last value of the spread series is used to calculate the normalized score. This algorithm then uses the normalized score to assess the possibility of opening or closing trades. Also, it needs two border values to compare with the score. Changing these values have a direct impact on profitability, so after experimenting with different values, 3 and 0.5 were selected as border values. The first open condition is, if the score is lower or equal to -3.0 then a open-buy signal is generated for instrument A, and an open-sell signal is generated for instrument B. The second is, if the score is greater or equal to 3.0 then a open-sell signal is generated for instrument A, and an open-buy signal is generated for instrument B. In other words, when the previously calculated spread is more than a certain threshold, an opening signal should be generated. To close positions without depending on stop losses, another condition was added. If the absolute value of the score is lower or equal to 0.5 then close signals are generated for both instruments. In this strategy no stop losses were implemented but only one position was allowed per instrument and no positions were kept open through the weekend.

Algorithm 4 Intraday OLSHR Strategy algorithm

```
 1: procedure CALCULATE SIGNALS
 2:     for symbol in symbol list do
 3:         Request price data for symbol
 4:         if Current daytime is beyond trading period then
 5:             end procedure
 6:         else
 7:             Perform OLS regression over pair
 8:             Obtain residuals series with spread
 9:             Calculate normalized score
10:             Assess score & check for open positions
11:             if Conditions applied to score and previous scores are met then
12:                 Generate signals
13:             end if
14:         end if
15:     end for
16: end procedure
```

[5] http://www.statsmodels.org/stable/

5.4.5 BINARY CLASSIFICATION

Binary Classification, as described in section 3.2, tries to predict when the market is trending up or down. In a strategy context, a binary classifier model can be seen as a black box, where price and price generated indicators are inputted, and in the other side, 1 or -1 is outputted. 1 meaning the price is going up in the next time period, and -1 meaning the price is going down. Also in section 3.2, is described how to evaluate the classifier performance. If the model shows a hit rate close to 50% its predictions may have a random nature, but if by analyzing the confusion matrix the big majority of misses are only in predicting market drops than the model may be used to predict the contrary.

In this work, Binary Classification was used in two different manners. In an algorithm that only produces signals by analyzing the classifier or classifiers predictions and as an indicator to help validate signals in other strategies, such as the ones discussed in 5.4.3 and 5.4.2. Being that the goal for this implementation was to show the potentialities of using a machine learning model, a simplistic version of the first solution suggested was implemented. In order to do it, an algorithm that only relies on binary classifiers was created using the scikit-learn package. This algorithm uses a Radial Support Vector Machine to produce a prediction. The selected model, was chosen based on the results shown in 6.2.1. The algorithm starts by training the model. The data used, previously processed using the principles explained in 3.1 included minute price data for the asset being traded, price data for a correlated asset and price generated indicators. The assets chosen were EUR/AUD and EUR/NZD, based on results from subsection 6.2.2 and the set of technical indicators included the kalman filter and the BB derivative in 5.4. All price data is lagged 5 times and included as features. After the training process, a prediction is computed. If the prediction is 1 and there are no open positions, an open-buy signal is created, but if there is an open short position a close signal is generated. If the prediction is -1 and there are no open positions, an open-sell signal is created, but if there is an open long position a close signal is generated. In addition, a stoploss similar to the one described in subsection 5.4.2 and the condition prohibiting open positions during weekends were implemented. Also, only one open position per instrument was allowed.

Algorithm 5 Binary Classification Strategy algorithm

```
1:  Train ml model with available historical data
2:  procedure CALCULATE SIGNALS
3:      for symbol in symbol list do
4:          Request price data for symbol
5:          if Price activates stoploss then
6:              Generate exit signals
7:          else
8:              Re-train model with new data
9:              Make prediction
10:             Assess prediction & check for open positions
11:             if Prediction exists then
12:                 Generate signals based on prediction
13:             end if
14:         end if
15:     end for
16: end procedure
```

5.5 BACKTEST REPORTS

In order to fulfil its goals the backtest system had to include a complete report on strategy performance. In the interest of tackling this challenge, IPython notebooks were considered. Of common use in the world of Data Science, they are used to share experimental results, presenting full reports or creating interactive lessons. Using notebooks, each report containing performance metrics and graphs, such as the equity curve or signals generated, could be saved and shared between researchers.

Jupyter Notebooks[6] were used to consume performance data and create visualization tools. The system outputs a report in JSON format, as shown in the example 6 below. The report is then stored both in a MongoDB database instance and in a static JSON file.

Code 6 Performance output example for a Bollinger Bands Strategy

```
{
"volatility": {"EUR/USD": 0.077059243862789686},
"indicators": "BB",
"max_drawdown": 1.202000000000325,
"profit": 5.6571999999992739,
"instruments": ["EUR/USD"],
"profit_sell": 1345.4000000000106,
"consecutive_losses": 5,
"consecutive_wins": 13,
"filename": "_M15_2016",
"granularity": "M15",
"loss_buy": -1091.9600000000228,
"len_sell": 459,
"profit_buy": 1390.1600000000017,
"len_bars": 24956,
"sharpe": 0.60708926794324158,
"drawdown_duration": 6056.0,
"strategy": "BollingerBandsStrategy",
"loss_sell": -1077.879999999992,
"len_buy": 449
}
```

[6]https://jupyter.org/

The notebook can then import the report, from either the database or the static file, and show a prettified version of it. In order to import from a json file the Pandas package is used, transforming the imported data in a pandas dataframe. To import data from a MongoDB collection the Python's package pymongo[7] was used.

5.5.1 PERFORMANCE METRICS AND GENERATED SIGNALS

Every backtest system offer different metrics on its reports. As it can be seen in figures 2.3 and 2.2 a backtest report can show completely different metrics. In this work the metrics used were picked from both Quantopian's[8] solution and Metatrader4[9] while considering the suggestions presented in section 3.4 from [45],[9] and [37].

The first metrics implemented were the Sharpe Ratio, the Maximum Drawdown and Maximum Drawdown Duration, because of their importance in assessing long term profit of a strategy. Wins and losses were described by including the number of consecutive occurrences, the number of occurrences and the Profit and Loss generated by each trade direction. Also information such as the Strategy used, the instruments traded, the volatility of the instruments traded, the data source, data granularity and indicators used is given.

Also, any backtest relies on trading signals to assess a Strategy's performance. Analyzing when the signals were issued and in which order is vital to improve profitability. These trading signals, generated in the Strategy module, include a price value, a timestamp, a type and a direction. Because the use case was the FOREX market, three types of signals were considered: **OPEN-BUY**, **OPEN-SELL** and **CLOSE**, which can involve selling or buying depending on the open position trading direction.

In addition to outputting the signals, three more series were considered. They were the Drawdown, Equity Curve and Returns. The Returns series was created by storing the changes in market value of open positions in each iteration(The period of each iteration is one heartbeat). With the Returns series it is possbile to create an Equity Curve, implementing as shown in code 7 Drawdown was implemented using code 8 given the previously calculated equity curve.

Code 7 Equity Curve implementation

```python
import pandas

def generate_pnl(returns):
        #returns is a Pandas Series
        #Create equity curve with pandas's cumprod()
        pnl = (1.0 + returns).cumprod()
        return pnl
```

[7]https://pypi.python.org/pypi/pymongo/
[8]https://www.quantopian.com
[9]https://www.metatrader4.com/

Code 8 Drawdown implementation

```python
import pandas as pd

def generate_drawdowns(pnl):
        hwm = [0]
        # Create the drawdown and duration series
        drawdown = pd.Series(index = pnl.index)
        duration = pd.Series(index = pnl.index)
        # Loop over the index range
        for t in range(1, len(pnl.index)):
        hwm.append(max(hwm[t-1], pnl[t]))
        drawdown[t]= (hwm[t]-pnl[t])
        duration[t]= (0 if drawdown[t] == 0 else duration[t-1]+1)
        return drawdown, drawdown.max(), duration.max()
```

Both the three series(Drawdown, Returns and Equity Curve) and generated signals were stored in a static CSV file and in a MongoDB instance. The signals were stored by created a pair DayTime/Price, in which the first is the signal timestamp and the second is the price of the asset in the timestamp instant.

CHAPTER 6

Results

In this chapter it will be presented all the obtained experimental results. First it will be detailed all results from statistical analysis and learning algorithms. Then the results for each Strategy are analyzed and compared.

6.1 Live trading and backtest system

As explained in section 1.2 the goal of this work was to create a system to help researchers, a framework that could simplify the process and mitigate any bias. Thus, in this section it will be displayed the visual output for the strategy reports discussed in 5.5. In order to show the potential, or lack of it, in using machine learning, example results from the implemented learning algorithms performance will be displayed. In order to create example Strategies it was needed to study several price series, the more relevant results will also be in this section.

6.1.1 Reports

With the goal of further improving a strategy may be of interest to show a graph containing all trading signals with the asset historical price used in backtest and some useful indicators. There is the case of Pairs trading where trading signals are generated over a residuals time series of the difference between the historical price series of two assets. In this case the generated trading signals can be used with both historical price series and the residual time series, as they share the same time axis. The graph containing the trading signals then paired with the graph containing the returns over time. In this case pairing means that selecting a n number of points in the returns graph would also select points with the same X value(time) in the trading signals graph, as can be seen in figure 6.1 This helps to identify and study trading signals that generated large amounts of profit and loss. Following the same logic the graphs for Drawdown and Equity Curve are also linked. In addition to all the graphs a table containing all performance metrics is shown.

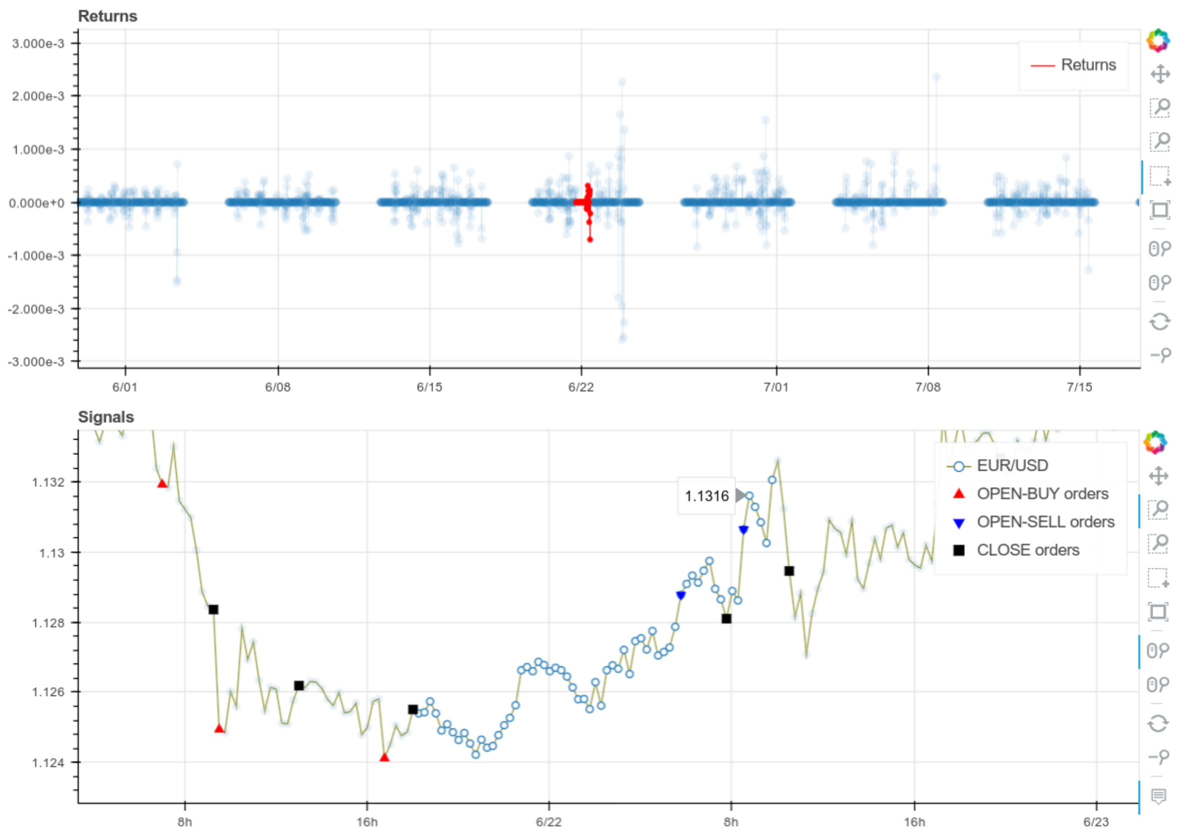
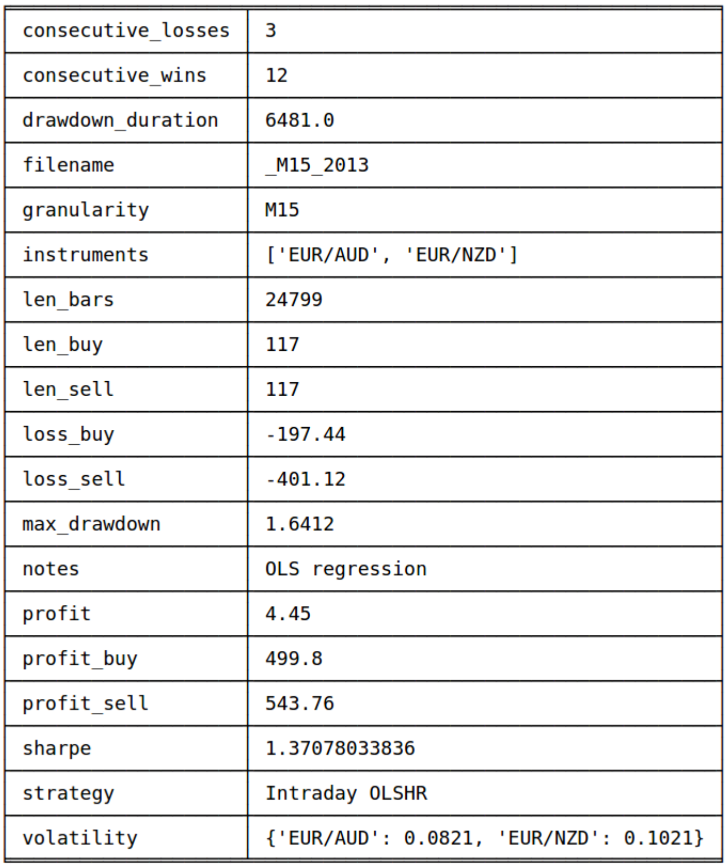


