



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Escola Superior d'Enginyeries Industrial,  
Aeroespacial i Audiovisual de Terrassa

---

MASTER'S DEGREE THESIS  
MASTER'S DEGREE IN MANAGEMENT ENGINEERING

---

DEVELOPMENT OF A  
CRYPTOCURRENCY BOT

---

REPORT

January 12<sup>th</sup>, 2023

*Author's full name:*

José Manuel Rodríguez  
Lomeña

*Mentor's full name:*

Rafel Amer Ramon

*Call for delivery:*

2022/23 - 1



# Acknowledgments

The development of this project would not have been possible with the support of some people. These lines are to express them my gratitude.

Firstly, I want to thank my project director, Rafel Amer, whose knowledge and passion for cryptocurrency has helped me in the crossroads that have been presenting during all these months. You are the definition of what a great tutor is.

I also gratefully acknowledge the support of my friends Sergio Sánchez and Guillem Soriano. They helped me whenever I have needed it with code programming. I am sure that your desire to learn and innovate will take you very far.

Finally, I want to thank all my loved ones for the constant support and patience. All my work is also yours.



# Declaration of Honour



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Escola Superior d'Enginyeries Industrial,  
Aeroespacial i Audiovisual de Terrassa

I declare that,

- the work in this Master's Degree Thesis is completely my own work,
- no part of this Master's Degree Thesis is taken from other people's work without giving them credit,
- all references have been clearly cited,

I understand that an infringement of this declaration leaves me subject to the foreseen disciplinary actions by *The Universitat Politècnica de Catalunya - BarcelonaTECH*.

José Manuel Rodríguez Lomeña

Student Name

Signature

January 12, 2023

Date

Title of the Thesis

Development of a Cryptocurrency Bot



# Contents

<b>Nomenclature</b>	<b>xiii</b>
<b>Abstract</b>	<b>xv</b>
<b>1 Aim</b>	<b>1</b>
<b>2 Scope</b>	<b>3</b>
2.1 In Scope . . . . .	3
2.2 Out of Scope . . . . .	3
<b>3 Justification</b>	<b>5</b>
<b>4 Introduction</b>	<b>9</b>
<b>5 Budget</b>	<b>11</b>
5.1 Breakdown of Unit Costs . . . . .	11
5.2 Total Budget . . . . .	13
<b>6 Environmental Impact</b>	<b>15</b>
<b>7 State of the Art</b>	<b>17</b>
7.1 Capfolio . . . . .	17
7.2 3Commas . . . . .	18
7.3 CCXT . . . . .	18
7.4 Blackbird . . . . .	18
7.5 StockSharp . . . . .	19
7.6 Freqtrade . . . . .	19
7.7 CryptoSignal . . . . .	20
7.8 Ctubio . . . . .	20
7.9 Catalyst . . . . .	20
7.10 Using Machine Learning . . . . .	20
<b>8 Technical Analysis of the Financial Markets</b>	<b>21</b>
8.1 Philosophy of Technical Analysis . . . . .	21
8.2 Dow Theory . . . . .	22
8.3 Basic Concepts of Trends . . . . .	25
8.4 Volume . . . . .	30
8.5 Chart Patterns . . . . .	31

8.5.1	Reversal Patterns . . . . .	32
8.5.1.1	The Head and Shoulders . . . . .	32
8.5.1.2	Triple Tops and Bottoms . . . . .	34
8.5.1.3	Double Tops and Bottoms . . . . .	35
8.5.2	Continuation Patterns . . . . .	36
8.5.2.1	Triangles . . . . .	36
8.5.2.2	Broadening Formation . . . . .	38
8.5.2.3	Flags and Pennants . . . . .	38
8.5.2.4	Wedge Formation . . . . .	40
8.5.2.5	Rectangle Formation . . . . .	41
8.5.3	Confirmation and Divergence . . . . .	42
8.6	Chart Construction . . . . .	43
8.6.1	Candlestick Charting . . . . .	44
8.6.2	Arithmetic versus Logarithmic Scale . . . . .	45
8.6.3	Long Term Charts . . . . .	46
8.7	Moving Averages . . . . .	46
8.7.1	The Simple Moving Average . . . . .	47
8.7.2	The Linearly Weighted Moving Average . . . . .	47
8.7.3	The Exponentially Smoothed Moving Average . . . . .	48
8.7.4	Using Two Moving Averages . . . . .	48
8.8	Oscillators . . . . .	50
8.8.1	Measuring Momentum . . . . .	51
8.8.2	Oscillator Using Two Moving Averages . . . . .	52
8.8.3	The Relative Strength Index (RSI) . . . . .	53
8.8.4	The Importance of Trend . . . . .	54
8.8.5	When Oscillators Are Most Useful . . . . .	54
<b>9</b>	<b>Introduction to Cryptocurrency</b>	<b>55</b>
9.1	Cryptocurrency . . . . .	55
9.1.1	Meaning and Definition . . . . .	55
9.1.2	Blockchain . . . . .	56
9.1.2.1	Transaction Process . . . . .	56
9.1.3	Cryptography . . . . .	58
9.1.3.1	Cryptographic Algorithms . . . . .	58
9.1.4	Cryptocurrency Wallet . . . . .	62
9.1.4.1	Types of Cryptocurrency Wallets . . . . .	63
9.1.5	Proof-of-Work against Proof-of-Stake . . . . .	64
9.1.5.1	Proof-of-Work . . . . .	64
9.1.5.2	Proof-of-Stake . . . . .	65
9.1.5.3	PoW & PoS Comparison . . . . .	65
9.1.6	History . . . . .	66
9.1.6.1	The Idea for Cryptocurrency . . . . .	67
9.1.6.2	The Beginning (2008-2010) . . . . .	67
9.1.6.3	The Market Begins to Form (2010-2014) . . . . .	67
9.1.6.4	Scams Dominate Headlines (2014-2016) . . . . .	68
9.1.6.5	Bitcoin Ascends to Worldwide Phenomenon (2016-2018)	68

9.1.6.6	Bust and Recovery, and Bust and Recovery (2018-Present) . . . . .	69
9.1.7	Types of cryptocurrency . . . . .	70
9.1.7.1	Coins . . . . .	70
9.1.7.2	Tokens . . . . .	71
9.1.7.3	Largest Cryptocurrencies by Market Cap . . . . .	72
9.1.8	Cryptocurrency Exchange . . . . .	74
9.1.8.1	Centralized Cryptocurrency Exchange . . . . .	74
9.1.8.2	Decentralized Cryptocurrency Exchange . . . . .	75
9.1.8.3	CEX against DEX . . . . .	75
9.2	Cryptocurrency Bot . . . . .	77
9.2.1	Backtesting . . . . .	77
9.2.2	Yahoo! Finance . . . . .	78
<b>10</b>	<b>Machine Learning and Statistics</b>	<b>79</b>
10.1	Machine Learning . . . . .	79
10.1.1	Neural Networks . . . . .	80
10.1.2	Recurrent Neural Network . . . . .	81
10.1.3	Long Short-Term Memory Network . . . . .	81
10.1.4	Modular Neural Network . . . . .	82
10.2	Statistical Concepts . . . . .	82
10.2.1	ARIMA model . . . . .	83
10.2.2	Combination Without Repetition . . . . .	84
<b>11</b>	<b>Python Libraries</b>	<b>85</b>
11.1	General Libraries . . . . .	85
11.2	Data Analysis Libraries . . . . .	86
11.3	Visualization Libraries . . . . .	87
11.4	Trading Libraries . . . . .	88
11.5	Machine Learning Libraries . . . . .	88
<b>12</b>	<b>Studies</b>	<b>89</b>
12.1	Simulation on Random Trade Closures . . . . .	89
12.2	Analysis of Strategy Variables . . . . .	92
<b>13</b>	<b>Software Description</b>	<b>97</b>
13.1	Atom . . . . .	97
13.2	GitHub Desktop . . . . .	98
13.3	Command Prompt . . . . .	98
<b>14</b>	<b>Software Implementation</b>	<b>99</b>
14.1	Requirements of the Algorithm . . . . .	99
14.2	Algorithm . . . . .	100
14.2.1	Bot . . . . .	100
14.2.1.1	botData . . . . .	101
14.2.1.2	botCore . . . . .	102
14.2.1.3	botMain . . . . .	107

14.2.2 Backtesting . . . . .	109
14.2.2.1 settings . . . . .	110
14.2.2.2 sizers . . . . .	111
14.2.2.3 downloadData . . . . .	111
14.2.2.4 strategy . . . . .	113
14.2.2.5 backtestingCore . . . . .	114
14.2.2.6 backtestingMain . . . . .	119
<b>15 Results</b>	<b>121</b>
15.1 Backtesting for a Frequency of 15 Minutes . . . . .	122
15.2 Backtesting for a Frequency of 5 Minutes . . . . .	122
15.3 Backtesting for a Frequency of 30 Minutes . . . . .	123
15.4 Strategy Analysis . . . . .	124
<b>16 Conclusions</b>	<b>125</b>
<b>17 Future Work</b>	<b>127</b>
<b>Appendices</b>	<b>128</b>
<b>A Bot Code</b>	<b>129</b>
A.1 botData . . . . .	129
A.2 botCore . . . . .	130
A.3 botMain . . . . .	134
A.4 settings . . . . .	137
A.5 sizers . . . . .	137
A.6 downloadData . . . . .	138
A.7 strategy . . . . .	141
A.8 backtestingCore . . . . .	142
A.9 backtestingMain . . . . .	147
<b>B Studies Codes</b>	<b>149</b>
B.1 Simulation on Random Trade Closures . . . . .	149
B.2 Combination Without Repetition . . . . .	152
B.3 ROI vs Number of Trades for $\omega_p$ Range . . . . .	153
B.4 ROI vs Number of Trades for $p$ Range . . . . .	154
B.5 ROI vs Number of Trades for $A$ Range . . . . .	155

# List of Figures

Figures from articles, books, these or other sources are cited in their respective caption. In any other case, they correspond to figures of own elaboration.

8.1	Failure Swing . . . . .	24
8.2	Nonfailure Swing . . . . .	24
8.3	Failure Swing Bottom . . . . .	24
8.4	Nonfailure Swing Bottom . . . . .	24
8.5	Example of an uptrend . . . . .	25
8.6	Example of a downtrend . . . . .	25
8.7	Example of a sideways trend . . . . .	26
8.8	Example the three degrees of trend . . . . .	26
8.9	Support and resistance levels in uptrend . . . . .	27
8.10	Example of an up trendline . . . . .	28
8.11	Example of an down trendline . . . . .	28
8.12	Breaking of an up trendline . . . . .	29
8.13	Breaking of a down trendline . . . . .	29
8.14	Example of an uptrend channel . . . . .	30
8.15	Example of a downtrend channel . . . . .	30
8.16	Example of volume bars . . . . .	31
8.17	Example of a head and shoulders top . . . . .	33
8.18	Example of an inverse head and shoulders . . . . .	34
8.19	Example of a triple top . . . . .	34
8.20	Example of a triple bottom . . . . .	35
8.21	Example of a double top . . . . .	35
8.22	Example of a double bottom . . . . .	35
8.23	Example of a bullish symmetrical triangle . . . . .	36
8.24	Example of an ascending triangle . . . . .	37
8.25	Example of a descending triangle . . . . .	37
8.26	Example of a broadening top . . . . .	38
8.27	Example of a bullish flag . . . . .	39
8.28	Example of a bullish pennant . . . . .	40
8.29	Example of a bullish falling wedge . . . . .	41
8.30	Example of a bearish wedge . . . . .	41
8.31	Example of a bullish falling rectangle . . . . .	42
8.32	Example of a bearish rectangle . . . . .	42
8.33	Example of a bar chart . . . . .	43
8.34	Example of line chart . . . . .	43
8.35	Example of candlestick . . . . .	44

8.36 Bar chart versus candlestick chart . . . . .	44
8.37 Candlesticks for long and short days . . . . .	45
8.38 A comparison of an arithmetic and logarithmic scale . . . . .	46
8.39 A 10 day moving average applied to a daily bar chart . . . . .	47
8.40 Exponential moving average versus simple moving average . . . . .	48
8.41 One moving average as trading signal . . . . .	49
8.42 Example of the double crossover method . . . . .	50
8.43 Example of momentum use . . . . .	51
8.44 Example of using momentum as indicator . . . . .	52
8.45 Example of a histogram using two moving averages . . . . .	53
8.46 Example of RSI oscillator . . . . .	54
9.1 Transaction of cryptocurrency in blockchain . . . . .	56
9.2 Cryptography jargon . . . . .	58
9.3 Symmetric encryption . . . . .	59
9.4 Asymmetric encryption . . . . .	60
9.5 Hash functions . . . . .	61
9.6 KeepKey Hardware Wallet . . . . .	63
9.7 Cryptocurrency miners . . . . .	66
9.8 Evolution of BTC-USD price . . . . .	69
9.9 Most common coin symbols . . . . .	71
10.1 Neural networks layers' diagram . . . . .	80
10.2 LSTM Network diagram . . . . .	81
10.3 Modular Neural Network diagram . . . . .	82
12.1 Simulation on random trade closures . . . . .	91
12.2 ROI vs number of trades for $\omega_p$ range . . . . .	94
12.3 ROI vs number of trades for $p$ range and $\omega_p = 0.498$ . . . . .	95
12.4 ROI vs number of trades for $p$ range and $\omega_p = 0.5008$ . . . . .	95
12.5 ROI vs number of trades for $A$ range . . . . .	96
13.1 Atom environment . . . . .	97
13.2 Windows Command Prompt . . . . .	98
14.1 Diagram of the trading bot . . . . .	100
14.2 Diagram of the backtesting . . . . .	110

# List of Tables

5.1	Electricity costs . . . . .	13
5.2	Budget of the project . . . . .	13
9.1	PoW & PoS comparison . . . . .	66
9.2	Cryptocurrency with more market cap . . . . .	72
15.1	Backtesting results for a frequency of 15 minutes . . . . .	122
15.2	Backtesting results for a frequency of 5 minutes . . . . .	123
15.3	Backtesting results for a frequency of 30 minutes . . . . .	123

*LIST OF TABLES*

---

# Nomenclature

The next list describes the symbols used within the body of the document:

ADA Cardano

AI Artificial Intelligence

ANN Artificial Neural Network

ARIMA Autoregressive Integrated Moving-Average

ATH All-Time High

BNB Binance Coin

BTC Bitcoin

CCXT Cryptocurrency Exchange Trading Library

CEX Centralized Cryptocurrency Exchange

CSV Comma Separated Values

dApps Decentralized Applications

DeFi Decentralized Finance

DEX Decentralized Cryptocurrency Exchange

DLT Distributed Ledger Technology

ETH Ethereum

FOMO Fear Of Missing Out

GARCH Generalized Autoregressive Conditional Heteroscedasticity

GUI Graphical User Interface

ICO Initial Coin Offering

IPO Initial Public Offering

KYC Know-Your-Customer

LSTM Long-Short Term Memory

LTC Litecoin

## *NOMENCLATURE*

---

MACD Moving Average Convergence/Divergence

ML Machine Learning

NFT Non-Fungible Token

NumPy Numerical Python

PKC Public Key Cryptography

PoS Proof-of-Stake

PoW Proof-of-Work

RNN Recurrent Neural Network

ROI Return Of Investment

RSI Relative Strength Index

SKC Secret Key Cryptography

SNN Simulated Neural Network

USD United States Dollar

USDC USD Coin

USDT Tether

XRP Ripple

XTZ Tezos

# Abstract

As an emerging market and research direction, cryptocurrencies and cryptocurrency trading have seen considerable progress and a notable upturn in interest and activity, even entering the market people without experience or sufficient knowledge. In addition, the tremendous volatility and the fact that this market never closes make the human factor affect crypto asset trading too much.

Hence, in this project a cryptocurrency trading bot is developed. To be exact, the algorithm consists of two distinguishable parts: the bot itself and the backtesting. Notwithstanding that both parts departs from the analysis of financial markets in general, and cryptocurrencies in particular, both present clear differences in terms of code and pretext.

On the one hand, the bot's algorithm is used to trade in reality, specifically through the Binance exchange. Here the user plays risks their monetary capital. On the other hand, backtesting consists of verifying the trading strategy based on historical data. Backtesting serves, then, as validation of the strategy to be followed by the bot.

Thus, all the necessary fundamentals to understand both the general cryptocurrency context and technical analysis relevant concepts are presented, along with a detailed explanation of the implemented algorithm and a proper analysis of the obtained results. Finally, further code improvements and new ideas to develop in the future are suggested, apart from presenting the code developed.



# **Chapter 1**

## **Aim**

Since their inception, cryptocurrency have been quickly evolving to the point of being a notorious asset nowadays. However, this breakneck progression is accompanied by a technical complication for both their understanding and their investment, which no longer makes them so engrossing for them to be acquired by everyone.

Although the constant evolution of technology has meant that new tools to ease the cryptocurrency investment have emerged, these tools usually entail large fees for the small investor. In fact, there are a lot of cases where platforms that provide this type of tools are only trying to make money through these fees, and they do not really care about their client's results. That is why this market is still reluctant for those who do not fully understand it.

Despite of the above, it is doubtless that this kind of tools are especially helpful, since they separate the mental factor of the investor from the investment itself, in addition to the fact that it does not involve having to be constantly aware of it.

With all of the above, the aim of this project is to develop a cryptocurrency trading bot programmed in Phyton. To achieve this, implementation of an algorithm connected with Binance is carried out.



# Chapter 2

## Scope

This chapter consists of a brief explanation of what the lecturer will find (Sec. 2.1 In Scope) and will not find (Sec. 2.2 Out of Scope) in this project.

### 2.1 In Scope

The points which this project covers are:

- Budget and environmental impact aspects.
- State of the art of cryptocurrency trading bots.
- Theoretical introduction to technical analysis of the financial markets.
- Theoretical introduction to cryptocurrency.
- Query of information for statistical and Machine Learning techniques.
- Briefly description of used Python libraries.
- Several studies on the bet trade percentage.
- Description of the used software.
- Study of the requirements which allow implementing an adequate algorithm.
- Detailed explanation of the bot and backtesting algorithms.
- Analysis of obtained data results for the backtesting of one specific trading strategy.
- Conclusions and future work.

### 2.2 Out of Scope

The points that this project does not cover are:

- Development and implementation of more than one trading strategy.

- Real results of the trading bot algorithm (trading in reality).
- Exhaustive study on trading strategy development.
- Analysis or implementation of any kind of strategy to trade in the stocks market.

# Chapter 3

## Justification

In recent years, the tendency of the number of financial institutions and private individuals to include cryptocurrencies in their portfolios has accelerated. Cryptocurrency is playing an increasingly important role in reshaping the financial system due to its growing popular appeal and merchant acceptance. While many people are making investments in this recent market, the dynamical features, uncertainty, the predictability of cryptocurrency are still mostly unknown, which dramatically risk the investments.

A lot of risk factors appear within cryptocurrency trading. Now, there are mainly five that really affect particular individuals who trade: the illusion of control, social learning and reinforcement, preoccupation, fear of missing out, and anticipated regret.

Despite the fact that skill and strategy can make a difference to outcomes, cryptocurrency trading offers many opportunities for people to overestimate the role that applying specific types of knowledge might play in these outcomes and, conversely, they do not realize that their strategy has a clear slope in the luck and chance of what the market dictates. **The illusion of control**, defined as a subjective overestimation of the objective ability to exert control (Langer, 1975 [44]), is known to be a common feature of gambling (Wood & Clapham, 2005 [58]). The effect of this risk is likely to be bolstered by other heuristics and biases, including: biased or self-serving attributions (outcomes due to personal action rather than external factors); hindsight bias (the outcome is seen as being hypothesised all along); and, the hot-hand fallacy (perceptions of predictable momentum shifts or winning periods).

Many of these effects are likely to be strong during favourable market conditions. As a result, traders will rarely be wrong in their choices and most decisions will be positively reinforced. Traders will therefore, often falsely, infer contingency between their actions and positive outcomes, an effect which is known to be stronger when the probability of reinforcement is high (Blanco, 2011 [4]). As a result, traders may gain a sense of invincibility or perception that they cannot lose and this may contribute to greater risk taking, for example: speculation of large amounts in just one speculative coin; not planning for strategies to exit the market at the right time; or, moving money from a more balanced portfolio towards purchasing riskier altcoins.

Crypto trading has also emerged during the era of social media. This has led to the emergence of a strong social media culture of crypto advisors or influencers in platforms such as *Youtube*. Searching online quickly shows that it is possible to find at least one positive endorsement of at least one major coin. Some of these arise from what appear to be more experienced and well-informed sources, but there are many others that are entirely speculative, ill-informed, and potentially misleading because they leave out key information [57].

Although similar promotional information has historically been available concerning conventional shares, the volume of material, the interactivity and superficial confidentiality of the information (i.e., the sense that one is in the know first, or is getting some special insights) is much stronger. Promoters of particular coins can show evidence of how much money they have already earned from buying in very low, and they can use graphics with great effect to show the anticipated growth. This can serve to create a sense of urgency and a need for immediate action. It also encourages a culture of mutual **social learning and reinforcement**, in which followers of channels seek to promote their successes, while also reading about the gains scored by others.

**Preoccupation** is a recognised feature of most major conceptual models of addiction (Browne & Rockloff, 2019 [7]). Those who engage excessively in a particular activity often find it difficult to disengage from the activity. They may continuously think about the activity (preoccupation) and prioritize the activity ahead of other essential responsibilities. Crypto trading would appear to be an activity that has the potential to be highly absorbing. Like day-trading, it involves regular scrutiny of price movements, news and other online media about coin-related developments, the need to make regular buy and sell decisions and research into the different coins. However, because crypto markets operate continuously, it is possible for people to be engaged with the activity at any hour of the day. This creates the potential for crypto trading to absorb a considerable amount of time and potentially with greater risk of disruptions to sleep and other daily commitments.

One of the strongest psychological factors that appears to influence crypto-trading is the **Fear Of Missing Out** (FOMO). Traders are confronted with displays of hundreds of coins. Some of them, they already own; others they do not. If one which they have purchased is going up rapidly, they may regret having not made a larger investment. If another unpurchased coin is going up which they had previously considered, they feel annoyed for having missed out on the opportunity. Perhaps most problematically of all is the situation, when they observe a ‘green screen of numbers’. The market is going up and they feel compelled to be part of the action. They purchase a coin when it has reached a short-term peak, only to watch the price fall soon afterwards. FOMO also applies to sell decisions. When altcoins, in particular, have rapidly increased in price, there is always the prospect that the rise might continue. Instead of taking the profit, the person starts to dream of what they might purchase if the price increases even more, but is then unprepared when the price falls when the bull-run ends [57].

Cognitive psychology has recognised for some time that many decisions are based

---

on the desire to minimise **anticipated regret** (Miller & Taylor, 1995 [45]). One of the central findings in this area is that acts of commission (doing something) usually led to stronger feelings of regret than acts of omission (not doing something). This asymmetry has been used to explain why people are reluctant to sell shares which they have held for some time. Acts of commission involving regret in trading would include decisions where a coin holding is sold, only for it to rapidly appreciate in value. Acts of omission involve situations where ultimately successful stocks or coins were forgone (not purchased) for other investment decisions.

It is undoubtedly that both acts are likely to cause significant regret in trading and that this may potentially be a risk factor that is strongly observed in this activity. So much so that crypto market allows traders to observe the folly of their decisions over a long period of time: what they missed out on (a coin that has considerably increase in value in last six months), and what coins they sold too early. For this reason, the act of commission and omission effects are both likely to be very strong.

Hence, there is no doubt that cryptocurrency trading carries with it a series of psychological factors that really affect the individual, and in turn mean that only the most experienced and skilled end up obtaining a net return when trading.

All of the above leads then to the need of finding an alternative method to trade in the crypto market. Specifically, the idea is to develop a cryptocurrency trading bot based on technical analysis and Machine Learning. This would allow operating by leaving aside psychological human factors, and consequently obtaining net profitability more easily.

Thus, it would be necessary to study how percentage of the money is invested in each trade, to select the exchange to trade on and to connect it with the algorithm developed, to define and implement a trading strategy, and to validate this strategy by means of historical data.



# Chapter 4

## Introduction

This last year master's degree thesis is an academic project developed in order to get a tough basis in the field of automated cryptocurrency trading. Hence, the knowledge obtained is used for the implementation of a bot which would allow trading to be made in reality.

In this line, it is notorious to remark that regardless of this volume only aims to satisfy the academic requirements that a last year master's degree thesis demands, it is imperative to leave proof of the author's prior knowledge on the field. This previous knowledge about cryptocurrencies, the Binance exchange, technical analysis of financial markets, object-oriented programming and Machine Learning, have allowed this project to be carried out in the context of a master's degree thesis.

With that, what follows is a general explanation of what the lecturer will read, focusing on giving an overall idea of how the project itself is presented.

First of all, Chap. 5 and Chap. 6 are related to the estimation of the budget and the environmental impact of this project. They serve to give the reader an idea of the expenses and the carbon footprint generated.

In Chap. 7, a state of the art description is made, focusing on nine different cryptocurrency trading bots. They all served as a source of inspiration for the development of the algorithm implemented in this project.

From Chap. 8 to Chap. 11, both included, the fundamentals in which the algorithm are based on are presented. Chap. 8 states constancy of the actual techniques and tools used in technical analysis. Chap. 9 serves as an introduction to cryptocurrencies and trading bots, and all the concepts that come with them. Later in Chap. 10, statistical and Machine Learning techniques applied to cryptocurrencies are presented. At turn, Chap. 11 states constancy of the Python libraries used to develop the trading bot code.

The proposal of Chap. 12 is to present some relevant studies whose results are needed to decide what percentage of the total capital is risked in each trade. First of all, a code based on repetition and on random trade closures is simulated. After that,

a mathematical development on that percentage is carried out, focusing on some variables of interest.

Chap. 13 describes the software used in this project. In turn, Chap. 14 is related to the software implementation, providing a study of the requirements that the algorithm must accomplish and detailing its development.

In Chap. 15 are presented the backtesting results, which serves to validate the trading strategy which would be followed by the bot. The, Chap. 16 presents the conclusions of the whole project, and Chap. 17 aims to leave proof of some comments and indications to future developments.

Finally, Appendix A and Appendix B form the appendix of the project. They are respectively related to the trading bot algorithm, to the code developed in Chap. 12 to carry out the studies on the bet trade percentage.

# Chapter 5

## Budget

The proposal of this chapter is to collect the expenses generated for the realisation of the project, in such a way that the budget of the same can be estimated. Before that, the unit cost of all expenses is calculated, taking into account the depreciation of physical assets.

### 5.1 Breakdown of Unit Costs

Due to the depreciation that physical assets suffer over time, the amortisation cost is calculated as:

$$a = \frac{V_P - V_R}{n_a} \quad (5.1)$$

$$r_h = \frac{a}{n_h} \quad (5.2)$$

Where:

- $a$ : amortisation [€/year]
- $V_P$ : purchase value [€]
- $V_R$ : residual value at the end of the amortisation period [€]
- $n_a$ : amortisation period [years]
- $r_h$ : hourly rate [€/h]
- $n_h$ : working time per year [hours/year]

Hence, for the calculation of the working time per year, 40 hours of weekly work are considered during the 45 non-holiday weeks of each year, which make a total of:

$$n_h = 40 \frac{\text{hours}}{\text{week}} \cdot 45 \frac{\text{weeks}}{\text{year}} = 1800 \frac{\text{hours}}{\text{year}}$$

## Computer

The personal laptop used during the project is a LENOVO Ideapad330, whose purchase value  $V_P$  is 680€, over an amortisation period of 5 years, and a residual value  $V_R$  of 20%.

$$a = \frac{680 - 136}{5} = 108.8\text{€/year}$$

$$r_h = \frac{108.8}{1800} = 0.060\text{€/h}$$

## Software

Secondly, the license price of the used programs has to be included. All of them are detailed in the following list.

- Microsoft Office 2010. Computing tool. Used for analyzing data. License price: 126€/year.

$$r_h = \frac{126}{1800} = 0.070\text{€/h}$$

- GitHub Desktop. Cloud-based repository. Used for code control version. License price: free.
- Atom. Programming tool. Used for editing code. License price: free.
- MATLAB R2021b. Engineering tool. Used for developing studies. License price: free (college agreement).
- Mendeley Desktop. Referencing tool. Used for referencing. License price: free (college agreement).
- L<sup>A</sup>T<sub>E</sub>X, latex text editor. Used for the writing of the report. License price: free.
- Adobe Reader. Reader. Used for reading articles. License price: free.
- Google Chrome. Web browser. Used for the query of information. License price: free.

## Personnel Cost

The total time dedication to carry out the present project is a total of 300 hours. Here, an engineer's salary is considered to be 25€ per hour.

## Electricity

It is considered both the electricity needed to charge the laptop and due to the light.

The laptop has been used full time during the development of the project, thus 300 hours. Since its autonomy is 6 hours, 50 charges have been made. Considering that the computer is fully charged in 3.5 hours, a total of 175 hours is obtained.

On the other hand, it is estimated that the use of the light has been only required for 75% of the time, i.e. 225 hours.

With that, Table (5.1) presents each of the above costs, considering an electricity average cost of 0.25€/kWh<sup>1</sup>.

Table 5.1: Electricity costs

Concept	Power [kW]	Hours [h]	Energy [kWh]	Cost [€]
Laptop	0.12	175	21	5.25
Light	0.24	225	54	13.5
<b>TOTAL COST</b>				<b>18.75</b>

## 5.2 Total Budget

In view of the above results, Table (5.2) details the costs in each field and provides the budget of the project.

Table 5.2: Budget of the project

Concept	Period [h]	$r_h$ [€/h]	Cost [€]
Computer	300	0.060	18
Microsoft Office	50	0.070	3.5
Personnel cost	300	25	7,500
Electricity			18.75
<b>TOTAL COST</b>			<b>7,540.25</b>

At that point, it is essential to note that the personnel cost has not had to be assumed, as it is the author himself who carries out the project. Therefore, the actual spending has been 40.25€.

<sup>1</sup>This price has been obtained from the monthly and daily averages of the development period of the project, in the place of residence of the author.



# Chapter 6

## Environmental Impact

The study of environmental impact is a fundamental point in the development of any project. Specifically, it is presented the carbon footprint of this project. In other words, it is calculated how many kilos of  $CO_2$  are dispersed by the atmosphere because of this project.

As commented in Chap. 5, this project has been carried out only by means of a laptop. Although actually its manufacture leaves an environmental footprint, it is not considered in the impact of this project owing to its use time is negligible in relation to their useful life. Hence, what does have a clear influence on the environment is spent electricity.

From Table (5.1), it can be seen that the total electrical energy that has been expanded during the whole project is  $75 \text{ kWh}$ . Since the electricity company that supplies the author's house is Endesa Energía S.A., and according to [10] its impact carbon factor is of  $0.38 \text{ kg}$  of  $CO_2$  per  $\text{kWh}$  of electrical energy:

$$M_{CO_2} = 0.38 \frac{\text{kg } CO_2}{\text{1 kWh}} \cdot 75 \text{ kWh} = 28.5 \text{ kg}$$

Taking into account that electricity due to the light is of  $54 \text{ kWh}$ , which supposes a 72% of the total, it is clearly that most of the environmental footprint is caused by the use of light. Hence, it would be captivating to reduce that impact by using sunlight during the day. To achieve this, the workspace would have to be adapted and take advantage of the natural light that enters through the windows.



# Chapter 7

## State of the Art

In this chapter is presented a brief state of the art description regarding to cryptocurrency trading bots<sup>1</sup>. Specifically, nine different trading bots of the main ones existing in the market are presented. Notwithstanding that the intention is mainly to illustrate which trading bots operates nowadays, they also serve as a source of inspiration for the present project.

### 7.1 Capfolio

Capfolio is a next generation cryptocurrency trading platform with an all-in-one solution for beginner, intermediate and advanced cryptocurrency traders. It allows testing out the crypto markets, cloning leaders, or even building a sophisticated trading algorithm [8].

Capfolio is a secure, low-latency, institutional quality platform, in which a wide range of tools is available: from data gathering and signal tracking, through backtesting<sup>2</sup> and strategy development, to automated trading. This platform was built by crypto traders for crypto traders of all levels, aiming at offering everyone a straightforward access to the cryptocurrency markets.

Among the great diversity of tools Capfolio offers, backtesting stands out, since there is no need for complex programming or in-depth statistical knowledge, which makes backtesting easier than usual. Furthermore, it is also worth noting that it allows operating on five different exchanges<sup>3</sup> [8].

---

<sup>1</sup>The reader unfamiliar with cryptocurrency trading bots is recommended to read Sec. 9.2 Cryptocurrency Bot.

<sup>2</sup>The reader unfamiliar with *backtesting* is recommended to read Sec. 9.2.1 Backtesting.

<sup>3</sup>The reader unfamiliar with cryptocurrency exchanges is recommended to read Sec. 9.1.8 Cryptocurrency Exchange.

## 7.2 3Commas

3Commas helps users grow their crypto investments using a feature-rich terminal and proven automated bots that operate on 17 major exchanges [1].

3Commas helps traders win regardless of market conditions. offering always a trading strategy that can profit from it. 3Commas bots happen to be really good at reducing average acquisition costs, directly increasing the profit margins from each trade.

Apart from its security, based on the fact that it only works with exchanges using API keys<sup>4</sup>, and an innovative trade automation focused on executing trading strategies at scale, 3Commas offers the possibility of taking profit and stop-loss orders at the same time.

## 7.3 CCXT

The Cryptocurrency Exchange Trading Library (CCXT) is a cryptocurrency trading system with a unified API out of the box and optional normalized data. It supports many Bitcoin and altcoin exchange markets and merchant APIs [19].

Expert traders can create a trading strategy based on this data and access public transactions through the APIs [22]. The CCXT library is used to connect and trade with cryptocurrency exchanges and payment processing services worldwide. It provides quick access to market data for storage, analysis, visualization, indicator development, algorithmic trading, strategy backtesting, automated code generation and related software engineering.