Figure 6.1: Graph with linked returns and price with trading signals



consecutive_losses	3
consecutive_wins	12
drawdown_duration	6481.0
filename	_M15_2013
granularity	M15
instruments	['EUR/AUD', 'EUR/NZD']
len_bars	24799
len_buy	117
len_sell	117
loss_buy	-197.44
loss_sell	-401.12
max_drawdown	1.6412
notes	OLS regression
profit	4.45
profit_buy	499.8
profit_sell	543.76
sharpe	1.37078033836
strategy	Intraday OLSHR
volatility	{'EUR/AUD': 0.0821, 'EUR/NZD': 0.1021}

Figure 6.2: Example of performance metrics in a backtest report

Besides analyzing the table of performance metrics it is useful to analyze each trade and their respective returns. In figure 6.3 it is possible to see two trades, both Long trades, triggered with a **OPEN-BUY** signal. Analysing the difference in price between **OPEN** and **CLOSE** signals it is possible to assess if the simulated trade was profitable or not. This, of course, doesn't take into account transaction costs. Also the signal structure was made with the use case in mind, other markets may imply a different signal structure. In the case a), visible in the image, the price continues rising until the position is exited. In this case the trade was profitable with positive returns. But in the case b), immediately after the opening, price drops, closing in a lower price. This produces negative returns. With Short trades the same logic applies, if a **OPEN-SELL** signal was issued but the price went up until the **CLOSE** signal arrival than the simulated trade was not profitable. In order to calculate the loss/profit the following equation is used:

$$(Close - Open) \times TradeSize \tag{6.1}$$

where *Close* is the price when the **CLOSE** signal was issued and open when the **OPEN** signal arrived. *TradeSize* represents the quantity of units bought.

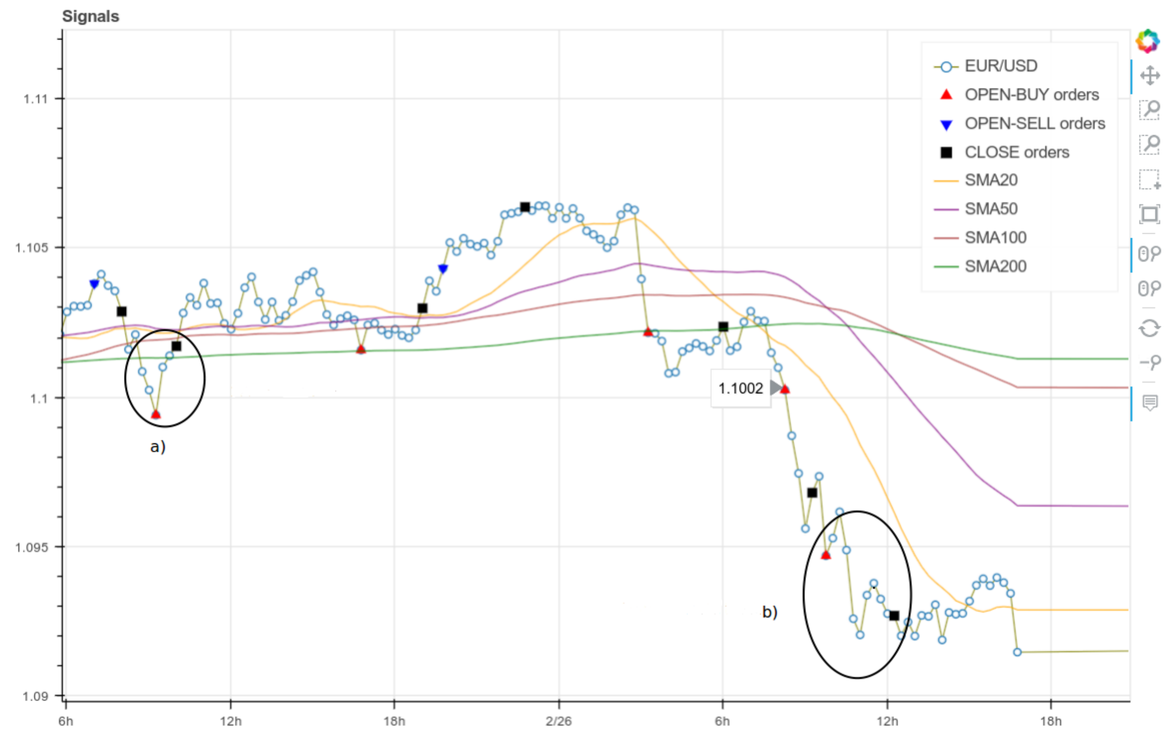


Figure 6.3: Graph with price, trading signals and a set of moving averages

In the case of a Pairs Trading strategy it can be interesting to study the residual time series. In the example detailed in 6.4 the Kalman Filter is used to further analyze the series [78].

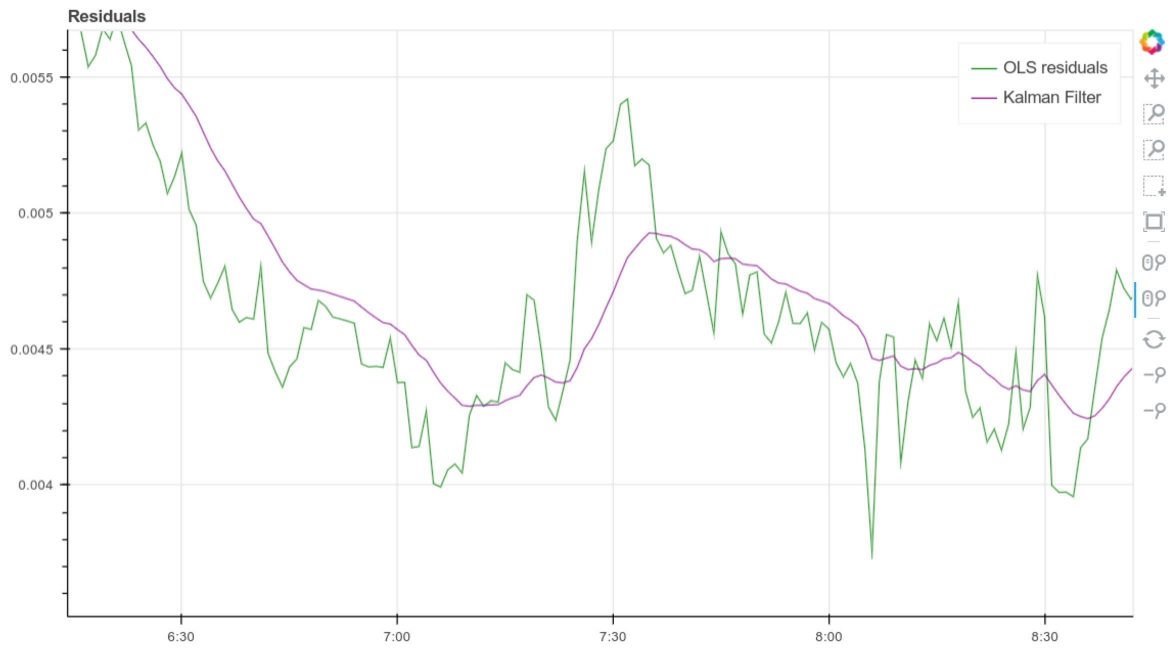


Figure 6.4: Example of residuals series with Kalman Filter

Having a positive profit means that at the end of the backtest period the balance between gains and losses were positive. However this doesn't not reflect profit distribution. A Strategy can be unprofitable for 90% of the backtest period and gain only in the last 10%. Also, there can be big losses hidden behind several small gains. These big losses can be the end of a trading account by exhausting all the margin available, forcing the brokerage service to take action. To tackle this 3 graphs were created with the goal of further analysing profit characteristics.

Figures 6.5, 6.6 and 6.7 are from the same backtest report, representing a backtest on 1 year of 15 minute data. In the returns graph, shown in figure 6.5, it is possible to analyze return behavior. In this case there are several High's and Low's. It is then in the interest of the researcher to find the respective trades. If there is a big return High but the position wasn't closed soon after, this return was not realized as profit and money was lost. If there is a big Low in returns, the researcher should investigate if the position was closed soon after, provoking big losses or later with a better return. Also, big Low's can be dangerous for a trading account, if the negative return is higher than the available cash in the account it can destroy all the trading margin.

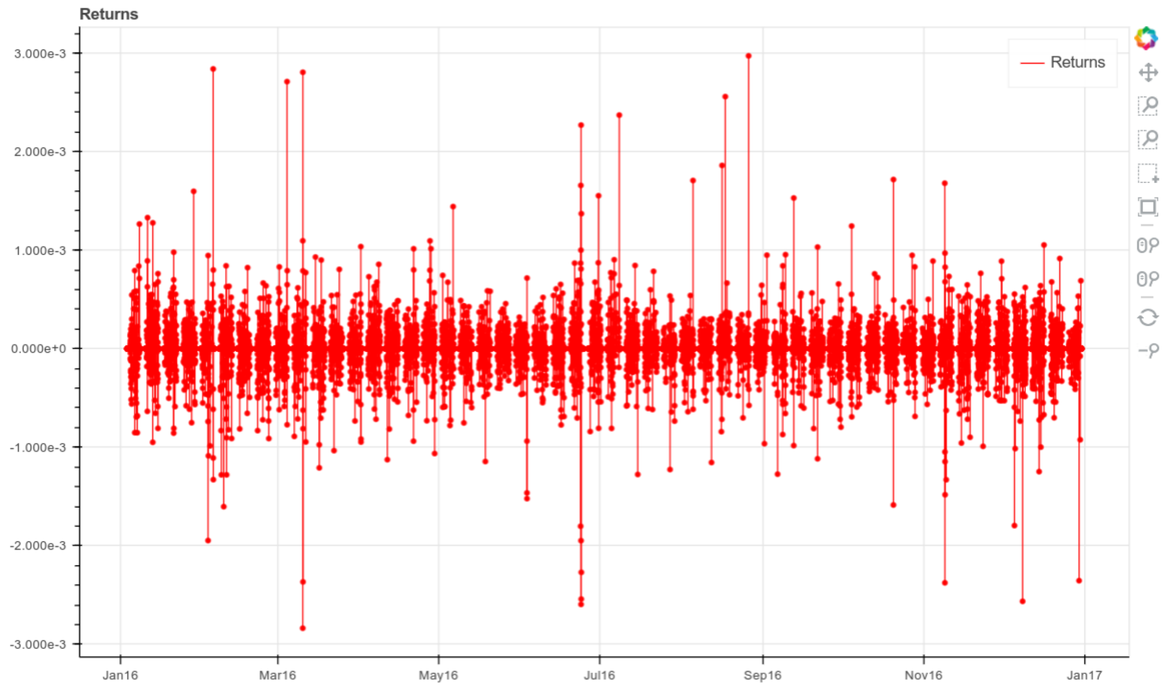


Figure 6.5: Example of the returns graph

In figure 6.6 it is possible to see an equity curve, which is a graphical representation of the change in value of a trading account. In this case the equity curve increases, revealing a positive slope. Having an equity curve with a positive slope, although with some retracements, means that at the end of the backtest period the Strategy was profitable. A negative slope would indicate the opposite, an unprofitable strategy. At least with the data/asset used.

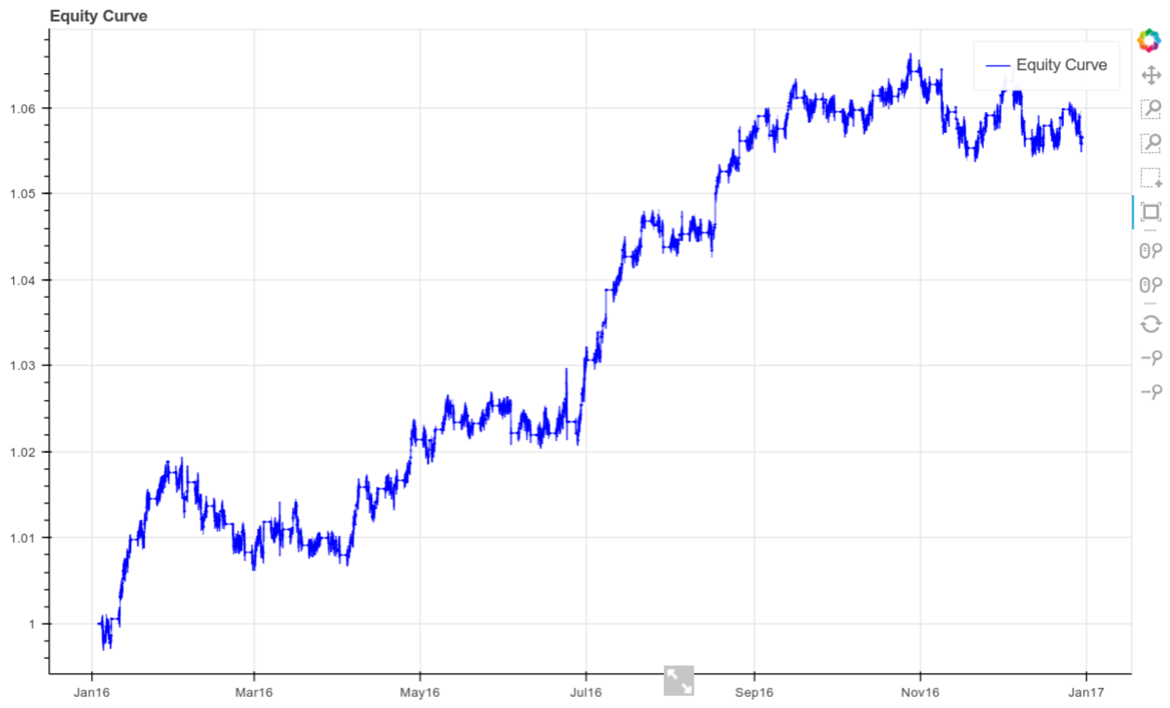


Figure 6.6: Example of the equity curve graph

Lastly there is Drawdown. Drawdown is the peak-to-trough decline during a specific period and help determine an investment's financial risk. In this case, the graph presented in 6.7 show a Drawdown with some major peaks. These peaks can be interpreted by researchers as instants of sudden drop in returns and should be a risk red flag.

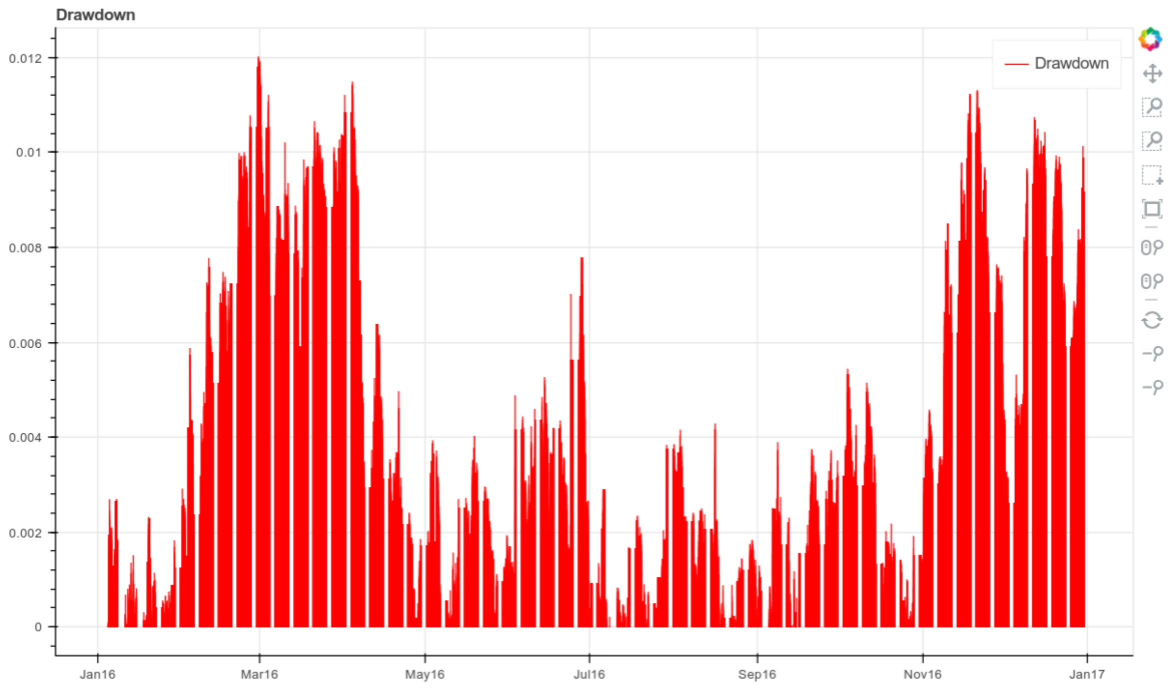


Figure 6.7: Example of the drawdown graph

6.2 TIME SERIES ANALYSIS AND MACHINE LEARNING

The implementation described in section 5.3 produced several results. In this section such results will be presented and analyzed. The machine used to perform all computations and run the algorithms has an Intel Core i5-2520M with 4 x 2.50GHz and 8gb of memory. The OS used was Ubuntu 16.04 LTS.

6.2.1 LEARNING ALGORITHMS

During the course of this work, several methods were created to use and evaluate machine learning models. In order to test system capabilities a few experiments were conducted, as detailed in section 5.3.2.

Figures 6.8, 6.9, 6.10 and 6.11 show the distribution of prediction scores for each binary classifier. Each prediction score is the result of testing one classifier trained with one month of data. None of the tests included today's price data, meaning all the price data used to train the models and generate indicators was lagged. Including today's data would produce far better results but would incur in a Lookahead Bias due to the end-of-period nature of the OHLC data. (Using the same methods, but including non-lagged price data, some articles show scores over 70%)

Figure 6.8 shows the prediction score for each model trained with 20 features. These 20 features were the product of lagging price data 5 times. Each lag contained 4 features: Open, High, Low and Close.

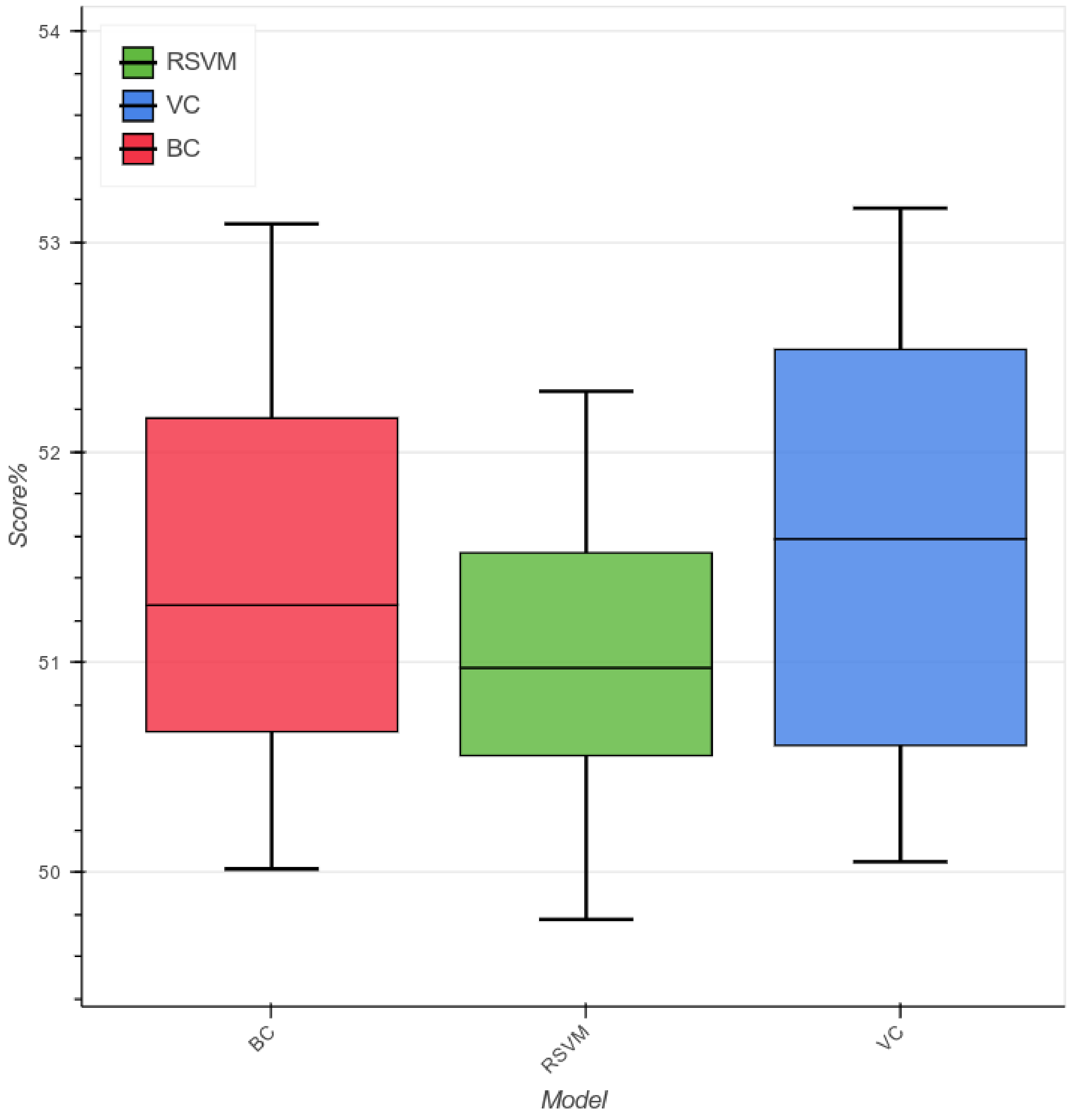


Figure 6.8: Prediction score distribution for 5 lags of OHLC price data

Figure 6.9 shows the prediction score for each model trained with 80 features. These 80 features were the product of lagging price data 20 times.

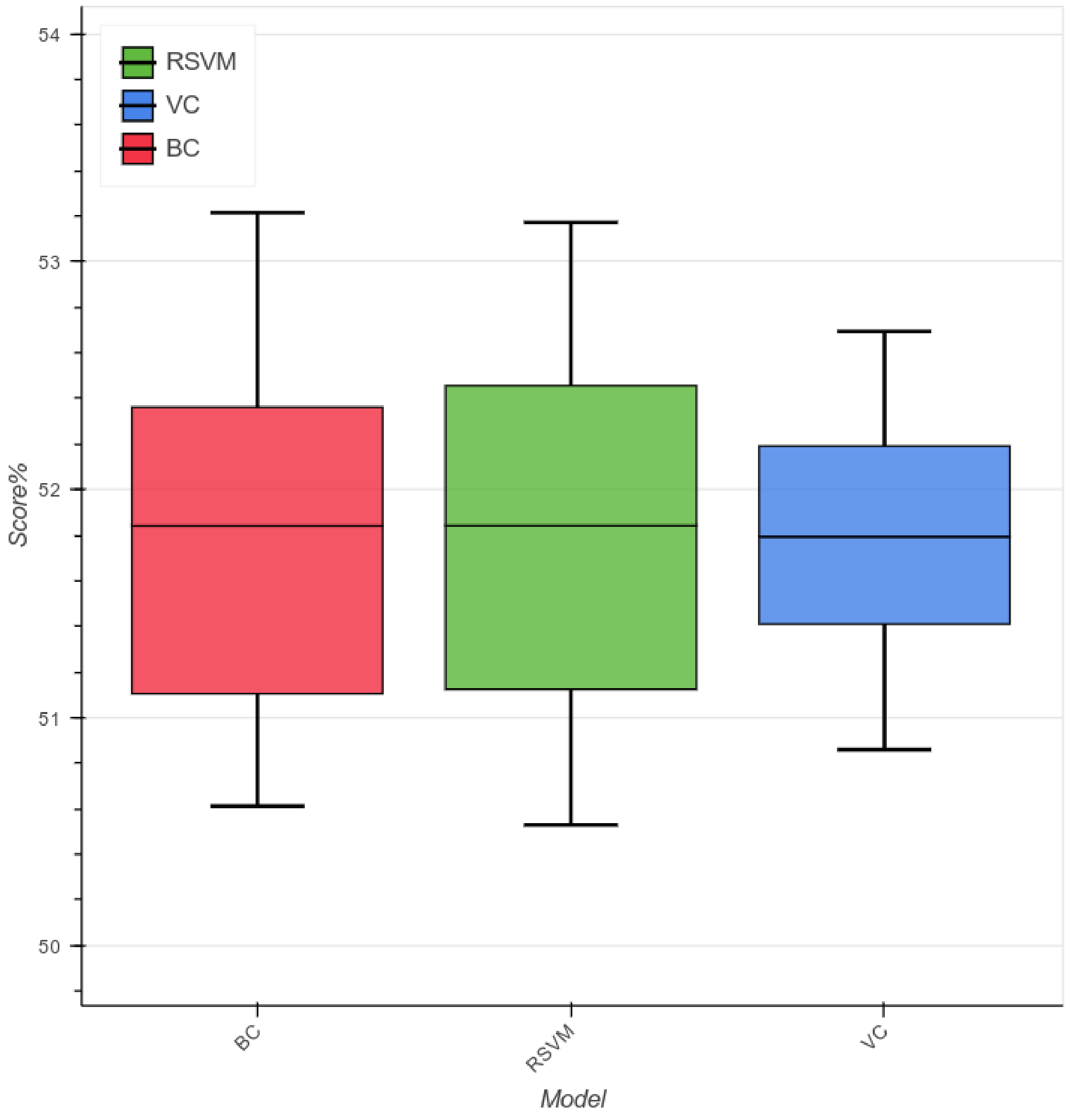


Figure 6.9: Prediction score distribution for 20 lags of OHLC price data

Figure 6.10 shows the prediction score for each model trained with 23 features. These 23 features were the product of lagging price data 5 times and adding 3 series of the bollinger derivative. Each one of the 3 bollinger features had different periods. The periods were 13, 20 and 31.