As commented, it is designed for coders, skilled traders, data scientists and financial analysts to build trading algorithms. Hence, current CCXT features include [19]:

- Support for many cryptocurrency exchanges.
- Fully implemented public and private APIs.
- Optional normalized data for cross-exchange analysis and arbitrage.
- Out-of-the-box unified API, very easy to integrate.

## 7.4 Blackbird

Blackbird is a C++ trading system that automatically executes long and short arbitrage between Bitcoin exchanges. It can generate market-neutral strategies that do not transfer funds between exchanges [20].

The motivation behind Blackbird is to naturally profit from these temporary price differences between different exchanges while being market neutral. Unlike other

---

<sup>4</sup>An API key is used as a form of authentication to provide users with authorized access to the data it returns. The authentication access is provided in the form of a secret token. In simple words, an API key is akin to a password that lets the API confirm your identity.

Bitcoin arbitrage systems, Blackbird does not sell but actually short sells Bitcoin on the short exchange. This feature offers two important advantages. Firstly, the strategy is always market agnostic: fluctuations (rising or falling) in the Bitcoin market will not affect the strategy returns, which at turn eliminates the huge risks of this strategy. Secondly, this strategy does not require transferring funds between Bitcoin exchanges. Buy and sell transactions are conducted in parallel on two different exchanges, which involves in no need to deal with transmission delays [19].

## 7.5 StockSharp

StockSharp is an open-source trading platform for trading at any market of the world including 48 cryptocurrency exchanges [27]. It has a free C# library and free trading charting application. Manual or automatic trading (algorithmic trading robot, regular or HFT) can be run on this platform. StockSharp consists of five components that offer different features [19]:

- **S#.Designer:** Free universal algorithm strategy app, easy to create strategies;
- **S#.Data:** Free software that can automatically load and store market data;
- **S#.Terminal:** Free trading chart application (trading terminal);
- **S#.Shell:** Ready-made graphics framework that can be changed according to needs and has a fully open source in C#;
- **S#.API:** A free C# library for programmers using Visual Studio. Any trading strategies can be created in S#.API.

## 7.6 Freqtrade

Freqtrade is a free and open-source cryptocurrency trading robot system written in Python. It is designed to support all major exchanges and is controlled by Telegram. It contains backtesting, mapping and money management tools, and strategy optimization through Machine Learning [24]. Thus, Freqtrade has the following features [19]:

- **Persistence:** Persistence is achieved through SQLite technology.
- **Strategy optimization through Machine Learning:** Use Machine Learning to optimize your trading strategy parameters with real trading data.
- **Marginal Position Size:** Calculates winning rate, risk-return ratio, optimal stop-loss and adjusts position size, and then trades positions for each specific market.
- **Telegram management:** use telegram to manage the robot.
- **Dry run:** Run the robot without spending money;

## 7.7 CryptoSignal

CryptoSignal is a professional technical analysis cryptocurrency trading system [23]. Investors can track over 500 coins of thirteen different exchanges. Therefore, CryptoSignal presents an automated technical analysis which includes momentum, RSI, Ichimoku Cloud, MACD, among others; and also a system which gives alerts through email or Telegram. Additionally, it offers modular code for easy implementation of trading strategies, and it is easy to install with Docker<sup>5</sup> [19].

## 7.8 Ctubio

Ctubio is a C++ based low latency (high frequency) cryptocurrency trading system [26]. This trading system can place or cancel orders through supported cryptocurrency exchanges in less than a few milliseconds. Moreover, it provides a charting system that can visualize the trading account status including trades completed, or even target position for fat currency [19].

## 7.9 Catalyst

Catalyst is a cryptocurrency trading system which makes trading strategies easy to express and backtest them on historical data (daily and minute resolution), providing analysis and insights into the performance of specific strategies [21]. Also, it integrates statistics and Machine Learning libraries (such as *matplotlib*, *scipy*, *statsmodels* and *sklearn*) to support the development, analysis and visualization of the latest trading systems [19].

## 7.10 Using Machine Learning

Once the nine cryptocurrency trading bots have been presented, it is notorious to remark the import role of Machine Learning (ML) technology in these current systems, since it constructs computer algorithms that automatically improve themselves by finding patterns in existing data without explicit instructions [19].

Machine Learning is increasingly used in the prediction of cryptocurrency returns<sup>6</sup>, by means of algorithms which solve both classification and regression problems from a methodological point of view. Indeed, different kind of ML algorithms are present in the vast majority of trading bots.

---

<sup>5</sup>Docker is an open platform for developing, shipping, and running applications.

<sup>6</sup>Further information about Machine Learning in cryptocurrency is presented in Sec. 10.1 Machine Learning.

# Chapter 8

## Technical Analysis of the Financial Markets

It is prominent that financial markets can be analyzed from both the technical and the fundamental point of view. While technical analysis concentrates on the study of market action, fundamental analysis focuses on the economic forces of supply and demand that cause prices to move higher, lower, or stay the same. Hence, the fundamental approach examines all of the relevant factors affecting the price of a market in order to determine the intrinsic value of that market, whereas the important point for the technician is the effect.

Although it is essential having at least a passing awareness of the fundamentals, trading bots are usually based on technical analysis. For that reason, this section aims at stating constancy of the actual techniques and tools used in technical analysis, by means of *Technical Analysis of the Financial Markets* from John J. Murphy [46]. A reading of the original work by John J. Murphy is recommended for a more in-depth treatment.

### 8.1 Philosophy of Technical Analysis

There are three premises on which the technical approach is based:

1. **Market action discounts everything:** That means that anything that can possibly affect the price - fundamentally, politically, psychologically, or otherwise - is actually reflected in the price of the market.
2. **Prices move in trends:** The identification of trends in early stages of their development allows trading in the direction of those trends.
3. **History repeats itself:** Patterns that have worked well in the past are assumed that they will continue to work well in the future, since they are based on the study of human psychology, which tends not to change. Future is thus understood just as a repetition of the past.

## 8.2 Dow Theory

Charles Dow made his reputation as a financial expert late 1890's, when his writings formed the basis for the Dow Theory in market analysis. In fact, Dow Theory still forms the cornerstone of the study of technical analysis, even in the face of today's sophisticated computer technology, and the proliferation of newer and supposedly better technical indicators. Hereinafter, the six basic tenets of Dow Theory are described [46].

### 1. The Averages Discount Everything

This idea is based on the fact that the markets reflect every possible knowable factor that affects overall supply and demand. It is one of the basic premises of technical theory, and argues that while markets cannot anticipate events such as earthquakes, they quickly discount such occurrences, and almost instantaneously assimilate their affects into the price action.

### 2. The Market Has Three Trends

Before discussing how trends behave, it must be clarified what Dow considered a trend. He defined an uptrend as a situation in which each successive rally closes higher than the previous rally high, and each successive rally low also closes higher than the previous rally low. In other words, an uptrend has a pattern of rising peaks and troughs. The opposite situation, with successively lower peaks and troughs, defines a downtrend.

Therefore, Dow considered a trend to have three parts: *primary*, *secondary*, and *minor*, which he compared to the tide, waves and ripples of the sea. Specifically, the primary trend represents the tide, the secondary or intermediate trend represents the waves that make up the tide, and the minor trends behave like ripples on the waves.

An observer can thus determine the direction of the tide by noting the highest point on the beach reached by successive waves. If each successive wave reaches further inland than the preceding successive wave recedes, the tide has turned out and is ebbing. Unlike actual ocean tides, which last a matter of hours, Dow conceived of market tides as lasting for more than a year, and possible for several years.

The secondary, or intermediate, trend represents corrections in the primary trend and usually lasts three weeks to three months. These intermediate corrections generally retrace between one-third and two-thirds of the previous trend movement and most frequently about half, or 50%, of the previous move.

Lastly, the minor trend often lasts less than three weeks. This trend represents fluctuations in the intermediate trend.

Greater detail on trend concepts is presented in Sec. 8.3 Basic Concepts of Trends, obviously by using that same basic concepts and terminology.

### 3. Major Trends Have Three Phases

Dow focused his attention on primary or major trends, which he felt usually take place in three distinct phases: an accumulation phase, a public participation phase, and a distribution phase.

The accumulation phase represents informed buying by the most astute investors. If the previous trend was down, then at this point astute investors recognize that the market has assimilated all the so-called *bad news*. The public participation phase, in which most technical trend-followers begin to participate, occurs when prices begin to advance rapidly and business news improves. Finally, the distribution phase takes place when bullish stories increasingly appear; when economic news is better than ever; and when speculative volume and public participation increase. During this last phase, the same informed investors who began to "accumulate" near the bear market bottom, begin to "distribute" before anyone else starts selling.

### 4. The Averages Must Confirm Each Other

Dow meant that no important bull or bear market signal could take place unless both averages gave the same signal, thus confirming each other. He felt that both averages must exceed a previous secondary peak to confirm the inception or continuation of a bull market. He did not believe that the signals had to occur simultaneously, but recognized that a shorter length of time between the two signals provided stronger information. Hence, when the two averages diverged from one another, Dow assumed that the prior trend was still maintained.

### 5. Volume Must Confirm the Trend

Dow recognized volume as a secondary but important factor in confirming price signals. Simply stated, volume should expand or increase in the direction of the major trend. In a major uptrend, volume would then increase as prices move higher, and diminish as prices fall. In a downtrend, volume should increase as prices drop and diminish as they rally.

### 6. A Trend Is Assumed to Be in Effect Until It Gives Definite Signals That It Has Reversed

This tenet forms much of the foundation of modern trend-following approaches. A number of technical tools are available to traders to assist in the difficult task of spotting reversal signals, including the study of support and resistance levels, price patterns, trendlines, and moving averages. Some indicators can provide even earlier warning signals of loss of momentum. All of that notwithstanding, the odds generally favor that the existing trend will continue.

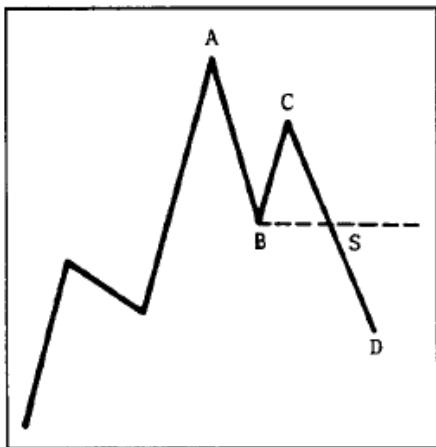


Figure 8.1: Failure Swing [46]

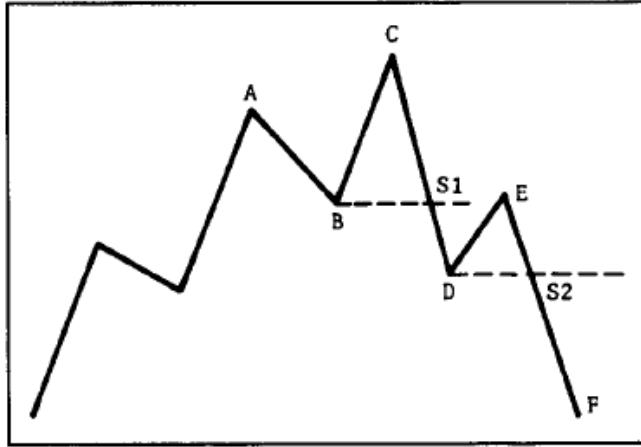


Figure 8.2: Nonfailure Swing [46]

In Fig. 8.1, notice that the rally at point C is lower than the previous peak at A. Price then declines below point B. The presence of these two lower peaks and two lower troughs gives a clear-cut sell signal at the point where the low at B is broken (point S). This reversal pattern is sometimes referred to as a *failure swing*.

At turn, in Fig. 8.2, the rally top at C is higher than the previous peak at A. Then price declines below point B. Some Dow theorists would see a sell signal at S1, while others would need to see a lower high at E before turning bearish at S2. This reversal pattern is referred to as a *nonfailure swing*, and is much more powerful pattern than the *failure swing*.

Fig. 8.3 and Fig. 8.4 show the same scenarios at a market bottom, respectively. In Fig. 8.3, it can be clearly observed that the buy signal takes place when point B is exceeded (at B1). At turn, Fig. 8.4 shows the buy signal occurrences at points B1 or B2.

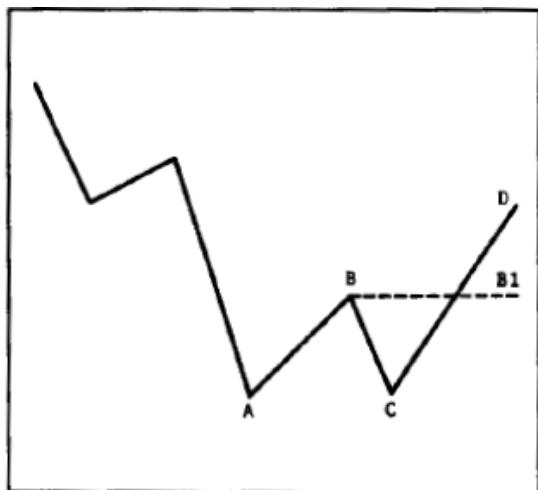


Figure 8.3: Failure Swing Bottom [46]

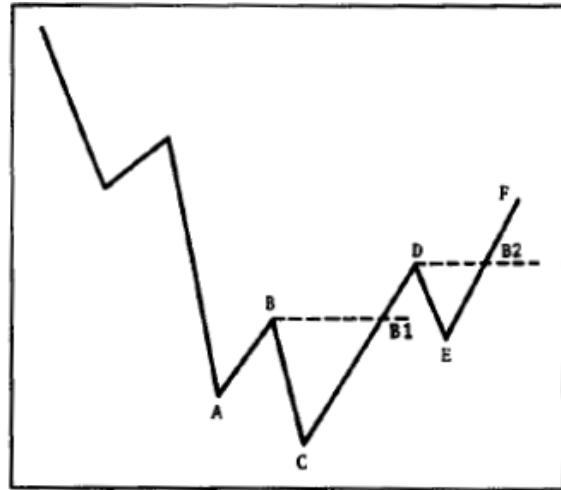


Figure 8.4: Nonfailure Swing Bottom [46]

## 8.3 Basic Concepts of Trends

The concept of *trend* is absolutely essential to the technical approach to market analysis. All of the tools used by the chartist - support and resistance levels, price patterns, moving averages, trendlines, etc. - have the sole purpose of helping to measure the trend of the market for the purpose of participating in that trend.

In a general sense, the trend is simply the direction of the market: which way it is moving. Even though, at that point it should be considered that markets do not generally move in a straight line in any direction, but are characterized by a series of *zigzags*, which resemble a series of successive waves with fairly obvious peaks and troughs. Therefore, it is the direction of those peaks and troughs that constitutes market trend.

### Trend Has Three Directions

With all of the above, an *uptrend* would be defined as a series of successively higher peaks and troughs, as shown in Fig. 8.5; a *downtrend* is just the opposite, a series of declining peaks and troughs (Fig. 8.6); and horizontal peaks and troughs would identify a sideways price trend, as evidenced in Fig. 8.7.

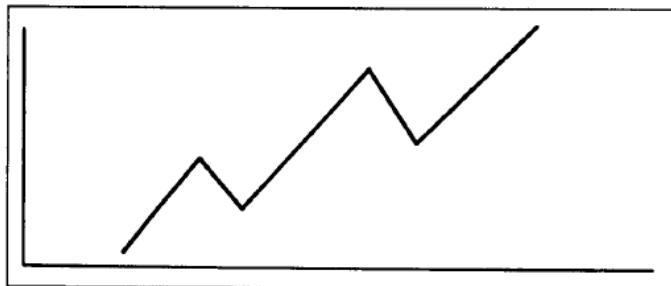


Figure 8.5: Example of an uptrend with ascending peaks and troughs [46]

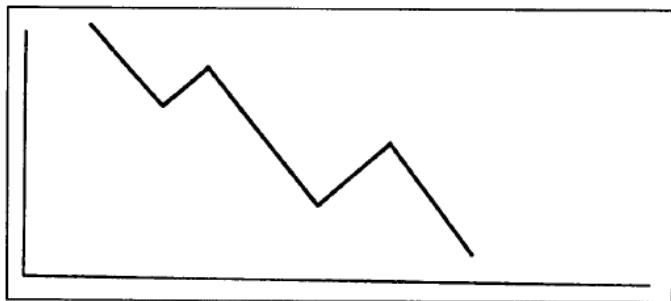


Figure 8.6: Example of a downtrend with descending peaks and troughs [46]

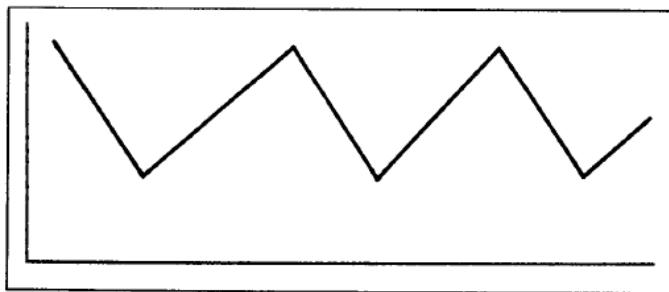


Figure 8.7: Example of a sideways trend with horizontal peaks and troughs [46]

It is absolutely relevant to be aware of the distinction of having a sideways trend (more commonly referred to as *trendless*), in which prices move in a flat, horizontal pattern that is referred to as a *trading range*. This type of sideways action reflects a period of equilibrium in the price level where the forces of supply and demand are in a state of relative balance.