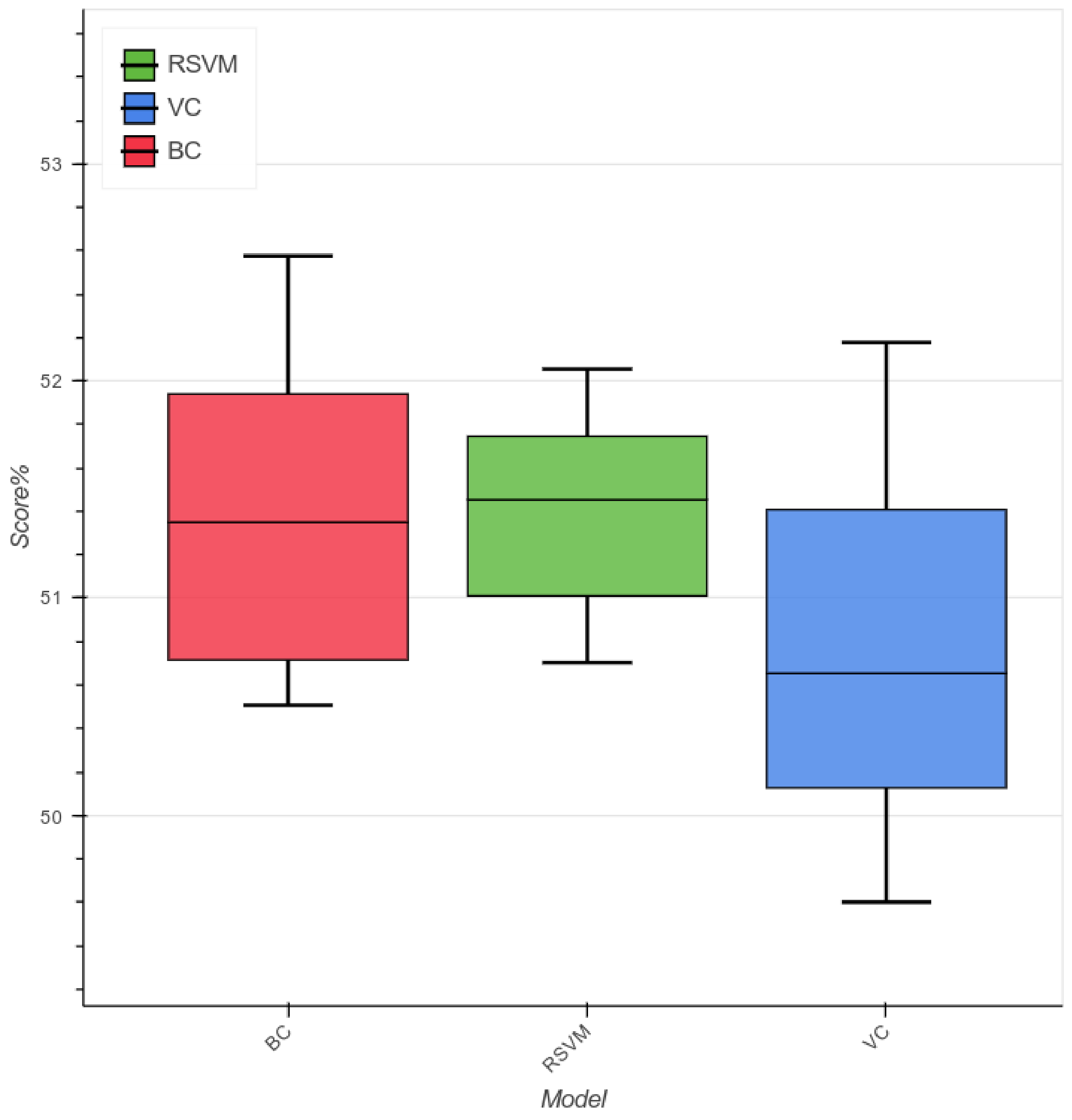


Figure 6.10: Prediction score distribution for 5 lags of OHLC price data and bollinger bands derivative

Figure 6.11 shows the prediction score for each model trained with 21 features. These 21 features were the product of lagging price data 5 times and adding the state means, obtained by applying the Kalman Filter to a series of Close price lagged once.

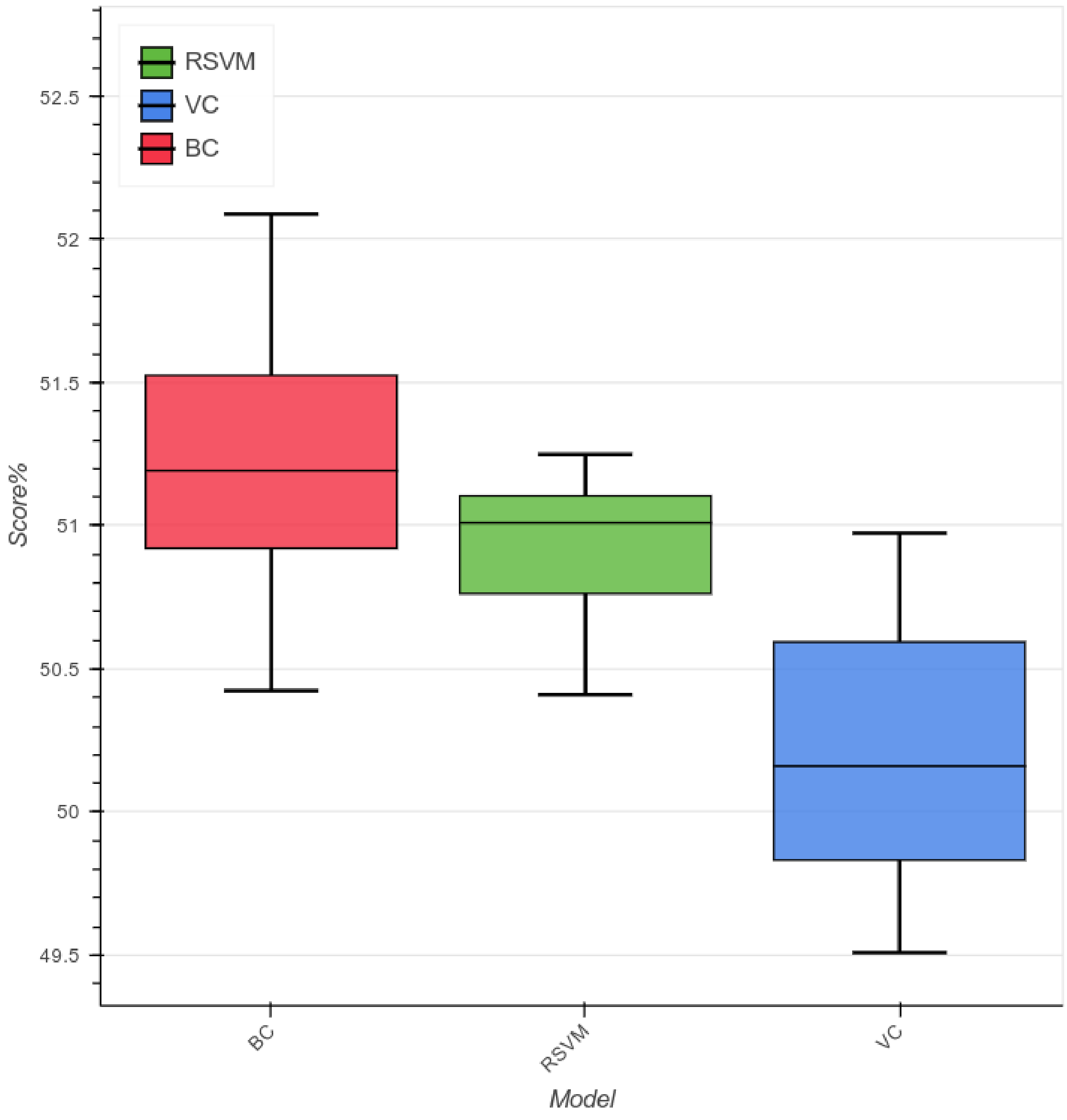


Figure 6.11: Prediction score distribution for 5 lags of OHLC price data and Kalman Filter

None of the previous charts shows scores above 54%, revealing the seemingly random nature of the trained models. The VC was the most successful in predicting the changes in price, but still with low accuracy.

Further tests using a different features, different dimensionality or PCA and RFE produced similar results. For example, tests were made using features derived from both cointegrated currency pairs EUR/NZD and EUR/AUD in order to predict price change in EUR/NZD. None of these test scores were satisfatory.

In addition to this superficial view of the learning algorithm's accuracy it was important to assess system performance while executing it. In order to test it, the three learning algorithms were trained with 20 features of 1 month of 1 minute price data and evaluated separately. A python script, using the library psutil[1], was executed in parallel to the training of the learning algorithms in order to retrieve memory usage and CPU times. The tests were repeated 3 times and the results averaged.

In figure 6.12 it is possible to see the averaged percentage of used memory during the execution. Figure 6.13 represents the seconds the CPU has spent both in user and system mode. Both graphs confirm that both the BC and VC models require more resources. Being the BC algorithm the most heavy.

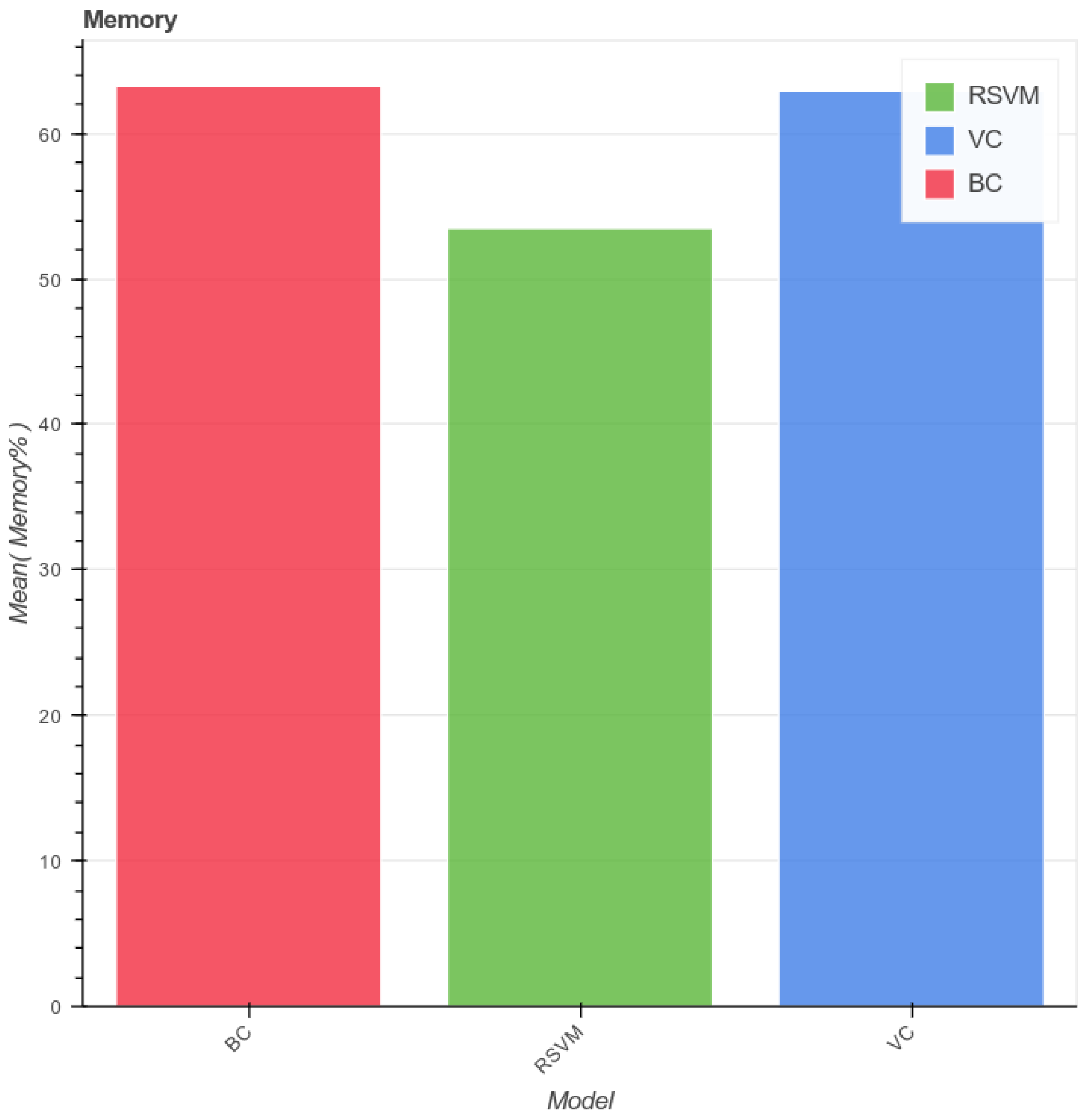


Figure 6.12: Process time and Wall time in seconds for each model trained with 5 lags of OHLC price data

[1]https://github.com/giampaolo/psutil

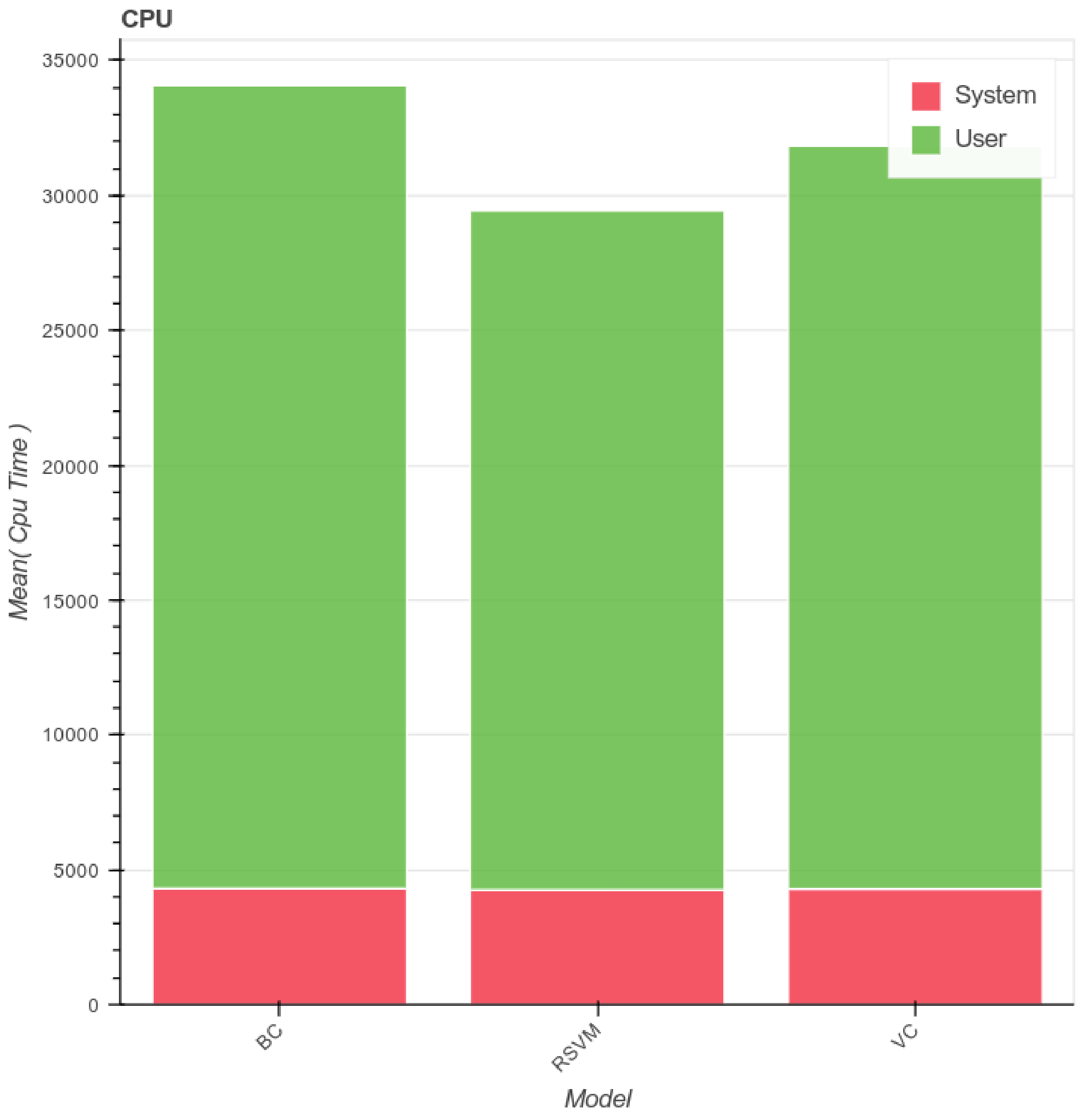


Figure 6.13: Time spent by the CPU with user processes and kernel processes

Figure 6.14 shows the average process time(Time spent processing between two point in code) and wall time(Time elapsed between two point in code) elapsed during the process of training each model. As expected BC was the slowest, spending in average 640 seconds to train. The VC model spent in average 300 seconds, half of the time spent by the previous model. The RSVM algorithm, being the simplest, spent an average of 120 seconds training.

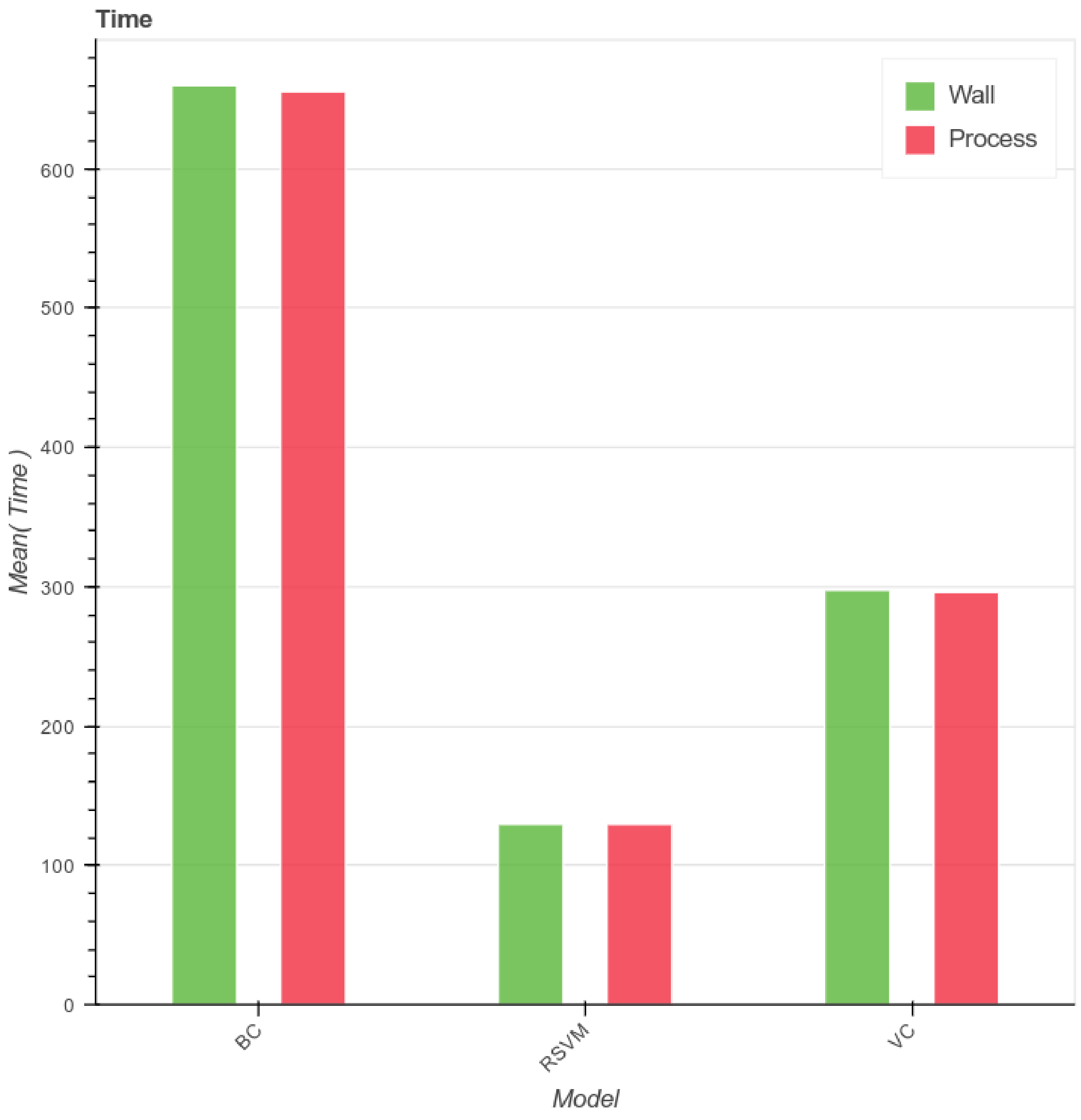


Figure 6.14: Process time and Wall time in seconds for each model trained with 5 lags of OHLC price data

By analyzing both the prediction scores and time spent training of each model a decision on which model to use, in the binary classification strategy, was made. The RSVM was chosen because it produced prediction scores similar to the BC and VC algorithms while spending less time during the training process.

6.2.2 STATISTICAL ANALYSIS

During the course of this work there was the need to perform statistical analysis on price time series. This need was due to both the requirement of a Cointegrated pair of instruments to test the Pairs trading strategy and to experiment different statistical scenarios.

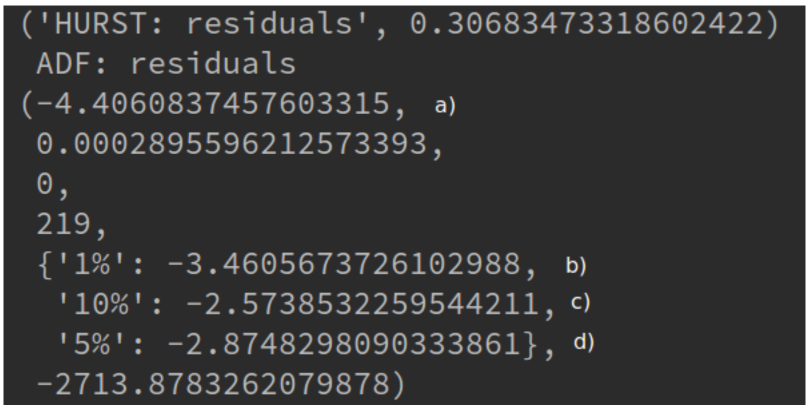
After calculating both the ADF and HE it was necessary to interpret the results. Table

6.1 shows how to determine a series behavior by analyzing the HR result.

Figure 6.15 shows an example result for ADF and HE tests over a residual time series(Series obtained from EUR/AUD and EUR/NZD 15 minute price data from 2012). To compute this example, a similar implementation to the one explained in section 2.8 is used for obtaining a residuals time series from two price series with the OLS regression beta coefficient as hedge ratio. In this example it is possible to see that the ADF outputs a critical value a) inferior to the 1%, 5%, and 10% values, which means the null hypothesis that there isn't a cointegrating relationship can be rejected [37][45].

Test Result	Series
HE > 0.5	Trending
HE $= 0.5$	Geometric Brownian(Random Walk)
HE < 0.5	Mean Reverting

Table 6.1: Nature of Series based on the HE value



```
('HURST: residuals', 0.30683473318602422)
 ADF: residuals
(-4.4060837457603315, a)
 0.0002895596212573393,
 0,
 219,
 {'1%': -3.4605673726102988, b)
  '10%': -2.5738532259544211, c)
  '5%': -2.8748298090333861}, d)
 -2713.8783262079878)
```

Figure 6.15: Example of CADF and HE results

Based on the available historical price data an hypothesis was constructed. The hypothesis was that due to New Zealand's economy dependence on Australia's economy, the pairs EUR/NZD and EUR/AUD would be cointegrated [80]. In order to test this assumption the CADF test and the HE were performed on all combinations of currency pairs with the EUR/AUD.

- **A** - EUR/AUD and EUR/NZD
- **B** - EUR/AUD and EUR/JPY
- **C** - EUR/AUD and EUR/GBP
- **D** - EUR/AUD and EUR/CAD
- **E** - EUR/AUD and EUR/USD

Each column in tables 6.4 and 6.5 represent one of two possible tests that could be performed on each pair of FOREX instruments. Using the equation 5.5 and changing *priceA* with *priceB* it is possible to obtain two different residual series. For example, the two combinations for A are:

$$A1 = (EUR/AUD) - (HR \times (EUR/NZD)) \tag{6.2}$$

$$A2 = (EUR/NZD) - (HR \times (EUR/AUD)) \tag{6.3}$$

The results showed below were calculated by using price data with granularity of 1 minute and 15 minute. Both data sets included data from the period 2012-2016. Each test was made using one year of data, results from each year were then averaged with equal weights and displayed in tables 6.4, 6.5, 6.4, 6.3 and 6.2.

	CADF									
	A1	A2	B1	B2	C1	C2	D1	D2	E1	E2
Critical Value	-1.3650	-0.9441	-1.0012	-0.4155	-1.4421	-0.1133	-1.8339	-1.4209	-0.9745	-0.2114
1%	-3.4305	-3.4305	-3.4305	-3.4305	-3.4305	-3.4305	-3.4305	-3.4305	-3.4305	-3.4305
5%	-2.8616	-2.8616	-2.8616	-2.8616	-2.8616	-2.8616	-2.8616	-2.8616	-2.8616	-2.8616
10%	-2.5668	-2.5668	-2.5668	-2.5668	-2.5668	-2.5668	-2.5668	-2.5668	-2.5668	-2.5668

Table 6.2: Averaged CADF test results for 1 minute data

	Hurst Exponent									
	A1	A2	B1	B2	C1	C2	D1	D2	E1	E2
Result	0.4508	0.4490	0.4742	0.4844	0.4702	0.4648	0.4473	0.4616	0.4727	0.4811

Table 6.3: Averaged HE test results for 1 minute data

	CADF									
	A1	A2	B1	B2	C1	C2	D1	D2	E1	E2
Critical Value	-2.2985	-2.9818	-2.0264	-0.8148	-1.4134	-1.6823	-1.6330	-1.9175	-1.9469	-1.9772
1%	-3.4314	-3.4314	-3.4314	-3.4314	-3.4314	-3.4314	3.4314	-3.4314	-3.4314	-3.4314
5%	-2.8620	-2.8620	-2.8620	-2.8620	-2.8620	-2.8620	-2.8620	-2.8620	-2.8620	-2.5670
10%	-2.5670	-2.5670	-2.5670	-2.5670	-2.5670	-2.5670	-2.5670	-2.5670	-2.5670	-2.5670

Table 6.4: Averaged CADF test results for 15 minute data

	Hurst Exponent									
	A1	A2	B1	B2	C1	C2	D1	D2	E1	E2
Result	0.3981	0.4164	0.4987	0.5035	0.4862	0.4555	0.4912	0.4883	0.5104	0.5095

Table 6.5: Averaged HE test results for 15 minute data

After analyzing the Hurst Exponent and Cointegrated Augmented Dickey Fuller test results for the one minute data is possible to see that both A and D series show weak cointegration. The CADF test result shows no cointegration at all, as the four critical values of A and D series are higher than the 1%, 5% or 10% values. However the HE shows higher stationarity in the residuals series from A and D than the rest of pairs, despite the short difference between them. Both A and D have show the lowest HE values.