### Trend Has Three Classifications

In addition to having three direction, trend is usually broken down into the three categories mentioned in Sec. 8.2 Dow Theory: major, intermediate, and near term trends. Hence, each trend becomes a portion of its next larger trend. For example, in Fig. 8.8, the major trend is up as reflected by the rising peaks and troughs (points 1, 2, 3, 4). The corrective phase (2-3) represents an intermediate correction within the major uptrend. But notice that the wave 2-3 also breaks down into three smaller waves (A, B, C). At point C, the analyst would say that the major trend was still up, but the intermediate and near terms trends was still up, but the intermediate near terms trends were down. At point 4, all three trends would be up.

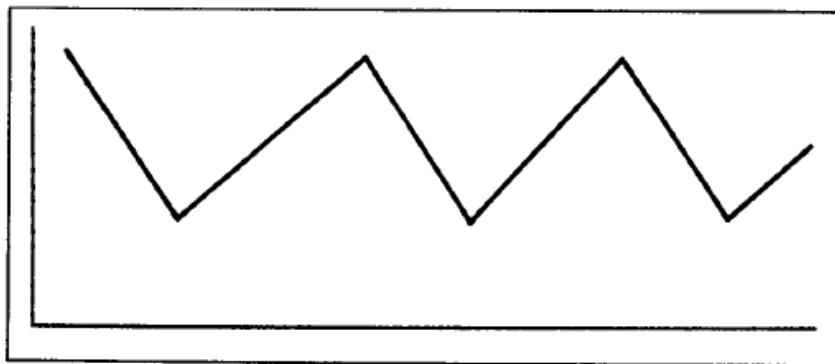


Figure 8.8: Example the three degrees of trend: major, secondary, and near term [46]

As a general statement, most trend-following approaches focus on the intermediate trend, which may last for several months. The near term trend is used primarily for timing purposes. In an intermediate uptrend, short term setbacks would be used to initiate long positions.

## Support and Resistance

It is evident that prices move in a series of peaks and troughs, and that the direction of those peaks and troughs determines the trend of the market. It is precisely from those peaks and troughs that the concepts of *support* and *resistance* arise.

The troughs, or reaction lows, are called *support*. The term is self-explanatory and indicates that support is a level or area on the chart under the market where buying interest is sufficiently strong to overcome selling pressure. As a result, a decline is halted and prices turn back up again. Usually a support level is identified beforehand by a previous reaction low. In Fig. 8.9, points 2 and 4 represent support levels in an uptrend.

On the other hand, *resistance* is the opposite of support and represents a price level or area over the market where selling pressure overcomes buying pressure and a price advance is turned back. Usually a resistance level is identified by a previous peak. In Fig. 8.9, points 1 and 3 are resistance levels.

For an uptrend to continue, each successive low (support level) must be higher than the one preceding it, apart from each rally high (resistance level) being higher than the one before it. In case of the corrective dip in an uptrend coming all the way down to the previous low, it may be an early warning that the uptrend is ending or at least moving from an uptrend to a sideways trend. If the support level is violated,

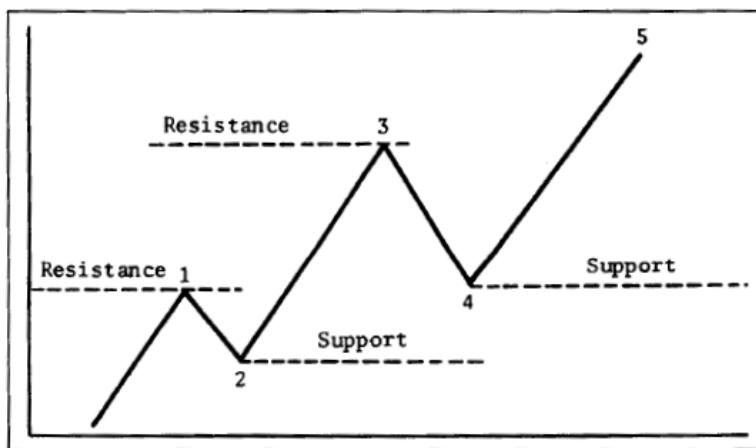


Figure 8.9: Support and resistance levels in uptrend [46]

then a trend reversal from up to down is likely. Absolutely the same concept is for a downtrend to continue, but of course the other way around.

With all of the above, it is important to note that support and resistance levels are defined by: the amount of time that the price has remained in that level; the volume; and the closeness in the time of the transaction, being the most recent activity the most powerful one.

### Trendlines

The basic trendline is one of the simplest of the technical tools employed by the chartist, but is also one of the most valuable. An *up trendline* is a straight line drawn upward to the right along successive reaction lows as shown by the solid line in Fig. 8.10. On the contrary, a *down trendline* is drawn downward to the right along successive rally peaks as shown in Fig. 8.11.

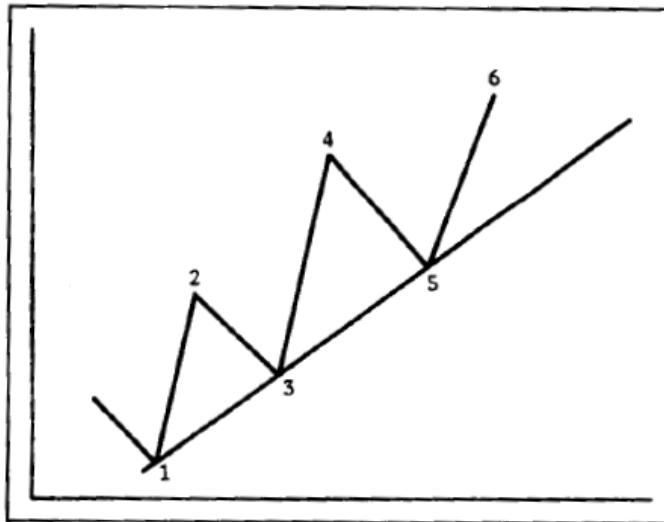


Figure 8.10: Example of an up trendline [46]

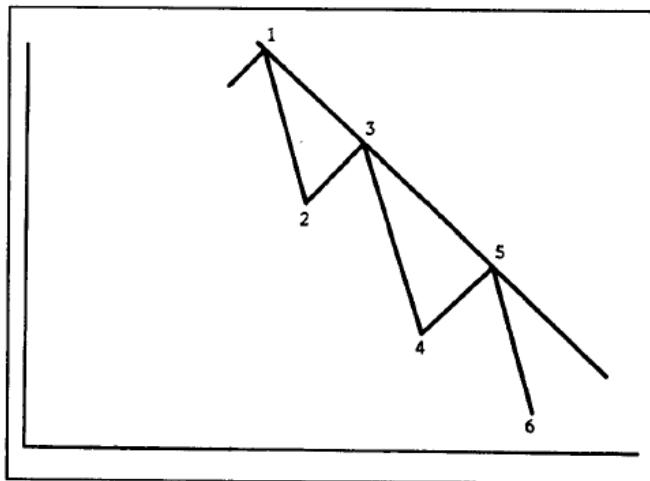


Figure 8.11: Example of a down trendline [46]

A trendline becomes very useful in a variety of ways. Once a trend assumes a certain slope or rate of speed, as identified by the trendline, it will usually maintain the same slope. The trendline then helps not only to determine the extremities of the corrective phase, but also to notice when that trend is changing.

In an uptrend, as in Fig. 8.12, the inevitable corrective dip will often touch or come very close to the up trendline. Because the intent of the trader is to buy dips in an uptrend, that trendline provides a support boundary under the market that can be used as a buying area. In the same way, a down trendline can be used as a resistance area for selling purposes, as can be seen in Fig. 8.13.

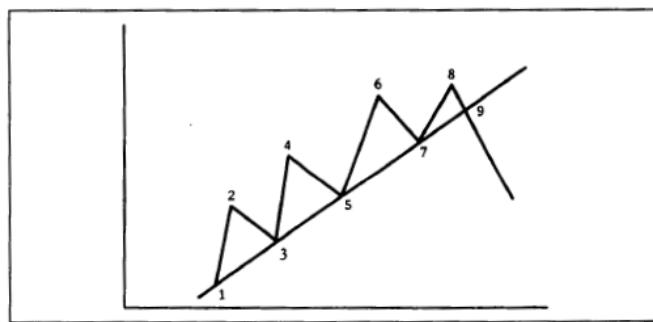


Figure 8.12: Breaking of an up trendline [46]

In both Fig. 8.12 and Fig. 8.13, the violation of the trendline at point 9 indicates a trend change, calling for liquidation of all positions in the direction of the previous trend. Very often, the breaking of the trendline is one of the best early warnings of a change in trend.

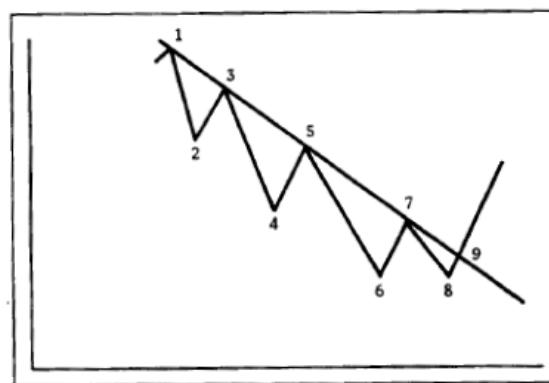


Figure 8.13: Breaking of a down trendline [46]

### Channel Line

The channel line is an useful variation of the trendline technique. Sometimes prices trend between two parallel lines - the basic trendline and the channel line. Obviously, when this is the case and when the analyst recognizes that a channel exists, this knowledge can be used to profitable advantage.

In an uptrend, as in Fig. 8.14, the basic up trendline can be used for the initiation of new long positions, while the channel line can be used for short term profit taking, or even to initiate a countertrend short position. Similarly, Fig. 8.15 represents the opposite situation.

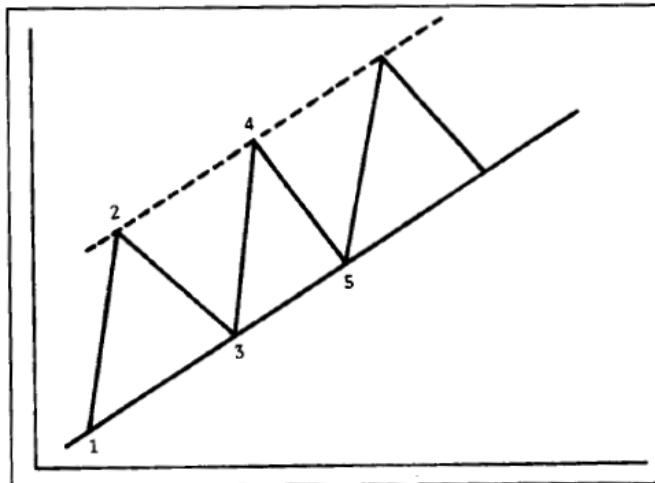


Figure 8.14: Example of an uptrend channel [46]

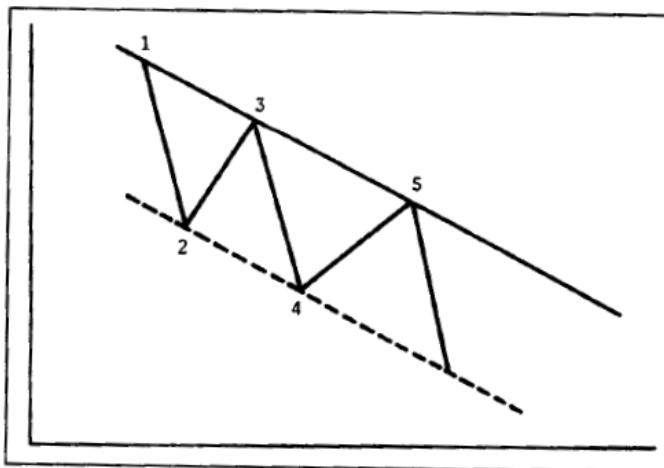


Figure 8.15: Example of a downtrend channel [46]

As in the case of the basic trendlines, the longer the channel remains intact and the more often it is successfully tested, the more important and reliable it becomes.

## 8.4 Volume

Most of the above discussion of charting theory has mainly concentrated on price action. Nevertheless, volume plays also an essential role in the forecasting process.

Volume is the number of entities traded during the time period under study. It is often plotted by a vertical bar at the bottom of the chart, just under the price action, as in Fig. 8.16.

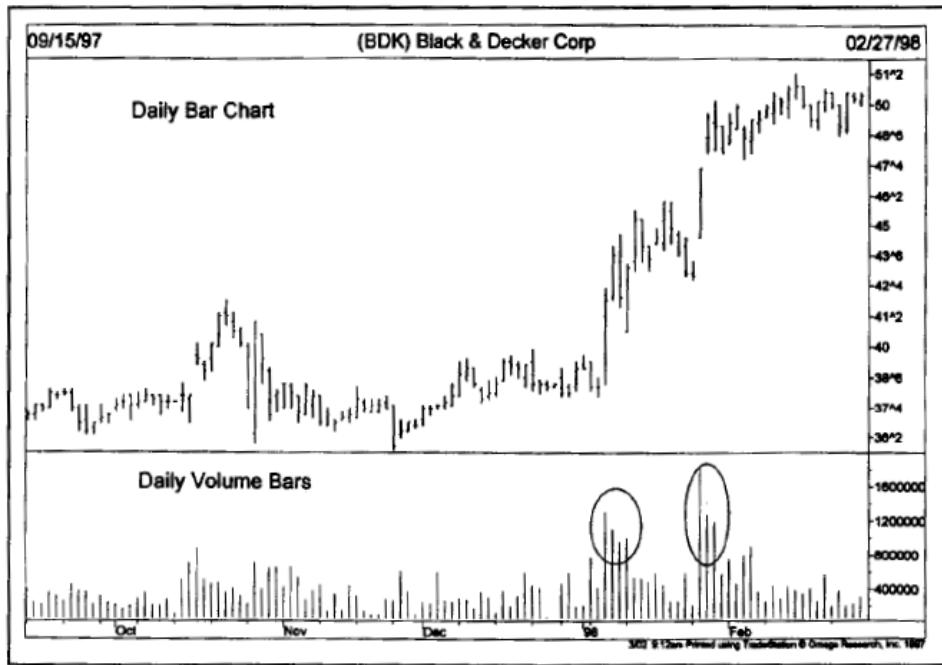


Figure 8.16: Example of volume bars [46]

In Fig. 8.16, notice that volume bars are noticeable larger as prices are rallying (see circles). That means that volume is confirming the price rise and is bullish.

As a general rule for interpreting volume, if it is increasing, then the current price trend will probably continue in its present direction (either up or down). If, however, volume is declining, the action can be viewed as a warning that the current price trend may be nearing an end. Hence, volume should increase or expand in the direction of the existing price trend. In an uptrend, volume should be heavier as the price moves higher, and should decrease or contract on price dips. As long as this pattern continues, volume is said to be confirming the price trend.

## 8.5 Chart Patterns

It would be a mistake to assume that most changes in trend are very abrupt affairs. the fact is that important changes in trend usually require a period of transition. The point is that these periods of transition do not always signal a trend reversal. Sometimes these sideways periods just indicate a pause or consolidation in the existing trend after which the original trend is resumed.

Hence, the study of these transition period and their forecasting implications leads to the question of price patterns, which are no more than formations that appear on price charts and can be classified into different categories, and that have predictive value.

There are two major categories of prices patterns: reversal and continuation. As these names imply, reversal patterns indicate that a considerable reversal in trend

is taking place, whereas the continuation patterns suggest that the market is only pausing for a while, possibly to correct a near term overbought or oversold condition, after which the existing trend will be resumed. The trick then is to distinguish between the two types of patterns as early as possible during the formation of the pattern itself.

For the reader to note, almost all the explanation below are justified in an uptrend, only not to repeat the same, but they are absolutely valid for downtrends too, obviously with the situation reversed.

### 8.5.1 Reversal Patterns

This section aims then at examining the three most commonly used major reversal patterns: the head and shoulders, triple tops and bottoms, and double tops and bottoms.

But before beginning that discussion of the individual major reversal patterns, there are a few preliminary points to be considered that are common to all of these reversal patterns:

1. A prerequisite for any reversal pattern is the existence of a prior trend.
2. The first signal of an impending trend reversal is often the breaking of an important trendline.
3. The larger the pattern, the greater the subsequent move.
4. Topping patterns are usually shorter in duration and more volatile than bottoms.
5. Bottoms usually have smaller price ranges and take longer to build.
6. Volume is usually more important on the upside.

#### 8.5.1.1 The Head and Shoulders

This is the best known and most reliable of all major reversal patterns. In fact, most of the other reversal patterns are just variations of the head and shoulders and will not require as extensive treatment.

Picture a situation in a major uptrend, where a series of ascending peaks and troughs gradually begin to lose momentum. The uptrend then levels off for a while. During this time, the forces of supply and demand are in relative balance. Once this distribution phase has been completed, support levels along the bottom of the horizontal trading range are broken and a new downtrend has been established. That new downtrend now has descending peaks and troughs. Fig. 8.17 is presented so as to better understand the pattern.

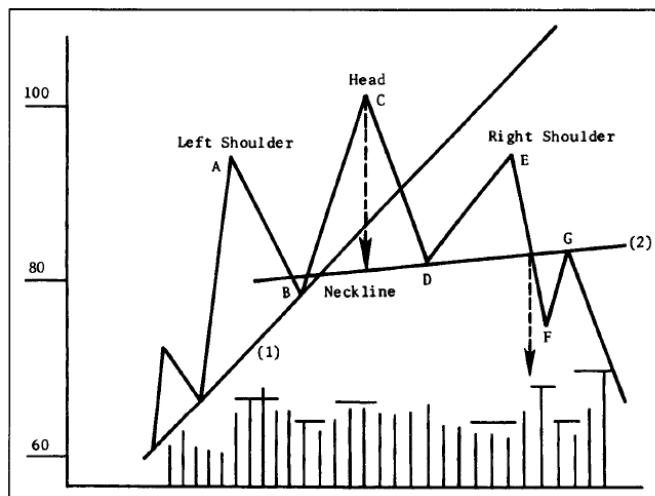


Figure 8.17: Example of a head and shoulders top [46]

At point A, the uptrend is proceeding as expected with no signs of a top. Volume expands on the price move into new highs, which is normal. The corrective dip to point B is on lighter volume, which is also to be expected. At point C, however, the alert chartist might notice that the volume on the upside breakout through point A is a bit lighter than on the previous rally. The change is not in itself of a major importance, but a little yellow caution light goes on in the back of the analyst's head.

Prices then begin to decline to point D and something even more disturbing happens. The decline carries below the top of the previous peak at point A. Remember that, in an uptrend, a penetrated peak should function as support on subsequent corrections. The decline well under point A, almost the the previous reaction low at point B, is another warning that something may be going wrong with the uptrend.

The market rallies again to point E, this time on even lighter volume, and is not able to reach the top of the previous peak at point C. To continue an uptrend, each high point must exceed the high point of the rally preceding it. The failure of the rally at point E to reach the previous peak at point C fulfills half of the requirement for a new downtrend - namely, descending peaks.

By this time, the major up trendline (line 1) has already been broken, usually at point D, constituting another danger signal. But, despite all of these warnings, the only known point is that the trend has shifted from up to sideways. This might be sufficient cause to liquidate long positions, but not necessarily enough to justify new short sales.

Also, the accompanying volume pattern plays an important role in the development of the head and shoulders top, as it does in all price patterns. As a general rule, the second peak (the head) should take place on lighter volume than the left shoulder. This is not a requirement, but a strong tendency and an early warning of diminishing buying pressure. The most relevant volume signal takes place during the third peak (the right shoulder), where volume should be noticeably lighter than on the previous

two peaks, and then expand on the breaking of the neckline, decline during the return move, and finally expand again once the return move is over.

As Fig. 8.18 shows, the inverse head and shoulders is pretty much a mirror image of the topping pattern. Now, there are three distinct bottoms with the head (middle trough) a bit lower than either of the two shoulders.

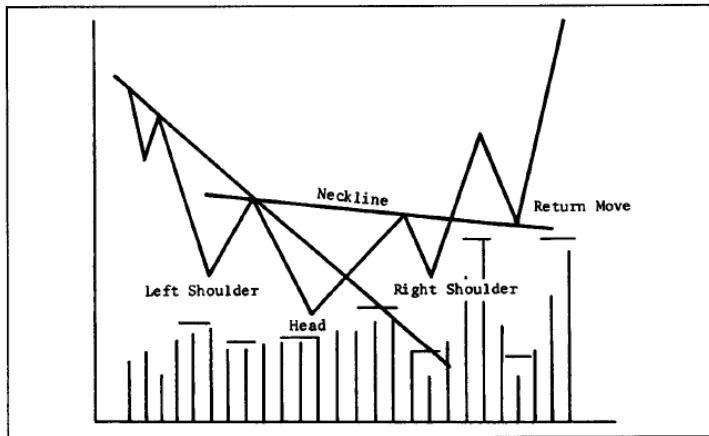


Figure 8.18: Example of an inverse head and shoulders [46]

### 8.5.1.2 Triple Tops and Bottoms

The triple top and the triple bottom, which are much rarer in occurrence, are just slight variations of their respective head and shoulder pattern. The main difference is that the three peaks or troughs are at about the same level, as can be observed in both Fig. 8.19 and Fig. 8.20.

As far as volume is concerned, it tends to decline with each successive peak at the top and should increase at the breakdown point. The triple top is not complete until support levels along both of the intervening lows have been broken. Conversely, prices must close through the two intervening peaks at the bottom to complete a triple bottom. Heavy upside volume on the completion of the bottom is also essential.

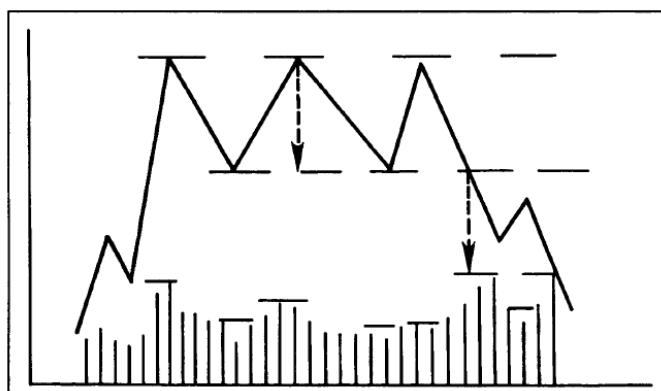


Figure 8.19: Example of a triple top [46]

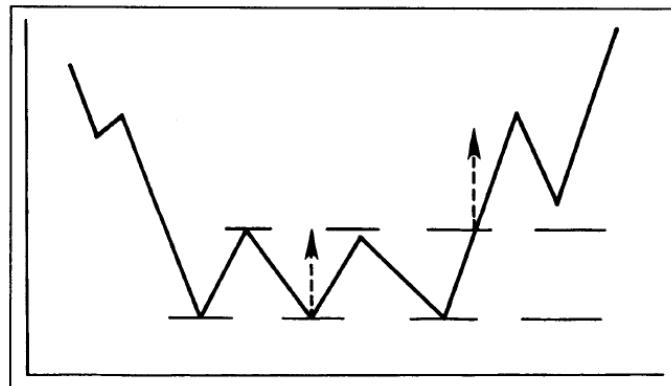


Figure 8.20: Example of a triple bottom [46]

### 8.5.1.3 Double Tops and Bottoms

A much more common reversal pattern is the double top or the double bottom. Next to the head and shoulders, it is the most frequently seen and the most easily recognized. Fig. 8.21 and Fig. 8.22 show both the top and bottom variety, respectively.

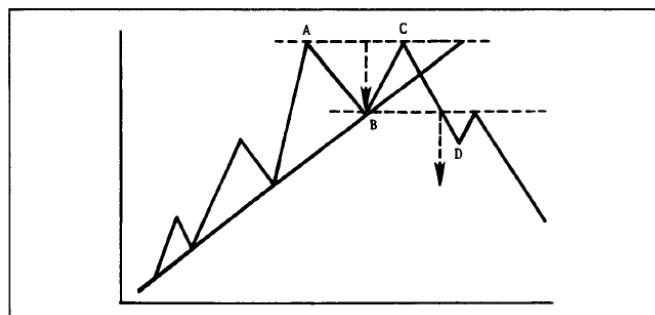


Figure 8.21: Example of a double top [46]

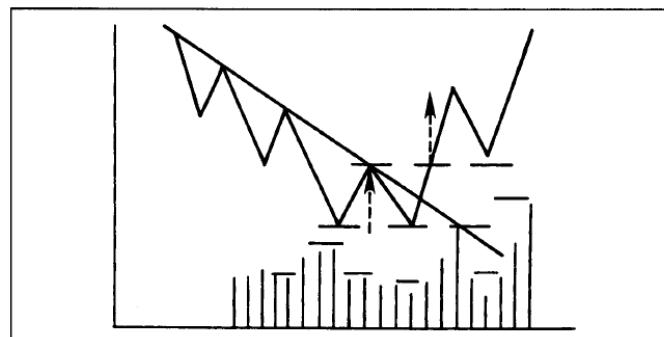


Figure 8.22: Example of a double bottom [46]

For obvious reasons, the top is often referred to as an *M* and the bottom as a *W*. The general characteristics of these double patterns are similar to that of their respective head and shoulders and triple patterns, except that only two peaks appear instead of three. The volume pattern is similar as is the measuring rule.

## 8.5.2 Continuation Patterns

These patterns often indicate that the sideways price action on the chart is nothing more than a pause in the prevailing trend, and that the next move will be in the same direction as the trend that preceded the formation. In comparison with reversal patterns, continuation patterns are usually shorter term in duration and more accurately classified as near term or intermediate patterns.

### 8.5.2.1 Triangles

One of the most common continuation patterns is the triangle. There are three types of triangles: symmetrical, ascending, and descending, respectively represented in Fig. 8.23, Fig. 8.24 and Fig. 8.25.

Although each type of triangle has a slightly different shape and has different forecasting implications, the minimum requirement for all triangles to be formed is four reversal points, two per each trendline.

#### Symmetrical Triangle

In Fig. 8.23, the triangle actually begins at point 1, which is where the consolidation in the uptrend begins. Prices pull back to point 2 and then rally to point 3. Point 3, however, is lower than point 1. The upper trendline can only be drawn once prices have declined from point 3.

Notice then that point 4 is higher than point 2. Only when prices have rallied from point 4, the lower upslanting line can be drawn. It is at this point that the formation of the symmetrical triangles arises, because of having four reversal points and consequently two converging trendlines.

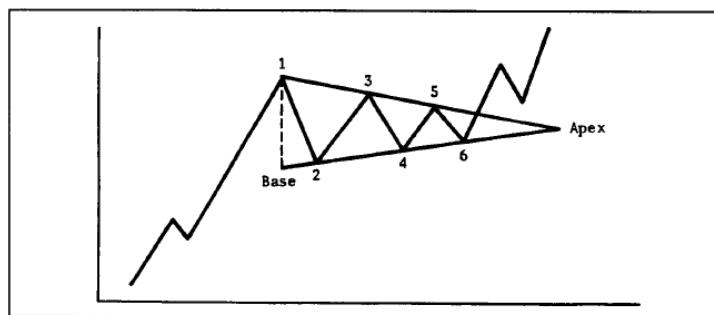


Figure 8.23: Example of a bullish symmetrical triangle [46]

For this pattern, there is a time limit for its resolution, and that is the point where the two lines meet - at the apex. As a general rule, prices should break out in the direction of the prior trend somewhere between two-thirds to three-quarters of the horizontal width of the triangle. That is, the distance from the vertical base on the left of the pattern to the apex at the far right. Because two lines must meet at some point, that time distance can be measured once the two converging lines are drawn. An upside breakout is signaled by a penetration of the upper trendline. If prices

remain within the triangle beyond the three-quarters point, the triangle begins to lose its potency, and usually means that prices will continue to drift out to the apex and beyond.

### Ascending Triangle

It is a variation of the symmetrical triangle. Notice that in Fig. 8.24, the upper trendline is flat, while the lower line is rising. This pattern indicates that buyers are more aggressive than sellers. It is considered a bullish pattern and is usually resolved with a breakout to upside.

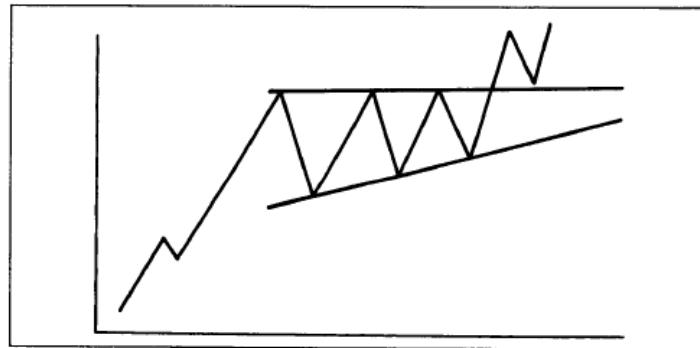


Figure 8.24: Example of an ascending triangle [46]

The measuring technique for the ascending triangle is relatively simple. Simply measure the height of the pattern at its widest point and project that vertical distance from the breakout point. This is just another of using the volatility of a price pattern to determine a minimum price objective.

### Descending Triangle

The descending triangle is just a mirror image of the ascending, and is generally considered a bearish pattern. Notice in Fig. 8.25 the descending upper line and the flat lower line. That indicates that sellers are more aggressive than buyers, and is usually resolved on the downside. The downside signal is registered by a decisive close under the lower trendline, usually on increased volume. A return move sometimes occurs which should encounter resistance at the lower trendline.

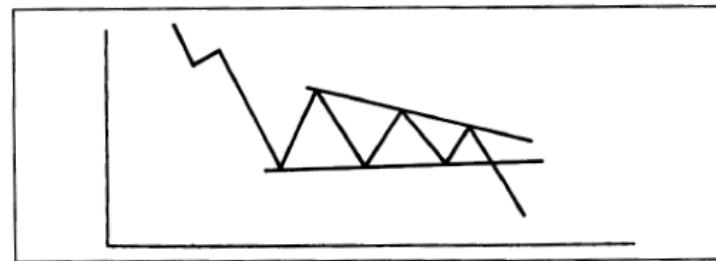


Figure 8.25: Example of a descending triangle [46]

The measuring technique is exactly the same as the ascending triangle, in the sense that the height of the pattern at the base to the left must be measured, and then that distance must be projected down from the breakdown point.

### 8.5.2.2 Broadening Formation

This is an usual variation of the triangle and is relatively rare. It is actually an inverted triangle. As pattern in Fig. 8.26 shows, the trendlines actually diverge in the broadening formation, creating a picture that looks like an expanding triangle.

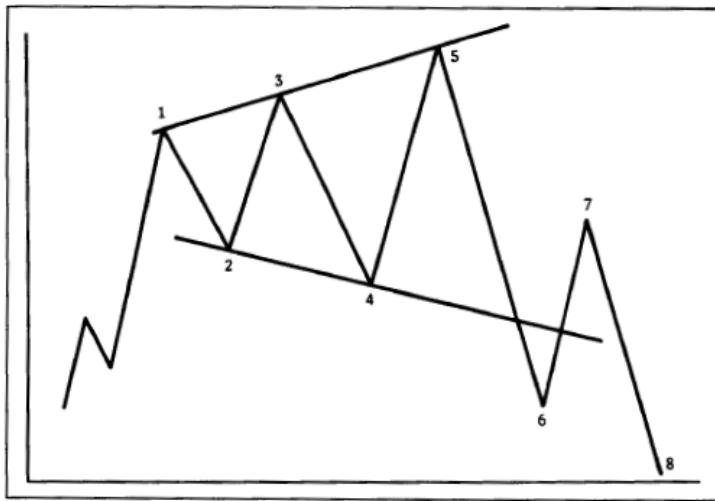


Figure 8.26: Example of a broadening top [46]

Here the volume pattern also differs, since the volume tends to expand along with the wider price swings, representing then a market that is out of control and usually emotional. That is associated with an unusual amount of public participation, which most occurs at major market tops. Consequently, this pattern is usually a bearish formation, and it generally appears near the end of a major bull market.

### 8.5.2.3 Flags and Pennants

The flag and pennant formations are quite common. They are usually treated together because of being very similar in appearance, since they both tend to show up at about the same place in an existing trend, and also have the same volume and measuring criteria.

The flag and pennant represent brief pauses in a dynamic market move. In fact, one of the requirements for both is that they be preceded by a sharp and almost straight line move. They represent situations where a steep advance or decline has gotten ahead of itself, and where the market pauses briefly to "catch its breath" before running off again in the same direction.

Flags and pennants are among the most reliable of continuation patterns and only rarely produce a trend reversal. Fig. 8.27 and Fig. 8.28 respectively show what these

two patterns look like. To begin with, notice the steep price advance preceding the formations on heavy volume. Notice also the dramatic drop off in activity as the consolidation patterns form and then the sudden burst of activity on the upside breakout.

The flag, as can be observed in Fig. 8.27, resembles a parallelogram or rectangle marked by two parallel trendlines that tend to slope against the prevailing trend. In a downtrend, the flag would have a slight upward slope.