With the 15 minute data is possible to confirm the conclusions made with the one minute data. The CADF test result for A series shows no cointegration for the case A1, but in A2, the critical value is lower than the 5% and 10%values, thus the null hypothesis of the ADF can be rejected and mean reversion can be found in the residuals series. The results for D series also show no cointegration with high critical values. As for the HE, it also shows stronger stationarity in series A, with an A1 value of 0.3981 and A2 with 0.4164. However for the 15 minute data the D series showed weaker stationarity with a D1 value of 0.4912 and D2 value of 0.4883, both very close to value of 0.5.

Taking into account both one minute data and fifteen minute data, the case A of using EUR/AUD and EUR/NZD produced the most stationary series. This is why this pairs of instruments were used as use case in the pairs trading strategy.

6.3 STRATEGY BENCHMARKING

The implemented strategies, detailed in section 5.4 were used to test the created backtest system. In this section the results will be analyzed, namely the sharpe ratio and the return percentage.

From the state of the art, detailed in sections 2.1 and 2.7, it can be concluded that a technical analysis approach works better for smaller trading periods opposite to a fundamental analysis approach which produces better results in higher periods. The example strategies were developed using a technical analysis approach, thus the granularities used for testing were 1 minute(M1), 15 minutes(M15) and 1 hour(H1) while higher periods like 1 day or 1 month were discarded. In addition, to backtest higher trading periods more data should be available, otherwise the results may not be significant.

For M1 data each backtest performed on 1 month of data, with a total of 36 backtests for each instrument(Or a set of instruments depending on the number of instruments being used at the strategy). For M15 and H1 each backtest was performed on 1 year of data, with a total of 3 backtests for each granularity. The data used to perform the backtests was OHLC price data ranging from 2014 to 2016. Also, strategies were improved using in-sample data from the years 2012 and 2013. By dividing the data in chunks, in-sample and out-of-sample, the goals was to follow the concepts explained in chapter 3. In this case the following instruments were used: EUR/USD, EUR/AUD, EUR/NZD, EUR/GBP and EUR/CAD.

Table 6.6 shows the averaged sharpe ratios and total return for the backtests performed with 1 minute price data. As detailed in section 3.4 strategies with a sharpe ratio below 1 reveal high risk, taking into account the amount of return. The Bollinger Bands strategy is the only shown in the M1 table that regardless of the instrument has sharpe ratios greater than 1. In three of four cases the ratios are even higher than 2, which is the minimum for strategies used in quantitative trading investment funds. The total returns for the four instruments are close to 2%, revealing an homogeneous behaviour of small returns. The Trend Following strategy shows smaller sharpe ratios, all below 1 and returns near 1%. Despite having positive returns this strategy will hardly be profitable in a live trading context, taking into account the low sharpe ratios and future transaction costs. The Binary Classification strategy appears to have a random nature with both positive and negative sharpe ratios. The fact that it uses a model with low prediction scores, as exposed in section 6.2.1, only confirms its the random nature.

M1	Bollinger Bands		Trend Following		Binary Classification	
Instrument	Sharpe	Returns	Sharpe	Returns	Sharpe	Returns
EUR/USD	2.09	2.86	0.53	1.12	-0.16	-0.12
EUR/AUD	3.35	2.84	0.77	0.75	-0.56	-0.34
EUR/GBP	1.34	1.71	0.41	0.42	0.81	0.23
EUR/CAD	3.46	2.48	0.53	1.12	0.27	0.46

Table 6.6: Averaged sharpe ratio and total return for 1 minute price data

Table 6.7 shows the averaged sharpe ratios and total return for the backtests performed with 15 minute price data. In this case BB strategy and Trend Following strategy show similar performance in terms of sharpe ratio and total returns. BB performs better in EUR/USD and EUR/CAD, with the sharpe ratio exceeding 1 and returns higher than 3%. However it shows negative sharpe ratio and returns for EUR/AUD. Trend Following strategy has a similar performance in the 15 minute period to its performance in the 1 minute period. Low sharpe ratios, all below 1, but all positive with only one negative return for EUR/CAD. Binary Classification shows once again a random nature with both positive and negative sharpe ratios, with both returns higher than 3.5% and lower than 4.5%.

M15	Bollinger Bands		Trend Following		Binary Classification	
Instrument	Sharpe	Returns	Sharpe	Returns	Sharpe	Returns
EUR/USD	1.31	3.13	0.60	2.62	-0.45	-4.89
EUR/AUD	-0.399	-0.06	0.95	3.88	0.02	-0.12
EUR/GBP	0.53	0.78	0.29	0.70	0.86	3.73
EUR/CAD	1.03	3.89	0.03	-0.43	-0.12	0.25

Table 6.7: Averaged sharpe ratio and total return for 15 minutes price data

Table 6.8 shows the averaged sharpe ratios and total return for the backtests performed with 1 minute price data. As the periods grow bigger sharpe ratios and profits went lower. BB strategy shows all sharpe ratios below 1, with the EUR/AUD showing a negative -0.14. The Trend Following strategy followed the same trend as the BB strategy with three negative sharpe ratios. Only EUR/AUD showed positive sharpe ratio, still lower than 1. Binary Classification has a sharpe ratio of 1.2 for EUR/GBP, but due to all the other values analyzed so far this becomes irrelevant.

H1	Bollinger Bands		Trend Following		Binary Classification	
Instrument	Sharpe	Returns	Sharpe	Returns	Sharpe	Returns
EUR/USD	0.07	0.08	-0.51	-3.07	0.48	1.97
EUR/AUD	-0.14	0.28	0.61	1.59	0.12	0.47
EUR/GBP	0.41	0.51	-0.20	-0.36	1.20	4.55
EUR/CAD	0.87	1.99	-0.71	-3.91	-0.56	-3.96

Table 6.8: Averaged sharpe ratio and total return for 1 hour price data

Because the Intraday OLSHR strategy requires at least two instruments being traded at the same time it was not directly compared with the previous strategies. Instead two pairs of instruments were chosen to test the strategy, the first, EUR/AUD & EUR/NZD, was the one that showed more cointegration and the second, EUR/AUD & EUR/USD, the one that showed the least. The same tendency for higher returns and sharpe ratios in smaller periods remained. The M1 period showed the best results in all backtests and, as expected, the strategy performed much better in the cointegrated pair, with an high 5.27 sharpe ratio and 6.80% average returns versus a 1.75 ratio and 2.01%returns. In the M15 period the cointegrated pair showed a sharpe ratio of 1.77 and returns of 4%, the H1 also showed positive values with a sharpe ratio of 0.96 and returns of 2.24%. Even the least cointegrated pair, EUR/AUD & EUR/USD, showed positive sharpe ratios and returns in every period.

	Intraday OLSHR					
	M1		M15		H1	
Instrument	Sharpe	Returns	Sharpe	Returns	Sharpe	Returns
EUR/AUD - EUR/NZD	5.27	6.80	1.77	4.01	0.96	2.24
EUR/AUD - EUR/USD	1.75	2.01	0.33	0.90	0.65	1.74

Table 6.9: Averaged sharpe ratio and total return for the Intraday OLSHR strategy

Figure 6.16 shows the distribution of the total return with the Intraday OLSHR strategy for M1, M15 and H1 periods. The previously discussed trend of higher returns in lower periods is visible, with the returns for M1 even exceeding the 14% threshold and never going below the 1.7%. The return percentages for M15 had a tighter distribution with values varying between 4.5% and 2.8%. H1 showed some negative returns though with a majority of values going above 0.5% and in some periods above 2%.

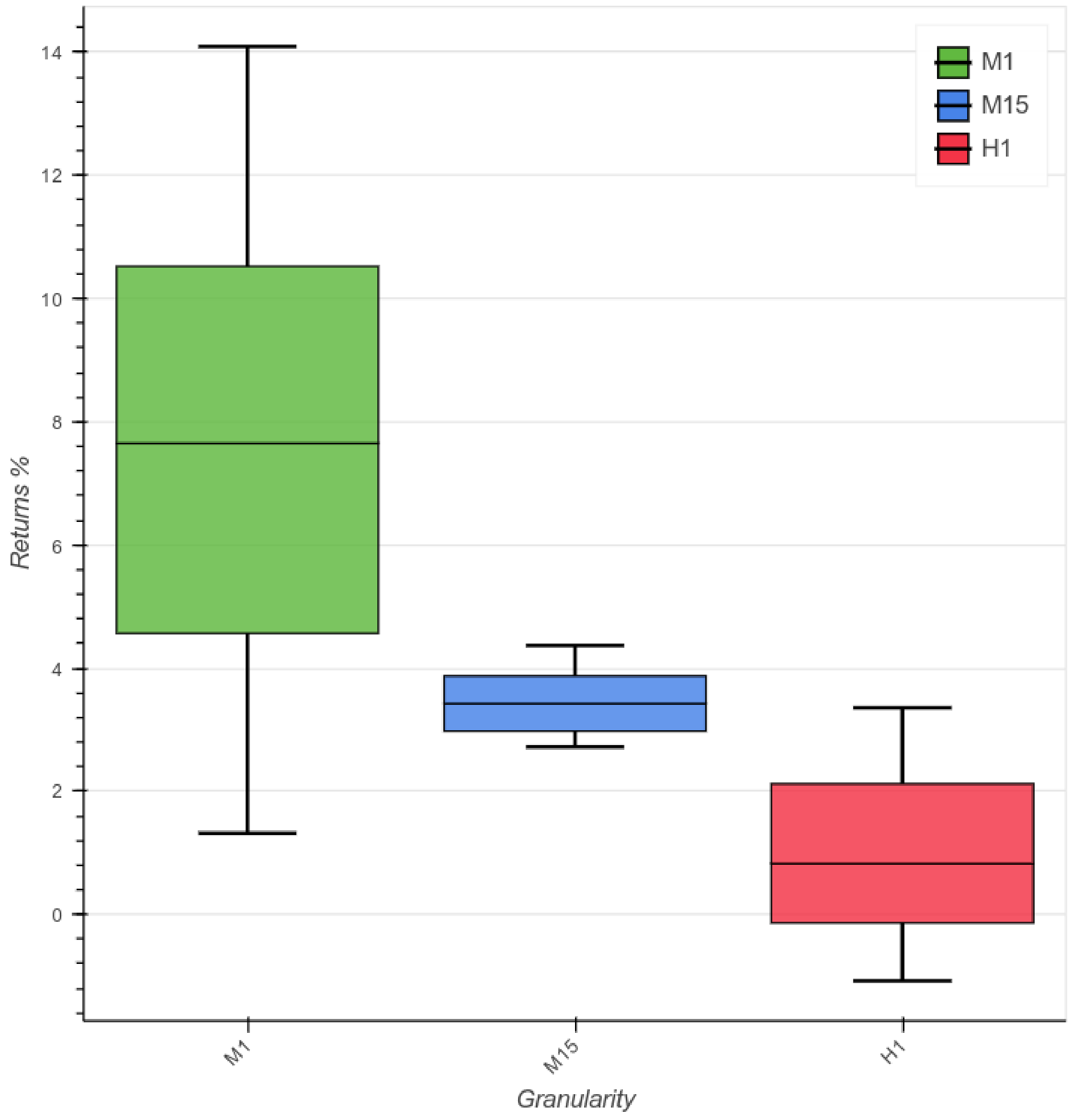


Figure 6.16: Total return distribution for the Intraday OLSHR strategy for different granularities

It is worth noting that, as expected, Trend Following and Bollinger Bands strategies don't perform well in the same instruments. A trend following strategy profits from a price series with large and prolonged trends, contrary to the BB strategy, which due to its mean reverting nature, profits from frequent mean reversals and short trends.

6.4 BACKTEST SYSTEM PERFORMANCE

In this section the created strategies will be compared with focus on execution time, cpu usage and the resulting sharpe ratio.

Figure 6.17 shows the comparison between strategies in terms of process time and wall time spent per backtest iteration in seconds. Also, figure 6.18 shows the percentage of cpu usage during each backtest iteration. The Trend Following strategy clearly consumes more time and requires more processing power. This strategy requires 100 units of OHLC data in order to compute the three needed indicators(RSI, and two EMAs). In contrast with the Bollinger Bands strategy that only requires 20 units of Close price data to calculate one indicator. The Intraday OLSHR strategy, despite performing an OLS regression over 100 units of Close price data still performs better than the Trend Following Strategy.