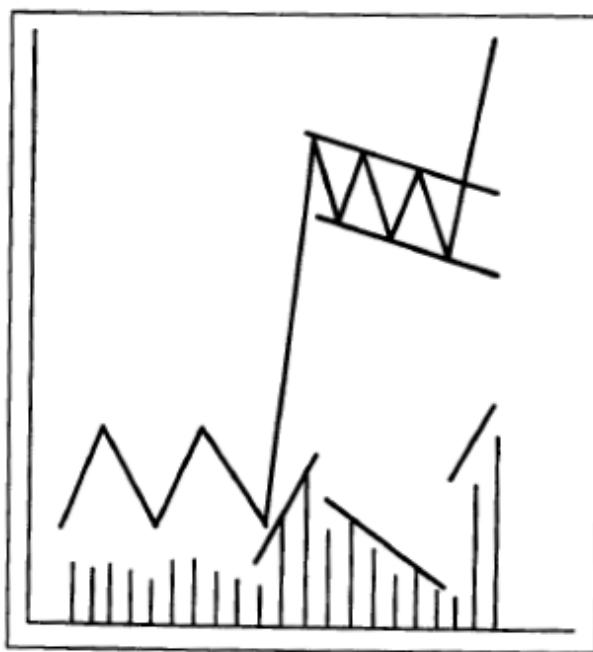


Figure 8.27: Example of a bullish flag [46]

At turn, the pennant is identified by two converging trendlines and is more horizontal, as can be seen in Fig. 8.28. It very closely resembles a small symmetrical triangle. An important requirement is that volume should dry up noticeably while each of the patterns is forming.

With that, the most relevant points of both patterns are:

1. They are both preceded by an almost straight line move on heavy volume.
2. Prices then pause for about one to three weeks on very light volume.
3. The trend resumes on a burst of trading activity.
4. Both patterns occur at about the midpoint of the market move.
5. The pennant resembles a small horizontal symmetrical triangle.
6. The flag resembles a small parallelogram that slopes against the prevailing trend.
7. Both patterns take less time to develop in downtrends.

8. Both patterns are very common in the financial markets.

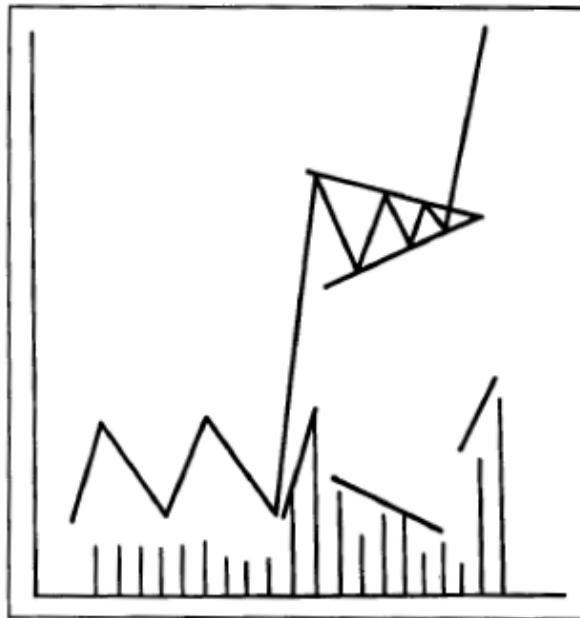


Figure 8.28: Example of a bullish pennant [46]

#### 8.5.2.4 Wedge Formation

The wedge formation is similar to a symmetrical triangle both in terms of its shape and the amount of time it takes to form. Like the symmetrical triangle, it is identified by two converging trendlines that come together at an apex. In terms of the amount of time it takes to form, the wedge usually lasts more than one month but not more than three months, putting it into the intermediate category.

What distinguishes the wedge is its noticeable slant. The wedge pattern has a noticeable slant either to the upside or the downside. As a rule, the wedge slants against the prevailing trend. Therefore, a falling wedge is considered bullish and a rising wedge is bearish, as Fig. 8.29 and Fig. 8.30 respectively show.

Notice that in Fig. 8.29 the bullish wedge slants downward between two converging trendlines, while in the downtrend in Fig. 8.30, the converging trendlines have an unmistakable upward slant.

Anyway, both wedges show up most often within the existing trend and usually constitute continuation patterns.

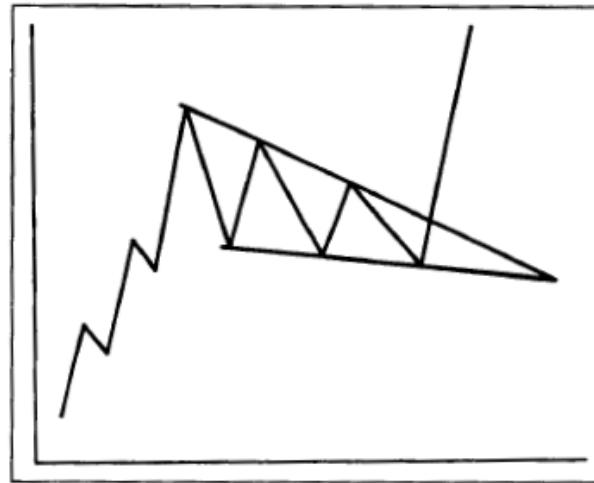


Figure 8.29: Example of a bullish falling wedge [46]

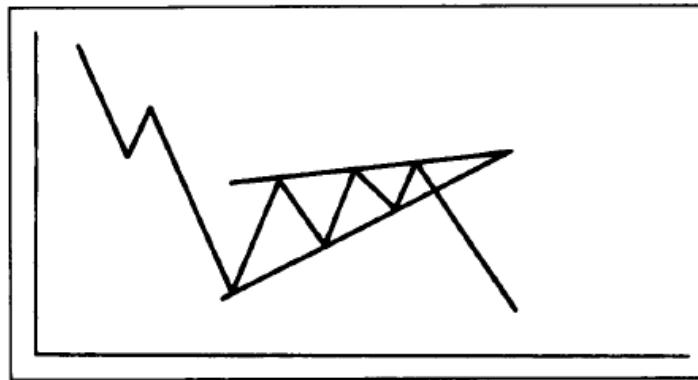


Figure 8.30: Example of a bearish wedge [46]

### 8.5.2.5 Rectangle Formation

The rectangle formation represents a pause in the trend during which prices move sideways between two parallel horizontal lines, as appreciated in Fig. 8.31 and Fig. 8.32.

The rectangle usually represents just a consolidation period in the existing trend, and is often resolved in the direction of the market trend that preceded its occurrence. In terms of forecasting value, it can be viewed as being similar to the symmetrical triangle but with flat instead of converging trendlines.

A decisive close outside either the upper or lower boundary signals completion of the rectangle and points the direction of the trend. The market analyst must always be on the alert, however, that the rectangular consolidation does not turn into a reversal pattern. In the Fig. 8.31, for example, notice that the three peaks might initially be viewed as a possible triple top reversal pattern.

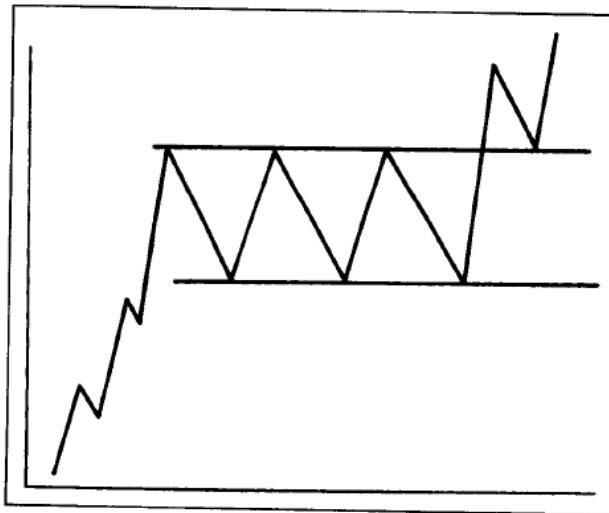


Figure 8.31: Example of a bullish falling rectangle [46]

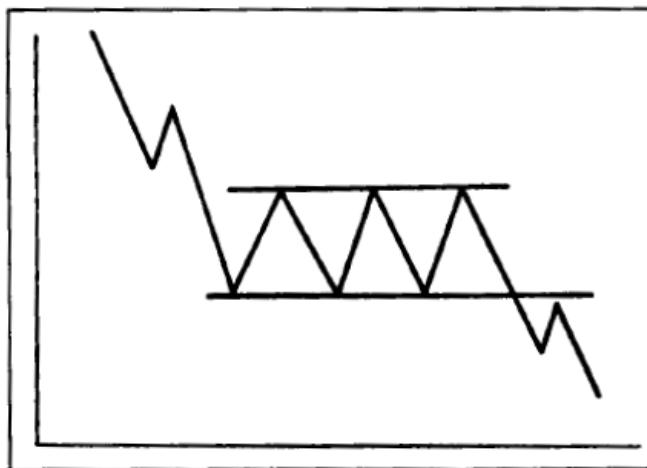


Figure 8.32: Example of a bearish rectangle [46]

### 8.5.3 Confirmation and Divergence

The principle of confirmation is one of the common themes running throughout the entire subject of market analysis, and it is used in conjunction with its counterpart - the divergence.

Confirmation refers to the comparison of all technical signals and indicators to ensure that most of those indicators are pointing in the same direction and are confirming one another.

Divergence is just the opposite of confirmation and refers to a situation where different technical indicators fail to confirm one another. While it is being used here in a negative sense, divergence is a valuable concept in market analysis, and one of the best early warning signals of impending trend reversals.

## 8.6 Chart Construction

In order to better comprehend the market data and properly analyze it, it is displayed in charts, such as line charts, bar charts and more recently, candlesticks, which are discussed in detail in Sec. 8.6.1 Candlestick Charting.

Fig. 8.33 shows a standard daily bar chart. It is called a bar chart because each day's range is represented by a vertical bar. Thus, the bar chart shows the open, high, low, and closing prices. The tic to the right of the vertical bar is the closing price. And the opening price is the tic to the left of the bar.

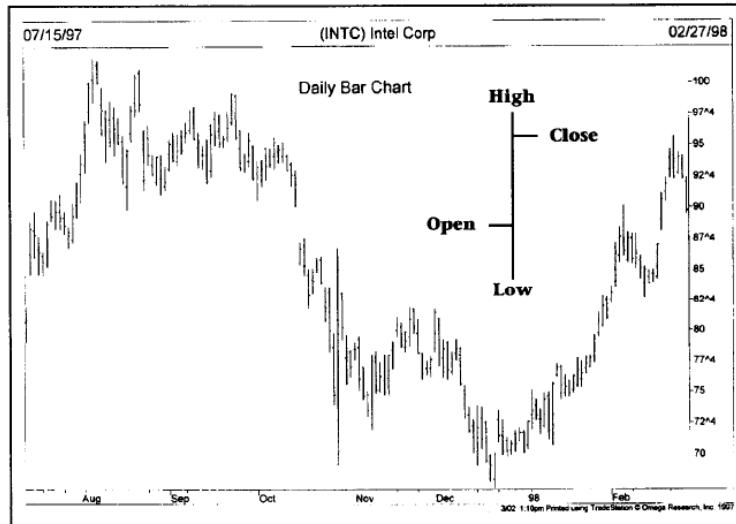


Figure 8.33: Example of a bar chart [46]

Fig. 8.34 shows what the same market looks like on a line chart, in which only the closing price is plotted for each successive day.

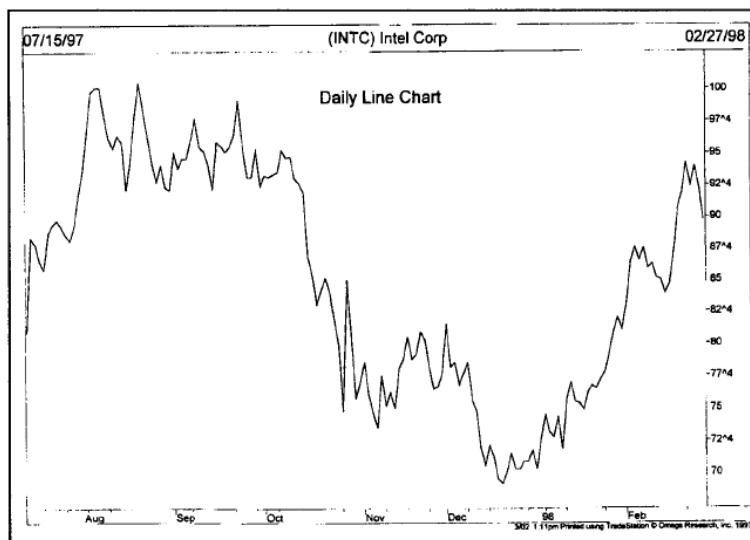


Figure 8.34: Example of line chart [46]

### 8.6.1 Candlestick Charting

Charting market data in candlestick form uses the same data available for standard bar charts: open, high, low, and close prices. While using the exact same data, candlestick charts offer a much more visually appealing chart. Information seems to jump off the page, and is more easily interpreted and analyzed. The box in Fig. 8.35 is a depiction of a single day of prices showing the difference between the bar (left) and the candlesticks.

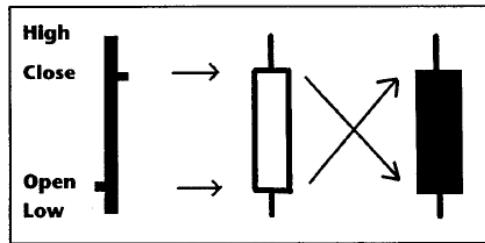


Figure 8.35: Example of candlestick [46]

The rectangle represents the difference between the open and close price for the day, and is called the body. Notice that the body can be either black or white. A white body means that the close price was greater (higher) than the open price, whereas the black body means that the close price was lower than the open price.

Therefore, the open and close prices are given much significance in candlesticks. The small lines above and below the body are referred to as *wicks* or *hairs* or *shadows*. They represent the high and low prices for the day and are normally not considered vital in the analysis.

Fig. 8.36 shows the same data in both the popular bar chart and the candlestick format. Although initially it takes some getting use to candlesticks, after a while their use ends up being better, since different shapes for candlesticks have different meanings.

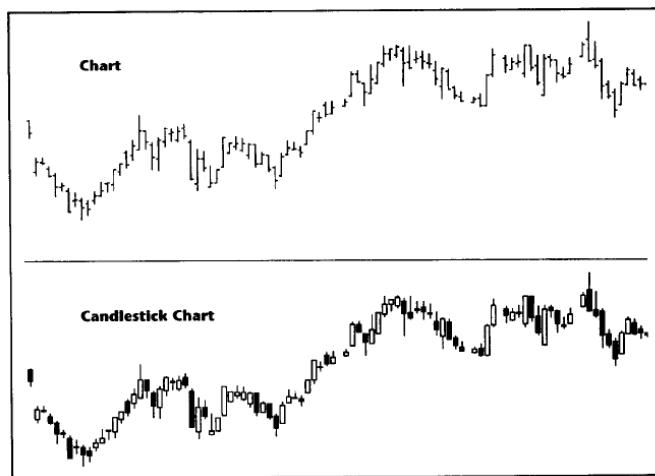


Figure 8.36: Bar chart versus candlestick chart [46]

Firstly, it is necessary to distinguish between long and short days. Long days are those in which the difference between the open and close prices is great, whereas in short days that difference is small. Fig. 8.37 visually represents that difference. Please note that the latter is only related to the size of the body and no reference is made to the high and/or low prices.

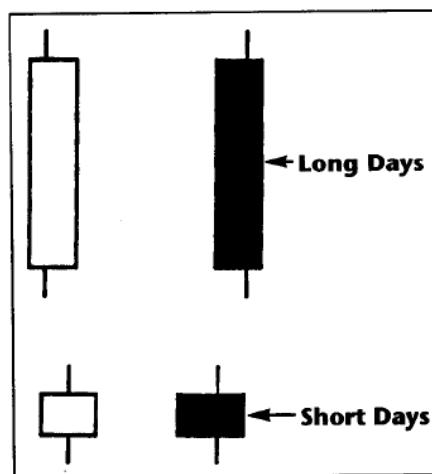


Figure 8.37: Candlesticks for long and short days [46]

From the identification of patterns with candlesticks, it is possible to forecast the trend of the market. Hence, candlestick patterns are psychological depictions of traders' mentality at the time. It vividly shows the actions of the traders as time unfolds in the market. The mere fact that humans react consistently during similar situations makes candle pattern analysis work.

A lot of candlestick patterns exist and are so useful for trading purposes. Nevertheless, they will not be collected here, due to the enormous amount of information that this would entail. The reader interested in candlestick pattern analysis is recommended to read *Chapter 12. Japanese Candlesticks* on [46].

### 8.6.2 Arithmetic versus Logarithmic Scale

Charts can be plotted using arithmetic or logarithmic price scales. For some types of analysis, particularly for very long range trend analysis, there may be some advantages to using logarithmic charts. Fig. 8.38 shows what the difference scales would look like.

On the arithmetic scale, the vertical price scale shown an equal distance for each price unit of change. Notice in this example that each point on the arithmetic scale is equidistant. On the log scale, however, note that the percentage increases get smaller as the price scale increases. The distance from points 1 to 2 is the same as the distance from points 5 to 10 because they both represent the same doubling in price.