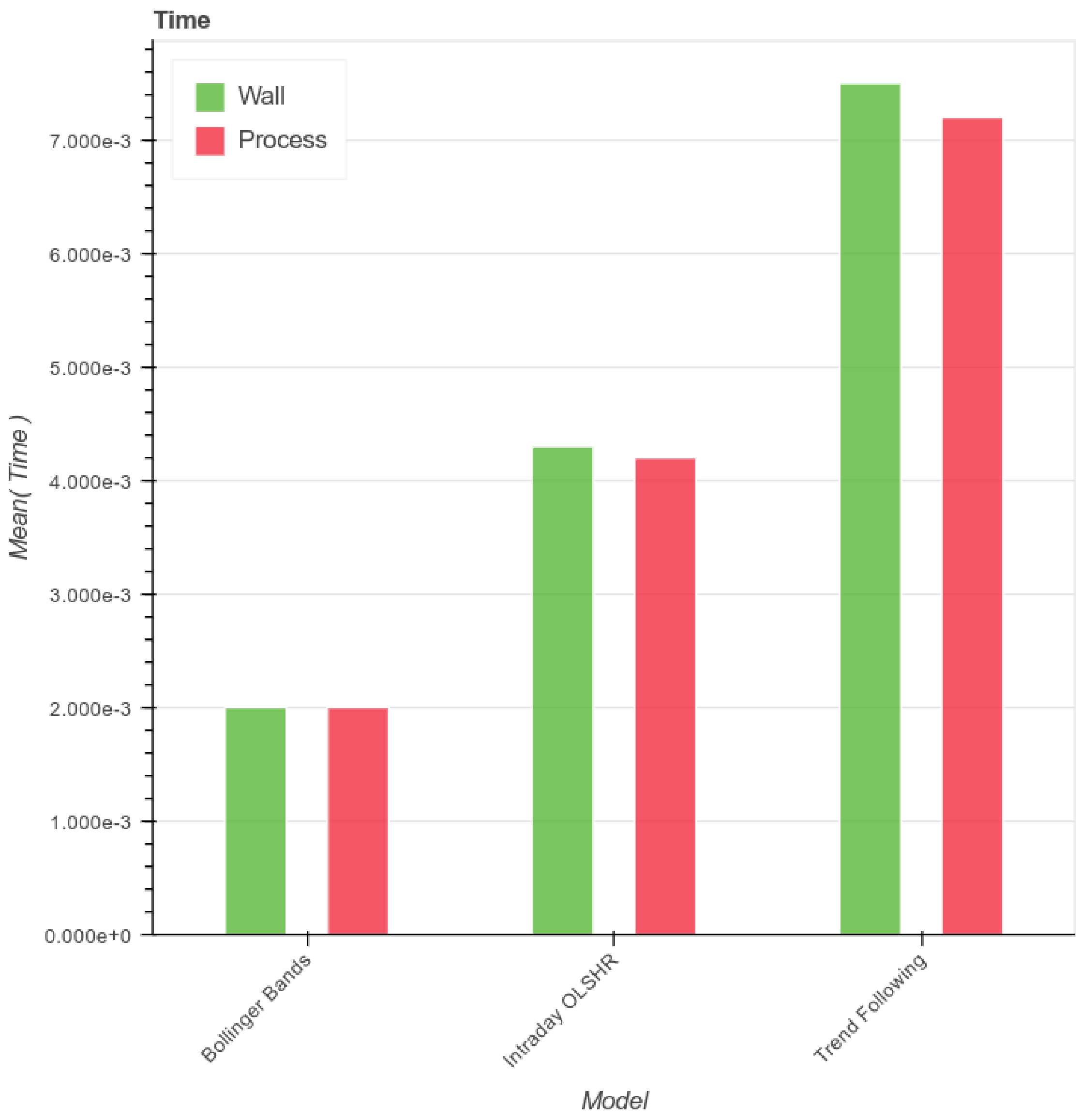


Figure 6.17: Process time and Wall time in seconds for each strategy

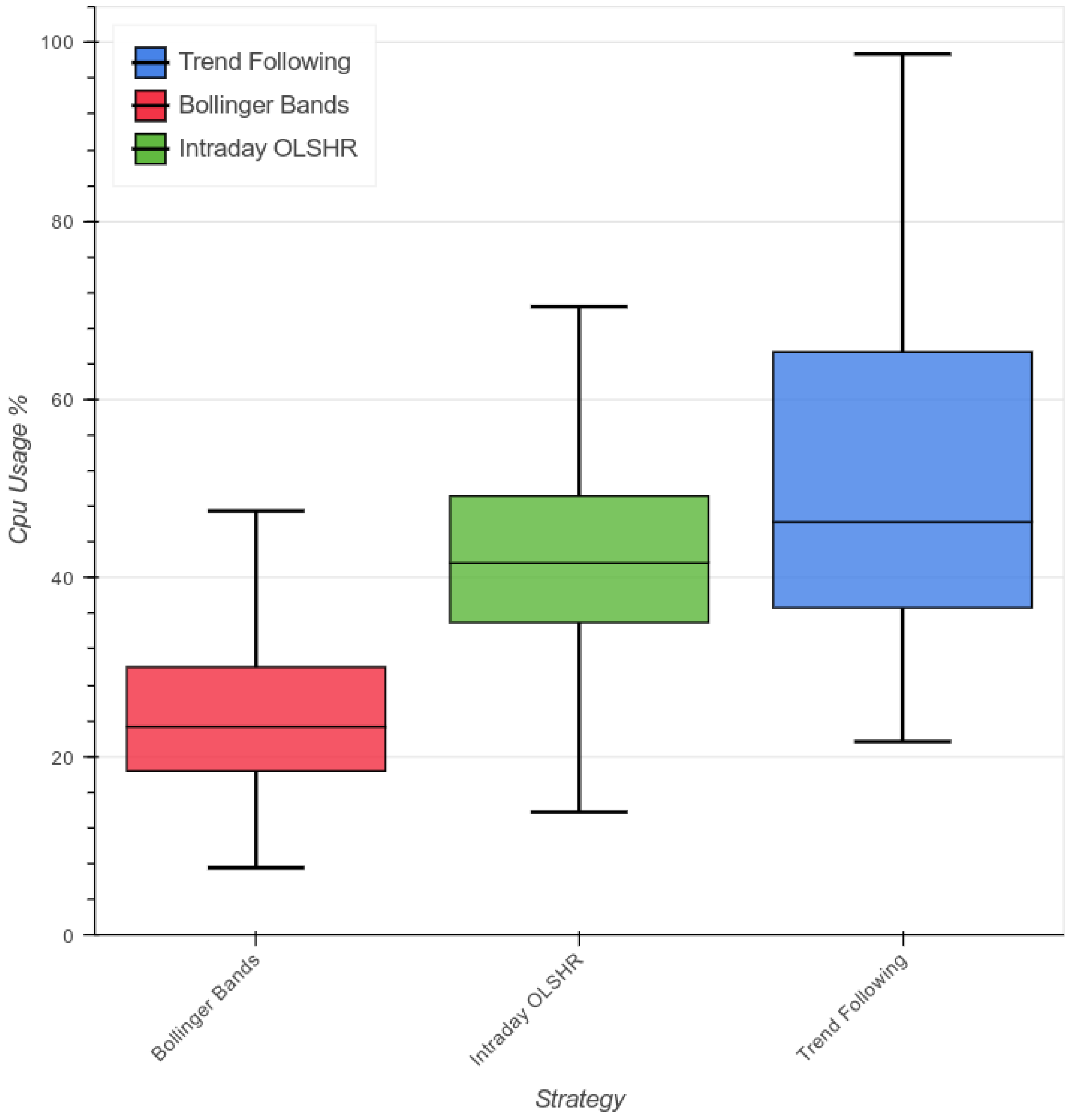


Figure 6.18: CPU usage in percentage for each strategy

The binary classification strategy was not included in this comparison due to its relative impact in system performance. Opposite to the previous strategies, binary classification's execution time and cpu usage depend on more variables. The model being used and the amount of data used to retrain it each iteration are the main variables. So while the most slow strategy, the Trend Following strategy, took less then 8ms to finish one iteration the classification strategy, with the smallest dataset available(approximately 10000 rows of OHLC data) and one of the quickest machine learning classifiers, took an average of 25s.

CHAPTER 7

Conclusion

This chapter concludes the presentation of the developed work for this dissertation, while discussing the obtained results and what work remains to be done to improve this project.

7.1 Final considerations

This work aimed to create a strong research tool to help researchers advance in the field of quantitative trading. Algorithmic trading, as an element of quantitative trading, has the process of backtesting in its core. To backtest is a science per se. It's a process full of pitfalls and with a relative intrinsic value, still a subject of discussion by economists. Thus it was imperative to learn the good practices of backtesting in order to create a reliable system. The goal was always to apply a scientific approach to the problem and create a system with reduced discretionary inputs both in backtesting and in live trading. Which meant going in the opposite direction of the classic Excel backtest approach and chart live trading.

As an hybrid between computer science and quantitative finance, this dissertation required extensive search in both areas. From which technologies to use to which strategies to create, the challenges were many. In the financial field it was required to learn about the main financial markets and the respective assets in order to choose the use cases for this dissertation. The different types of trading strategies and the world of technical analysis, including its wide range of technical indicators were also considered in the background investigation prior to this work. In addition, a set of trading platforms with backtest capabilities was studied to allow the creation of a solution that complied with the paradigm. Brokerage firms also required extensive search. Being the FOREX market the use case, there were many choices in terms of brokers. Ultimately the choice fell on OANDA, prioritizing the offered API instead of the broker's trading conditions. One factor that was not taken into account in this case, but imperative for live trading, was the transaction costs.

When it comes to computer science the main challenges were which programming language to use, how to establish communication between modules, the system architecture, which database to use and how to offer visualization of data. Productivity and flexibility were the main drivers behind each choice relating to technology choices. Also, instead of using one of many open source backtest frameworks available to achieve the goals the system was created from scratch. This decision rested upon the belief that by creating it, a more extensive knowledge about the backtest process would be obtained. Providing machine learning and statistical analysis applied to price series was one of the objectives of this work. Both were easily implemented by using python's powerful stack of packages and example results were produced. However, in the case of machine learning it also became a hardship. Making accurate predictions on price changes using binary classifiers revelead to be an hard task, despite being outside the scope of this work.

Four example trading algorithms were implemented. The goal was to show how the system could be used with four different types of strategies. The Pairs trading strategy proved to be the most profitable and should definitely be considered by any quantitative trader. The algorithm created to exemplify a trend following strategy was not as profitable as expected, at least taking into account its wide usage in the trading world. Finally the Mean Reverting strategy, implemented using BB showed an uniform distribution of positive profits and the Binary Classification revealed a random nature due to the low accuracy of the classifier. It should also be noted that all the algorithms performed better in lower trading periods confirming the notion that the technical analysis approach works better in these short periods.

The proposed goals for the system were all met. The created software performs backtesting and live trading, strategies that produce trading signals were implemented together with money management, several types of data were considered when building the system, independence from brokers was partially achieved and full reports for backtests were created complying with the state of the art. Overall, this work produced both valuable research in quantitative trading and a platform for further research. Using a data science approach to produce the backtest results with IPython notebooks, including statistical analysis and machine learning in the system and the scientific approach to backtesting made this work unique.

7.2 FUTURE WORK

The evolutionary nature of Quantitative Trading makes this dissertation an unfinished work, even though the specified requirements were met. There is room for improvement of the work that was done and new features.

Firstly it would be interesting to implement a GUI to allow researchers with no programming background to use the system. This GUI should provide every tool offered by the system, as well as configuration access, graphically to the user. This could also include live reports for live trading showing at any given time the exposure or total cash available. In addition, slippage, spread or system performance metrics could be provided to the user.

The offer on machine learning models and tools, such as pre-processing tools ou feature selection tools, should also increase. The same applies to statistical analysis, in addition to the HE and ADF test, tests like the Johansen Test may be of interest to researchers.

Live traders may want to deploy the live trading system in a remote machine to trade 24/7, this creates the need of communication between the system and other platforms. One option could be implementing push notifications from the remote host to a smartphone. Other is to use Metatrader 4 to execute trading orders, working as a middleman between the brokerage firm and the live trading system. By using Metatrader 4, one can use the android or ios app to keep track of trading accounts linked to Metatrader.

The goal is then to further improve the system designed and implemented in this dissertation and made it open source to the community.

References

[1] S. J. Grossman and J. E. Stiglitz, "On the impossibility of informationally efficient markets", *The American Economic Review*, vol. 70, 1980. [Online]. Available: `http://www.jstor.org/stable/1805228`.

[2] M. Sewell, "History of the efficient market hypothesis", 2011. [Online]. Available: `http://www.cs.ucl.ac.uk/fileadmin/UCL-CS/images/Research%7B%5C_%7DStudent%7B%5C_%7DInformation/RN%7B%5C_%7D11%7B%5C_%7D04.pdf`.

[3] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules", *Journal of Financial Economics*, vol. 51, pp. 245–271, 1999. [Online]. Available: `http://finance.wharton.upenn.edu/%7B~%7Dallenf/download/Vita/genetic.pdf`.

[4] T. Hendershott and R. Riordan, "Algorithmic trading and information *", 2011. [Online]. Available: `http://faculty.haas.berkeley.edu/hender/ATInformation.pdf`.

[5] Investopedia, *Beginner's guide to quantitative trading - quantstart.* [Online]. Available: `http://www.investopedia.com/terms/q/quantitative-trading.asp` (visited on 05/08/2017).

[6] M. L. Halls-Moore, *Quantitative trading.* [Online]. Available: `https://www.quantstart.com/articles/Beginners-Guide-to-Quantitative-Trading` (visited on 05/08/2017).

[7] M. Gsell and P. Gomber, "Algorithmic trading engines versus human traders - do they behave different in securities markets?", [Online]. Available: `http://aisel.aisnet.org/ecis2009`.

[8] S. Robin, *Simulation – The practice of model development and use.* Wiley, 2004, ISBN: 0470847727.

[9] E. Chan, *How to build your own algorithmic trading business.* 2009, ISBN: 9780470284889.

[10] D. Matloff, Norm S (University of California, "Introduction to discrete-event simulation and the simpy language", *Davis, CA. Dept of Computer Science. University*, pp. 1–33, 2008, ISSN: 02750708. DOI: `10.1007/978-0-387-68612-7_10`. [Online]. Available: `http://heather.cs.ucdavis.edu/%7B~%7Dmatloff/156/PLN/DESimIntro.pdf`.

[11] M. Pidd, *Computer simulation in management science*, fourth edi, Wiley, Ed. 1998.

[12] IRMA, *Networking and Telecommunications: Concepts, Methodologies, Tools, and Applications.* IGI Global, 2010, ISBN: 9781605669861. DOI: `10.4018/978-1-60566-986-1`.