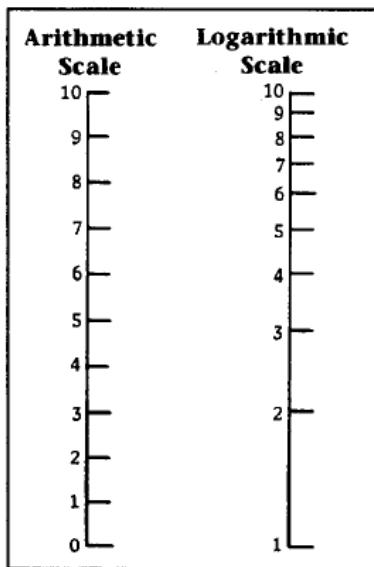


Figure 8.38: A comparison of an arithmetic and logarithmic scale [46]

### 8.6.3 Long Term Charts

The short term bar chart covers a relatively short period of time in the life of any market. A thorough trend analysis of a market, however, should include some consideration of how the daily market price is moving in relation to its long range trend structure. To accomplish that task, longer range charts must be employed, because they provide a perspective on the market trend that is impossible to achieve with the use of short term charts alone.

## 8.7 Moving Averages

The moving average is one of the most versatile and widely used of all technical indicators. Because of the way it is constructed and the fact that it can be so easily quantified and tested, it is the basis for many mechanical trend-following systems in use today. Moving average rules can easily be programmed into a computer, which then generates specific buy and sell signals. While two technicians may disagree as to whether a given price pattern is a triangle or a wedge, or whether the volume patterns favors the bull or bear side, moving average trend signals are precise and not open to debate.

A moving average is an average of a certain body of data. For example, if a 10 day average of closing prices is desired, the prices for the last 10 days are added up and the total is divided by 10. The term *moving* is used because only the latest 10 days' prices are used in the calculation. Therefore, the body of data to be averaged (the last 10 closing prices) moves forward with each new trading day. Fig. 8.39 shows a 10 day moving average applied to a daily bar chart.

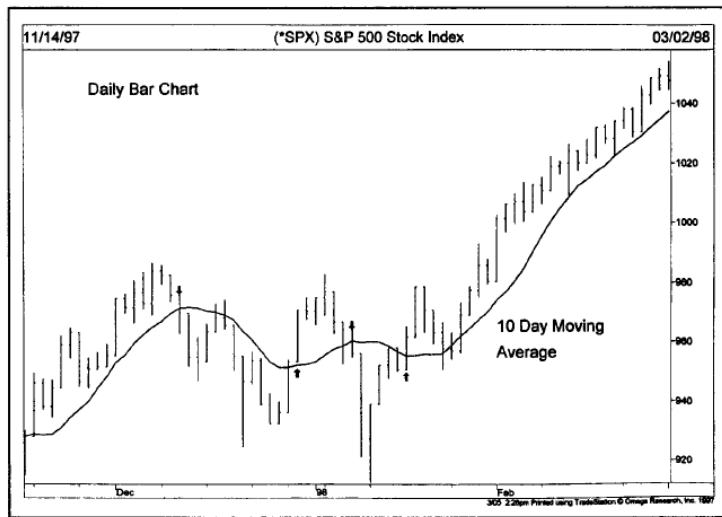


Figure 8.39: A 10 day moving average applied to a daily bar chart [46]

The above example deals with a simple 10 day moving average of closing prices. There are, however, other types of moving averages that are not that simple. Moreover, it is also relevant to decide how many days should be averaged, to decide either short or long term analysis.

Despite the fact that some analysts average a midpoint value of the price during the specific time span which is wanted to be analyzed, or even others make an average of the high, low and closing prices, the most common is to use the closing price for moving average analysis.

### 8.7.1 The Simple Moving Average

The simple moving average is the type used by most technical analysts. But there are some who question its usefulness on two points. The first criticism is that only the period covered by the average is taken into account. The second criticism is that the simple moving averages gives equal weight to each day's price. Hence, the last day receives the same weight as the first day in the calculation. Some analysts believe that a heavier weighting should be given to the more recent price action.

### 8.7.2 The Linearly Weighted Moving Average

In an attempt to correct the weighting problem, some analysts employ a linearly weighted moving average. In this calculation, a more recent closing price has always more significance than its previous one. For instance, a 10-day linearly weighted moving average follows:

$$\bar{x}_M = \frac{10x_i + 9x_{i-1} + 8x_{i-2} + \dots + 2x_{i-8} + x_{i-9}}{55} \quad (8.1)$$

where 55 is obtained by adding all the integers from 1 to 10, both included.

### 8.7.3 The Exponentially Smoothed Moving Average

This type of average addresses both of the problems associated with the simple moving average. First, the exponentially smoothed average assigns a greater weight to the more recent data. Therefore, it is a weighted moving average. But while it assigns lesser importance to past price data, it does include in its calculation all of the data in the life of the instrument. In addition, the user is able to adjust the weighting to give greater or lesser weight to the most recent day's price. This is done by assigning a percentage value to the last day's price, which is added to a percentage of the previous day's value. The sum of both percentage values adds up to 100. For example, the last day's price could be assigned a value of 10%, which is added to the previous day's value of 90%. That gives the last day 10% of the total weighting. That would be the equivalent of a 20 day average. By giving the last day's price a smaller value of 5%, lesser weight is given to the last day's data and the average is less sensitive. That would be the equivalent of a 40 day moving average. All of the above could be appreciated in Fig. 8.40.

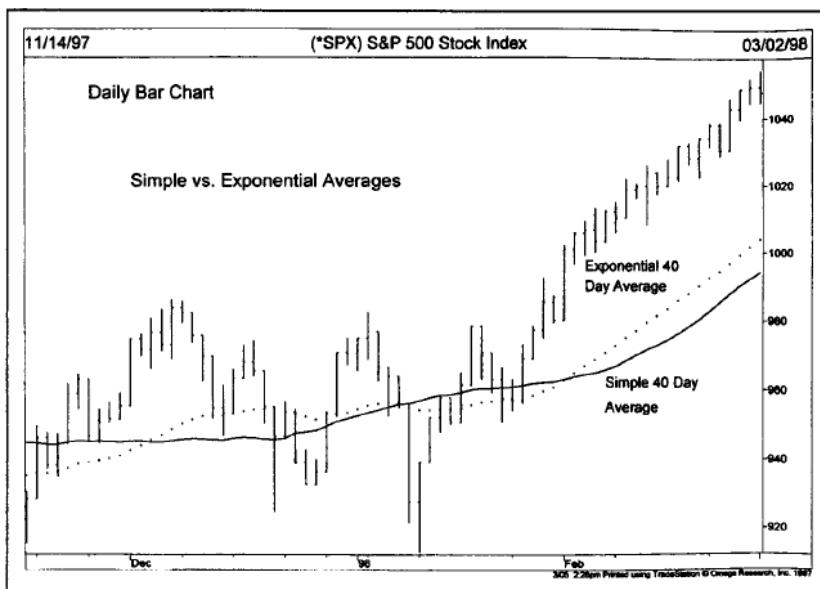


Figure 8.40: Exponential moving average is more sensitive than the simple one [46]

### 8.7.4 Using Two Moving Averages

Some traders use just one moving average to generate trend signals. The moving average is plotted on the bar chart in its appropriate trading day along with that day's price action. When the closing price moves above the moving average, a buy signal is generated. On the contrary, a sell signal is given when prices move below the moving average. This can be observed in Fig. 8.41.

Although it is quite straightforward to use only one moving average, it has some drawbacks. If a very short term average is employed (a 5 or 10 day), the average tracks prices very closely and several crossings occur. This action can be either good or bad. The use of a very sensitive average produces more trades (with higher

commission costs) and results in many false signals (whipsaws). If the average is too sensitive, some of the short term random price movement (or "noise") activates bad trend signals.

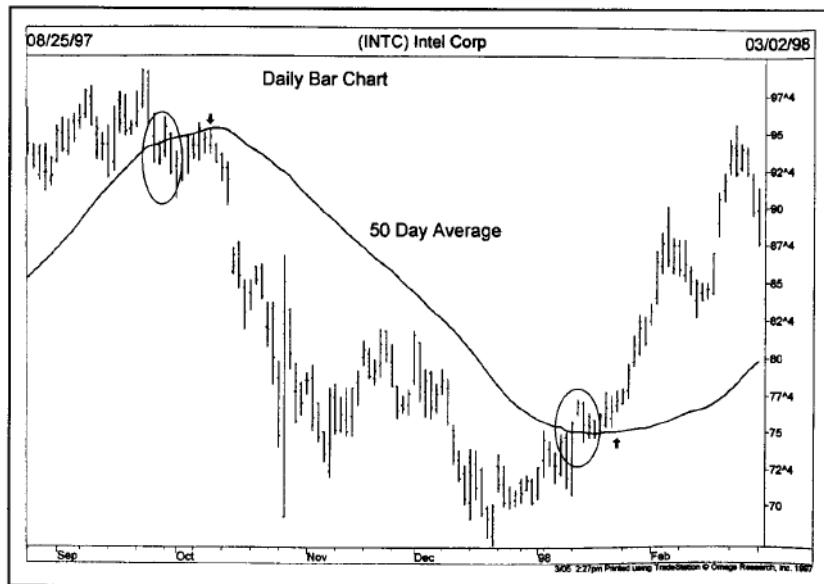


Figure 8.41: One moving average as trading signal [46]

While the shorter average generates more false signals, it has the advantage of giving trend signals earlier in the move. It stands to reason that the more sensitive the average, the earlier the signals will be. So there is a tradeoff at work here. The trick is to find the average that is sensitive enough to generate early signals, but insensitive enough to avoid most of the random noise.

At turn, a longer average performs better while the trend remains in motion, but it "gives back" a lot more when trend reverses. The very insensitivity of the longer average (the fact that it trailed the trend from a greater distance), which kept it from getting tangled up in short term corrections during the trend, works against the trader when the trend actually reverses. Therefore, the longer averages work better as long as the trend remains in force, but a shorter average is better when the trend is in the process of reversing.

It becomes clearer, thus, that the use of one moving average alone has several disadvantages. It is usually more advantageous to employ two moving averages.

The technique of using two average to generate signals is called the double crossover method. This means that a buy signal is produced when the shorter average crosses above the longer. Specifically, when the shorter average exceeds the longer average, a buy signal is generated, whereas the opposite case supposes a sell signal.

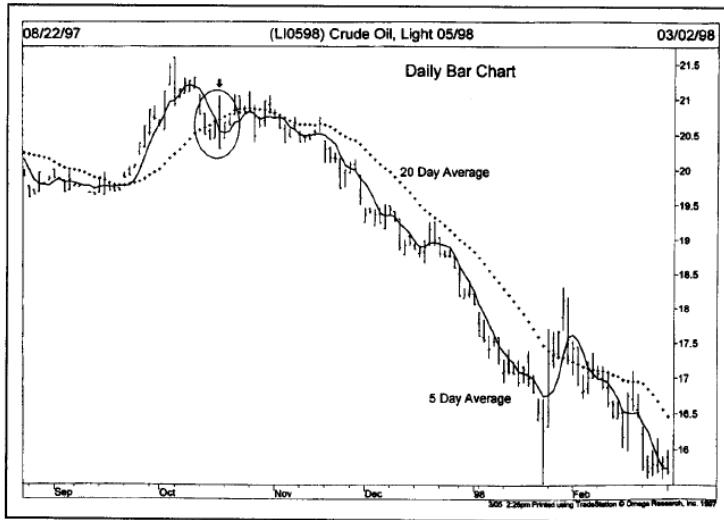


Figure 8.42: Example of the double crossover method [46]

## 8.8 Oscillators

Another alternative to trend-following approaches is the oscillator. It is extremely useful in non-trending markets where prices fluctuate in a horizontal price band, or trading range, creating a market situation where most trend-following systems simply do not work that well. The oscillator provides the technical traders with a tool that can enable them to profit from these periodic sideways and trendless market environments.

Nevertheless, the value of the oscillator is not limited to horizontal trading ranges. Used in conjunction with price charts during trending phases, the oscillator becomes an extremely valuable ally by alerting the trader to short term market extremes, commonly referred to as *overbought* or *oversold* conditions. The oscillator can also warn that a trend is losing momentum before that situation becomes evident in the price action itself. Oscillators can signal that a trend may be nearing completion by displaying certain divergences.

Hence, there are three situations when the oscillator is more useful.

1. The oscillator is most useful when its value reaches an extreme reading near the upper or lower end of its boundaries. The market is said to be overbought when it is near the upper extreme and oversold when it is near the lower extreme. This warns that the price trend is overextended and vulnerable.
2. A divergence between the oscillator and the price action when the oscillator is in an extreme position is usually an important warning.
3. The crossing of the zero (or midpoint) line can give important trading signals in the direction of the price trend.

### 8.8.1 Measuring Momentum

The concept of momentum is the most basic application of oscillator analysis. Momentum measures the velocity of price changes as opposed to the actual price levels themselves. Market momentum is measured by continually taking price differences for a fixed time interval. To construct a  $x$ -day momentum line, simply subtract the closing price  $N$  days ago from the last closing price. This positive or negative value is then plotted around a zero line. The formula for momentum is:

$$M = V - V^x \quad (8.2)$$

where  $V$  is the latest closing price and  $V^x$  is the closing price  $x$  days ago.

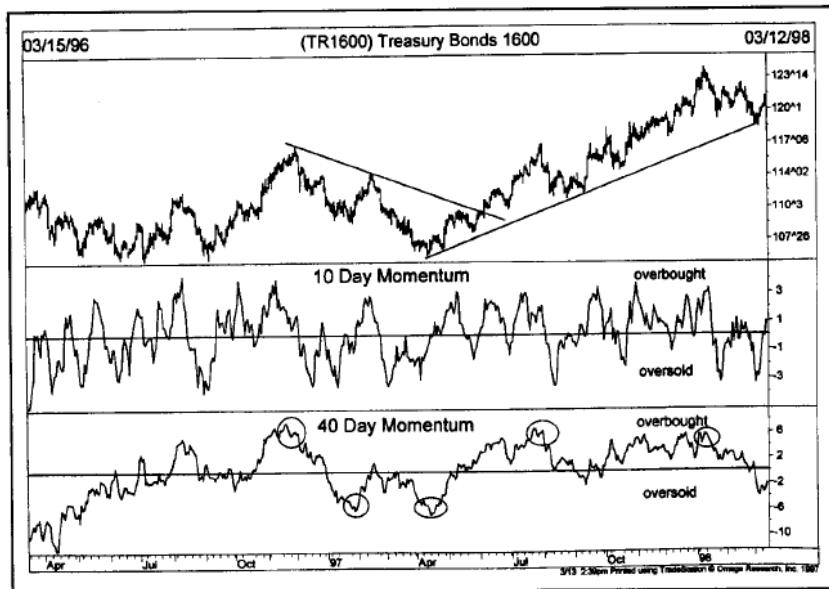


Figure 8.43: A comparison of 10 and 40 day momentum lines [46]

If the latest closing price is greater than that of  $x$  days ago (prices have moved higher), then a positive value would be plotted above the zero line. On the contrary, if the latest close is below the close  $x$  days earlier (prices have declined), then a negative value is plotted below the zero line, according to Equation (8.2).

In terms of momentum duration, the shorter the time period chosen, a more sensitive line with more pronounced oscillations is produced. Hence, a longer period of time results in a much smoother line in which the oscillator swings are less volatile. That can be observed in Fig. 8.43.

Because of the way it is constructed, the momentum line is always a step ahead of the price movement. It leads the advance or decline in prices, then levels off while the current price trend is still in effect. It then begins to move in the opposite direction as prices begin to level off. For instance, in Fig. 8.44 it can be observed that the value of the momentum indicator is that it turns sooner than the market itself, making it a leading indicator.

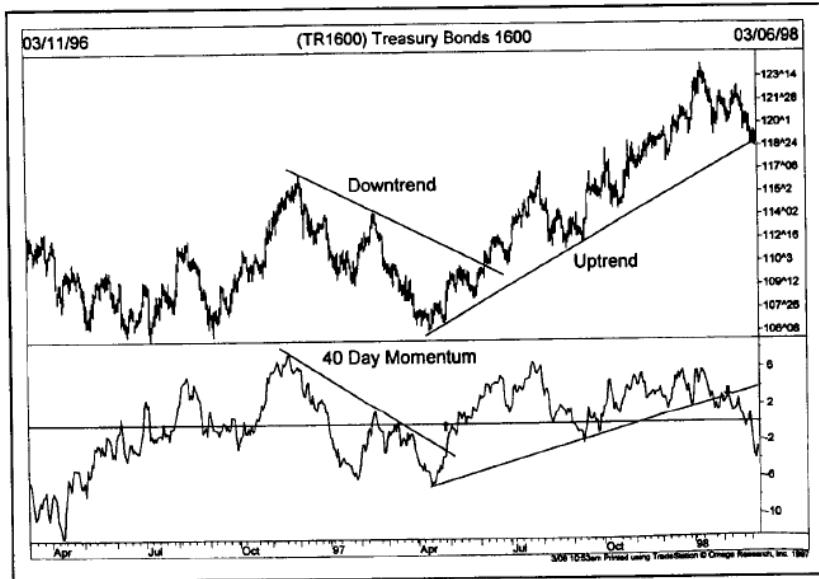


Figure 8.44: The trendlines on the momentum chart are broken sooner than those on the price chart [46]

As expected, the momentum chart has a zero line, whose crossing is generally used to generate buy and sell signals. A crossing above the zero line would be a buy signal, and a crossing below the zero line, a sell signal. Nevertheless, oscillator analysis should not be used as an excuse to trade against the prevailing market trend. For that reason, buy positions should only be taken on crossing above the zero line if the market trend is up. In the same way, short position should be taken on crossing below the zero line only if the price trend is down.