[13] Stefan Scherfke, *Discrete-event simulation with simpy*, 2014. [Online]. Available: `https://stefan.sofa-rockers.org/downloads/simpy-ep14.pdf%7B%5C#%7Dpage=5`.

[14] *Desmo-j quick overview.* [Online]. Available: `http://desmoj.sourceforge.net/overview.html` (visited on 06/19/2017).

[15] *Desmo-j quick overview.* [Online]. Available: `http://desmoj.sourceforge.net/overview.html` (visited on 06/17/2017).

[16] Plus500, *Plus500 | negoceie forex, mercadorias, cfd (contrato por diferenças), valores/ações/capitais/títulos e índices online.* [Online]. Available: `https://www.plus500.pt/` (visited on 05/10/2017).

[17] QuantLib, *Quantlib: a free/open-source library for quantitative finance.* [Online]. Available: `http://quantlib.org/index.shtml` (visited on 05/10/2017).

[18] Zipline, *Zipline — zipline 1.1.0 documentation.* [Online]. Available: `http://www.zipline.io/` (visited on 05/10/2017).

[19] TA-Lib, *Ta-lib : technical analysis library - home.* [Online]. Available: `http://www.ta-lib.org/` (visited on 05/10/2017).

[20] MetaQuotes, *Metaquotes software corp.* [Online]. Available: `https://www.metaquotes.net/` (visited on 05/10/2017).

[21] ProTrader, *Protrader — multi-asset trading platform for brokerage.* [Online]. Available: `https://protrader.com/` (visited on 05/10/2017).

[22] TradeStation, *Tradestation | online broker | trade with tradestation.* [Online]. Available: `http://www.tradestation.com/` (visited on 05/10/2017).

[23] *Metatrader 5 multi-asset trading platform.* [Online]. Available: `https://www.metatrader5.com/en` (visited on 06/19/2017).

[24] N. Baker, *Iacs seminar: quantopian*, 2015.

[25] IB, *Low-cost online trading | interactive brokers.* [Online]. Available: `https://www.interactivebrokers.com/en/home.php` (visited on 05/10/2017).

[26] John Fawcett, *Hacking your education: the next generation of students | wired.* [Online]. Available: `https://www.wired.com/insights/2013/06/hacking-your-education-the-next-generation-of-students/` (visited on 05/10/2017).

[27] *Learn forex trading at school of pipsology - babypips.com.* [Online]. Available: `https://www.babypips.com/learn/forex` (visited on 06/08/2017).

[28] *Foreign exchange.* [Online]. Available: `http://www.investopedia.com/terms/f/foreign-exchange.asp` (visited on 06/08/2017).

[29] J. Elmerraji, *Picking your first broker | investopedia.* [Online]. Available: `http://www.investopedia.com/articles/younginvestors/06/firstbroker.asp` (visited on 04/21/2017).

[30] P. Rosenstreich, *Forex Revolution: An Insider's Guide to the Real world of Foreign Exchange Tragin.* Prentice Hall, 2005.

[31] Bokeh, *Stocks.py — bokeh 0.12.5 documentation.* [Online]. Available: `http://bokeh.pydata.org/en/latest/docs/gallery/stocks.html` (visited on 05/10/2017).

[32] IPython, *Jupyter and the future of ipython — ipython.* [Online]. Available: `http://ipython.org/` (visited on 05/10/2017).

[33] Jupyter, *Project jupyter | home.* [Online]. Available: `https://jupyter.org/` (visited on 05/10/2017).

[34] Domino, *Building interactive dashboards with jupyter.* [Online]. Available: `https://blog.dominodatalab.com/interactive-dashboards-in-jupyter/` (visited on 05/10/2017).

[35] *R: the r project for statistical computing.* [Online]. Available: `https://www.r-project.org/` (visited on 06/26/2017).

[36] R. Ihaka and R. Gentleman, "A language for data analysis and graphics", *Journal of Computational and Graphical Statistics*, 1996. [Online]. Available: `https://www.stat.auckland.ac.nz/%7B~%7Dihaka/downloads/R-paper.pdf`.

[37] M. L. Halls-Moore, *Successfull Algorithmic Trading.*

[38] J. D. Farmer and S. Joshi, "The price dynamics of common trading strategies", *Journal of Economic Behavior & Organization*, vol. 49, pp. 149–171, 2002. [Online]. Available: `http://ac.els-cdn.com/S0167268102000653/1-s2.0-S0167268102000653-main.pdf?%7B%5C_%7Dtid=85a30d78-2692-11e7-a9f7-00000aacb361%7B%5C&%7Dacdnat=1492779820%7B%5C_%7D5f1deccf3767d0ea2afd367634927cc2`.

[39] J. Folger, *Pairs trading: introduction | investopedia.* [Online]. Available: `http://www.investopedia.com/university/guide-pairs-trading/` (visited on 04/20/2017).

[40] Y. Dai and Z. Yuning, "Machine learning in stock price trend forecasting", [Online]. Available: `http://cs229.stanford.edu/proj2013/DaiZhang-MachineLearningInStockPriceTrendForecasting.pdf`.

[41] B. Hurst, Y. H. Ooi, and L. H. Pedersen, "A century of evidence on trend-following investing", 2012. [Online]. Available: `www.aqr.com`.

[42] E. Seykota, *How to determine the trend*, 2016. [Online]. Available: `http://www.seykota.com/tribe/tsp/Trends/index.htm` (visited on 04/19/2017).

[43] N. S. Tobias, *Trend following stock trading system.* [Online]. Available: `http://www.stock-trading-warrior.com/trend-following-stock-trading-system.html` (visited on 05/14/2017).

[44] D. Kahneman, *Thinking, Fast and Slow by Daniel Kahneman.* 2011, p. 499.

[45] E. Chan, *Algorithmic Trading: Winning Strategies and Their Rationale*, 9. 2013, vol. 53, pp. 1689–1699, ISBN: 9788578110796. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[46] E. W. Weisstein, *Reversion to the mean.*

[47] J. Loeber, *Trading strategy: moving average mean reversion.* [Online]. Available: https://www.quantopian.com/posts/trading-strategy-moving-average-mean-reversion (visited on 05/14/2017).

[48] R. J. Shiller, *The real mandate is to bridge the wealth gap - the new york times*, 2008. [Online]. Available: http://www.nytimes.com/2008/11/09/business/09shiller.html?rref=collection%7B%5C%%7D2Ftimestopic%7B%5C%%7D2FShiller%7B%5C%%7D2C%20Robert%20J..

[49] T. J. Moskowitz and M. Grinblatt, "Do industries explain momentum?", *The Journal of The American Finance Association*, 1999.

[50] B. Bruder, J.-C. Richard, T. Roncalli, and T.-L. Dao, "Trend filtering methods for momentum strategies", 2011. [Online]. Available: http://thierry-roncalli.com/download/lwp-tf.pdf.

[51] G. P. Nason, "Stationary and non-stationary time series", [Online]. Available: http://www.cas.usf.edu/%7B~%7Dcconnor/geolsoc/html/chapter11.pdf.

[52] G. J. Miao, "High frequency and dynamic pairs trading based on statistical arbitrage using a two-stage correlation and cointegration approach", *International Journal of Economics and Finance*, vol. 6, no. 3, 2014, ISSN: 1916-9728. DOI: 10.5539/ijef.v6n3p96. [Online]. Available: http://dx.doi.org/10.5539/ijef.v6n3p96.

[53] Scalpingfx, *[mt4] statistical arbitrage – pairs trading | numbers are cool.* [Online]. Available: https://scalpingfx.wordpress.com/2013/06/12/mt4-statistical-arbitrage-pairs-trading/ (visited on 05/14/2017).

[54] K. Tseng, O. Kown, and L. C. Tjung, "Time series and neural network forecasts of daily stock", *Investment Management and Financial Innovations*, 2017. [Online]. Available: http://www.businessperspectives.org/media/zoo/applications/publishing/templates/article/assets/js/pdfjs/web/viewer.php?file=/pdfproxy.php?item%7B%5C_%7Did:4411.

[55] T. Z. Tan, C. Quek, and G. S. Ng, "Brain-inspired genetic complementary learning for stock market prediction", January, 2005, ISBN: 0-7803-9363-5. DOI: 10.1109/CEC.2005.1555027.

[56] G. J.Deboeck, *Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets.* 1994.

[57] J. B. Guerard Jr., *Introduction to Financial Forecasting in Investment Analysis*, ISBN: 978-1461452386.

[58] P.-C. Chang, C.-H. Liu, J.-L. Lin, C.-Y. Fan, and C. S. Ng, "A neural network with a case based dynamic window for stock trading prediction", *Expert Systems with Applications*, vol. 36, no. 3, pp. 6889–6898, 2009, ISSN: 09574174. DOI: 10.1016/j.eswa.2008.08.077. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2008.08.077.

[59] K. Joo Oh and K.-j. Kim, "Analyzing stock market tick data using piecewise nonlinear model", *Expert Systems with Applications 22(3):249-255*, 2002.

[60] Y. Wang, "Predicting stock price using fuzzy grey prediction system", *Expert Systems with Applications*, vol. 22, no. 1, pp. 33–38, Jan. 2002, ISSN: 09574174. DOI: 10.1016/S0957-4174(01)00047-1. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0957417401000471.

[61] A. A. BAASHER and M. W. FAKHR, "Forex trend classification using machine learning techniques", [Online]. Available: http://wseas.us/e-library/conferences/2011/Penang/ACRE/ACRE-05.pdf.

[62] L. C. Martinez, D. N. da Hora, J. R. de M Palotti, W. Meira Jr, and G. L. Pappa, "From an artificial neural network to a stock market day-trading system: a case study on the bm&f bovespa", [Online]. Available: http://homepages.dcc.ufmg.br/%7B~%7Dglpappa/papers/Conegundesetal-2009-IJCNN.pdf.

[63] *Sklearn.feature_selection.rfe — scikit-learn 0.18.1 documentation.* [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.feature%7B%5C_%7Dselection.RFE.html (visited on 05/15/2017).

[64] T. Balch, *Machine learning for trading by georgia tech.* [Online]. Available: https://www.udacity.com/course/machine-learning-for-trading--ud501.

[65] Investopedia, *Tick.* [Online]. Available: http://www.investopedia.com/terms/t/tick.asp (visited on 04/30/2017).

[66] ——, *Bid-ask spread*, 2017. [Online]. Available: http://www.morningstar.com/InvGlossary/bid-ask-spread.aspx (visited on 04/30/2017).

[67] P. Domingos, "A unified bias-variance decomposition and its applications", [Online]. Available: http://homes.cs.washington.edu/%7B~%7Dpedrod/papers/mlc00a.pdf.

[68] ——, "A few useful things to know about machine learning", *Communications of the ACM*, vol. 55, no. 79, 2012. DOI: 10.1145/2347736.2347755. [Online]. Available: http://delivery.acm.org/10.1145/2350000/

2347755/p78-domingos.pdf?ip=193.137.168.32%7B%5C&%7Did=2347755%7B%5C&%7Dacc=PUBLIC%7B%5C&%7Dkey=2E5699D25B4FE09E.861C198C983DE13B.4D4702B0C3E38B35.4D4702B0C3E38B35%7B%5C&%7DCFID=931470817%7B%5C&%7DCFTOKEN=32057196%7B%5C&%7D%7B%5C_%7D%7B%5C_%7Dacm%7B%5C_%7D%7B%5C_%7D=1493746720%7B%5C_%7D338603de04d3cb2b8cdc5.

[69] Investopedia, *Drawdown.* [Online]. Available: http://www.investopedia.com/terms/d/drawdown.asp (visited on 05/11/2017).

[70] ——, *Sharpe ratio.* [Online]. Available: http://www.investopedia.com/terms/s/sharperatio.asp (visited on 05/11/2017).

[71] *Pandas-datareader — pandas-datareader 0.1 documentation.* [Online]. Available: https://pandas-datareader.readthedocs.io/en/latest/ (visited on 05/21/2017).

[72] *Pandas.read_csv — pandas 0.20.1 documentation.* [Online]. Available: http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read%7B%5C_%7Dcsv.html (visited on 05/21/2017).

[73] *Pymongo 3.4.0 documentation — pymongo 3.4.0 documentation.* [Online]. Available: https://api.mongodb.com/python/current/ (visited on 05/21/2017).

[74] *Https://github.com/manahl/arctic.* [Online]. Available: https://github.com/manahl/arctic (visited on 05/17/2017).

[75] C. A. Peters, "Statistics for analysis of experimental data", [Online]. Available: https://www.princeton.edu/%7B~%7Dcap/AEESP%7B%5C_%7DStatchap%7B%5C_%7DPeters.pdf.

[76] T. Starke, *Some code from ernie chan's new book implemented in python.* [Online]. Available: https://www.quantopian.com/posts/some-code-from-ernie-chans-new-book-implemented-in-python (visited on 05/07/2017).

[77] D. Fernandez, *Simple trend following : using an ema, rsi based daily system | mechanical forex.* [Online]. Available: http://mechanicalforex.com/2010/07/simple-trend-following-using-ema-rsi.html (visited on 05/05/2017).

[78] G. Welch and G. Bishop, "An introduction to the kalman filter", 2006. [Online]. Available: https://www.cs.unc.edu/%7B~%7Dwelch/media/pdf/kalman%7B%5C_%7Dintro.pdf.

[79] *Psutil documentation — psutil 5.2.0 documentation.* [Online]. Available: https://pythonhosted.org/psutil/ (visited on 05/25/2017).

[80] *New zealand in profile: 2014.* [Online]. Available: http://www.stats.govt.nz/browse%7B%5C_%7Dfor%7B%5C_%7Dstats/snapshots-of-nz/nz-in-profile-2014/main-trading-partners.aspx (visited on 05/22/2017).