### 8.8.2 Oscillator Using Two Moving Averages

Sec. 8.7.4 Using Two Moving Averages discussed two moving averages being used to generate buy and sell signals. The crossing of the shorter average above or below the longer average registered buy and sell signals, respectively. The point is that these dual moving average combinations could also be used to construct oscillator charts. This can be done by plotting the difference between the two averages as a histogram, whose bars appear as a plus or minus value around a centered zero line. This type of oscillator has three uses:

1. To help spot divergences.
2. To help identify short term variations from the long term trend, when the shorter average moves too far above or below the longer average.
3. To pinpoint the crossing of the two moving averages, which occur when the oscillator crosses the zero line.

The shorter average is divided by the longer. In both cases, however, the shorter average oscillates around the longer average, which is in effect the zero line. As can be seen in Fig. 8.45, if the shorter average is above the longer, the oscillator would

be positive. A negative reading would be present if the shorter average were under the longer.

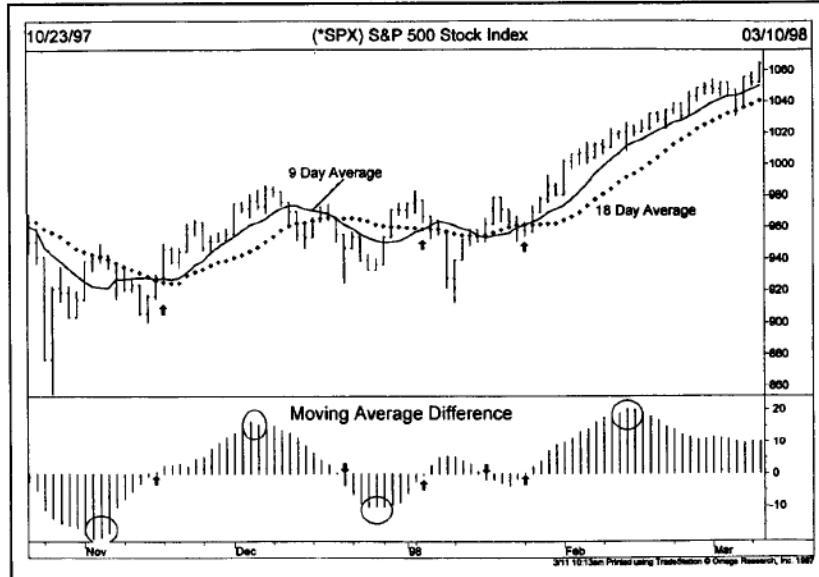


Figure 8.45: Example of a histogram using two moving averages [46]

### 8.8.3 The Relative Strength Index (RSI)

One of the two major problems in constructing a momentum line (using price differences) is the erratic movement often caused by sharp changes in the values being dropped off. A sharp advance or a decline some time ago can cause sudden shifts in the momentum line even if the current prices show little change. Some smoothing is therefore necessary to minimize these distortions. The second problem is that there is the need for a constant range for comparison purposes.

Hence, the RSI formula not only provides the necessary smoothing, but also solves the latter problem by creating a constant vertical range of 0 to 100. The actual formula is calculated as follows:

$$RSI = 100 - \frac{100}{1 + RS} \quad (8.3)$$

where:

$$RS = \frac{\text{Average of } x \text{ days' up closes}}{\text{Average of } x \text{ days' down closes}} \quad (8.4)$$

From Equation (8.3) can be deduced that the shorter the time period, the more sensitive the oscillator becomes and the wider its amplitude. RSI works best when its fluctuations reach the upper and lower extremes. Thus, if the user is trading on very short term basis and wants the oscillator swings to be more pronounced,

the time period can be shortened. The time period is then lengthened to make the oscillator smoother and narrower in amplitude.

RSI is plotted on a vertical scale of 0 to 100. As displayed in Fig. 8.46, movements above 70 are considered overbought, while an oversold condition would be a move under 30. Because of shifting that takes place in bull and bear markets, the 80 level usually becomes the overbought level in bull markets and the 20 level the oversold level in bear markets.

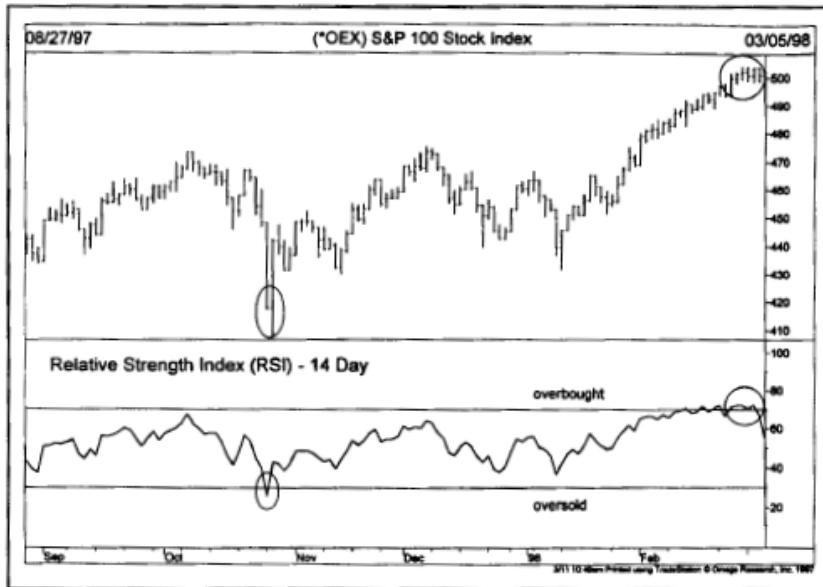


Figure 8.46: Example of RSI oscillator [46]

#### 8.8.4 The Importance of Trend

The importance of trading in the direction of the major trend cannot be overstated. The danger in placing too much importance on oscillators by themselves is the temptation to use divergence as an excuse to initiate trades contrary to the general trend. This action generally proves a costly and painful exercise. The oscillator, as useful as it is, is just one tool among many others and must always be used as an aid, not substitute, for basic trend analysis.

#### 8.8.5 When Oscillators Are Most Useful

There are times when oscillators are more useful than at other. Oscillators are really useful in sideways trading, since they track the price movement very closely, and the peaks and troughs on the price chart coincide almost exactly with the peaks and troughs on the oscillator.

Furthermore, oscillators should be ignored at the beginning of a new trend, but should be followed when the trend reaches maturity. The reason for this is that in the early stages of a new trend, following an important breakout, oscillators often reach extremes very quickly and stay there for awhile.

# Chapter 9

## Introduction to Cryptocurrency

All the fundamentals related to cryptocurrency required for developing the project are presented in this chapter. Notwithstanding that it is divided into two different sections so as to facilitate understanding and explanation, they are not independent of each other during the subsequent project.

### 9.1 Cryptocurrency

The proposal of this section is to define what a cryptocurrency is, and to outline the main concepts related to cryptocurrency, such as blockchain, cryptography, cryptocurrency wallet, and the difference between Proof-of-Work and Proof-of-Stake. Moreover, an idea of their history, the types of cryptocurrencies, and cryptocurrency exchange concept are also presented.

#### 9.1.1 Meaning and Definition

A cryptocurrency is any form of currency that exists digitally or virtually, and uses cryptography to protect transactions. Cryptocurrencies do not have a central regulatory or issuing authority; instead, they use a decentralized system to record transactions and issue new units.

Hence, a cryptocurrency is a digital payment system that does not rely on a third person to verify transactions. It is a peer-to-peer system that can allow anyone anywhere to send and receive payments. Instead of physical money that is transported and exchanged in the real world, cryptocurrency payments exist solely in the form of digital entries directed to an online database describing specific transactions.

Cryptocurrencies get their name because they use encryption to verify transactions. This means that advanced encryption is required to store and transmit cryptocurrency data between wallets and to public ledgers. Therefore, the purpose of encryption is to provide security, so that when one transfers cryptocurrency funds, the transactions are recorded in a public ledger, known as blockchain, and the cryptocurrencies themselves are stored in digital wallets.

### 9.1.2 Blockchain

As commented, cryptocurrencies operate on a distributed public ledger called a blockchain, a record of all transactions held and updated by coin owners. Therefore, a blockchain is a type of distributed ledger technology (DLT) that consists of growing list of records. Specifically, *block* is referred to a collection of data, and *chain* to a public database of these blocks, which are stored as a list.

By using a blockchain, participants can confirm transactions without a need for a central clearing authority. That is the main reason why they are used in cryptocurrency systems, as they allow to maintain a secure and decentralized record of transactions without the need for a trusted third party [36].

One key difference between a typical database and a blockchain is how the data is structured. A blockchain collects information together in groups, known as blocks, that hold sets of information. Blocks have certain storage capacities and, when filled, are closed and linked to the previously filled block, forming a chain of data known as the blockchain. All new information that follows that freshly added block is compiled into a newly formed block that will then also be added to the chain once filled.

Hence, this data structure inherently makes an irreversible timeline of data when implemented in a decentralized nature. When a block is filled, it is set in stone and becomes a part of this timeline. Each block in the chain is given an exact timestamp when it is added to the chain [36].

#### 9.1.2.1 Transaction Process

The goal of blockchain is to allow digital information to be recorded and distributed, but not edited. In this way, a blockchain is the foundation for immutable ledgers, or records of transactions that cannot be altered, deleted, or destroyed.

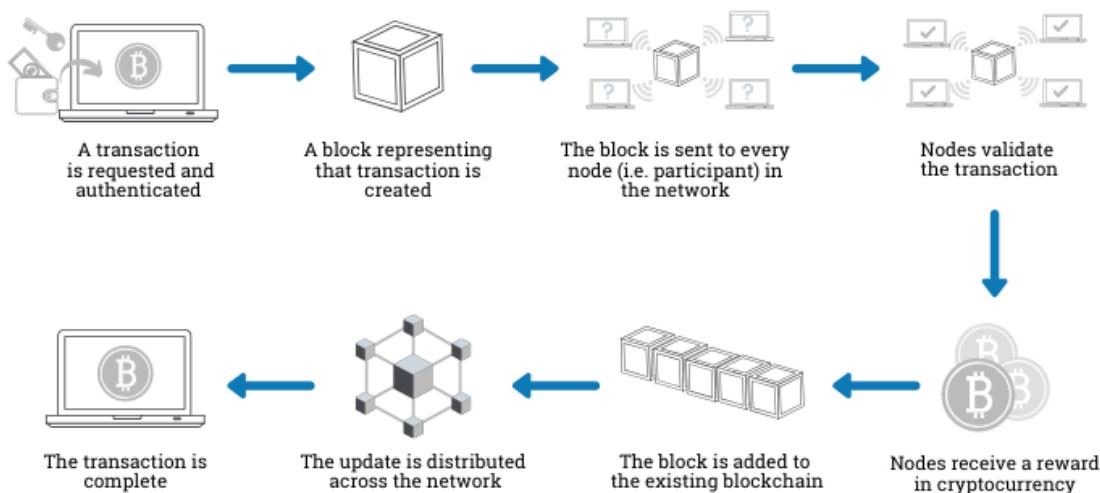


Figure 9.1: Transaction of cryptocurrency in blockchain [18]

No matter what industry the blockchain is used for, each blockchain transaction goes through the same steps shown in Fig. 9.1:

- **A transaction is requested and authenticated** from each party.
- **A block representing that transaction is created**, so that blocks are added to the blockchain; transactions are retrieved in the previous block, which is combined with the hash of the preceding block to obtain its hash, and then the derived hash is stored into the current block [19].
- **The block is sent to every node in the network**, meaning the data is replicated and stored instantaneously on each node across the system [5].
- **Nodes validate the transaction.** Confirmation is a critical concept in cryptocurrencies; they must be accepted and marked as legitimate. The two ways through which the validation is carried out are detailed in Sec. 9.1.5 Proof-of-Work against Proof-of-Stake.
- **Nodes receive a cryptocurrency reward**, as a way to send value for these transactions.
- **The block is added to the existing blockchain.** After the transaction is confirmed, each node must add it to its database. This process of transaction verification and recording is immediate and permanent [5]. Moreover, each block contains a code called a hash that is unique to that block. The block carries its own hash and the hash of the block before it so that users always know where the block should be located in the chain [54].
- **The update is distributed across the network.** When a transaction is recorded in the blockchain, details of the transaction such as price, asset, and ownership, are recorded, verified and settled within seconds across all nodes. A verified change registered on any one ledger is also simultaneously registered on all other copies of the ledger. Since each transaction is transparently and permanently recorded across all ledgers, open for anyone to see, there is no need for third-party verification [5].
- **The transaction is complete.** Blocks can contain many transactions. When it is completed, it is added to the chain, and the hash that it carries ensures that it is in proper chronological order [54].

In layman terms, the above process ensure that the transaction has become part of the blockchain. Hence, blockchain is a platform that drives cryptocurrency and is a technology that acts as a distributed ledger for the network, which creates a means of transaction and enables the transfer of value and information. Therefore, this work is undertaken to obtain cryptocurrency tokens, which can be thought of as tools on the blockchain, and in some cases can also function as resources or utilities. In other instances, they are used to digitise the value of assets. In summary, cryptocurrencies are part of an ecosystem based on blockchain technology [19].

### 9.1.3 Cryptography

Within the context of any application-to-application communication, there are some specific security requirements, including [42]:

- **Authentication:** The process of proving one's identity.
- **Privacy:** Ensuring that no one can read the message except the intended receiver.
- **Integrity:** Assuring the receiver that the received message has not been altered in any way from the original.
- **Non-repudiation:** A mechanism to prove that the sender really sent this message.

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. In simplest terms, cryptography is to make communications secret, which here means that, even in the presence of an eavesdropper who can monitor all communications, the intended message can still be delivered to the receiver while kept secret to others [15], often by means of mathematical techniques to achieve some desirable properties for secrecy.

As can be seen in Fig. 9.2, cryptography is often associated with the process of an ordinary plaintext being converted to ciphertext. Ciphertext is text that only the intended receiver can decode. The process that converts plaintext to ciphertext is known as encryption, while the process of conversion from ciphertext to plaintext is called decryption [15].

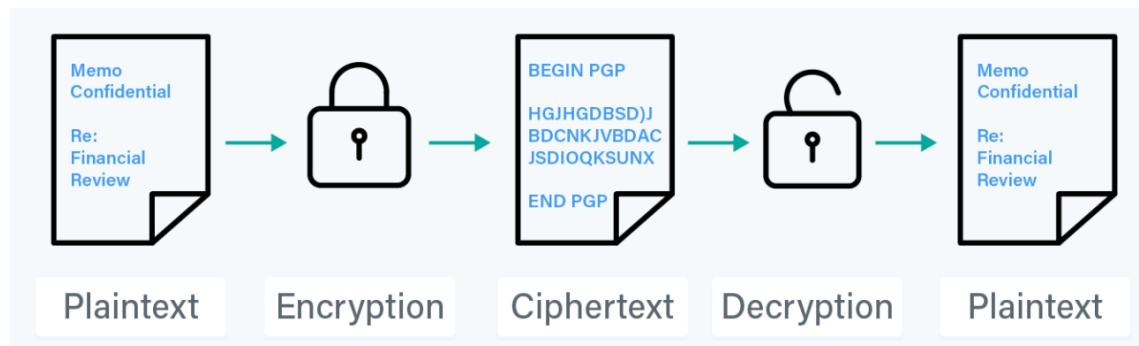


Figure 9.2: Cryptography jargon [15]

Blockchain and cryptocurrencies use cryptography in multiple ways: for wallets, transactions, security, and privacy-preserving protocols, by means of different subsequent algorithms.

#### 9.1.3.1 Cryptographic Algorithms

Based on the number of keys that are employed for encryption and decryption, cryptographic algorithms can be classified [42]:

- **Secret Key Cryptography (SKC) - Symmetric Encryption:** Uses one key for encryption and another for decryption.
- **Public Key Cryptography (PKC) - Asymmetric Encryption:** Uses a single key for both encryption and decryption.
- **Hash Functions:** Uses a mathematical transformation to irreversibly "encrypt" information.

At that point, it is remarkable to note that each of these algorithms have specific applications that are irreplaceable to each other, and for that reason they should be differentiated.

### Symmetric Encryption

Symmetric encryption uses a single key for both encryption and decryption, as shown in Fig. 9.3. The message sender uses the key to encrypt the plaintext and sends the ciphertext to the receiver. The receiver applies the same key to decrypt the message and recover the plaintext.

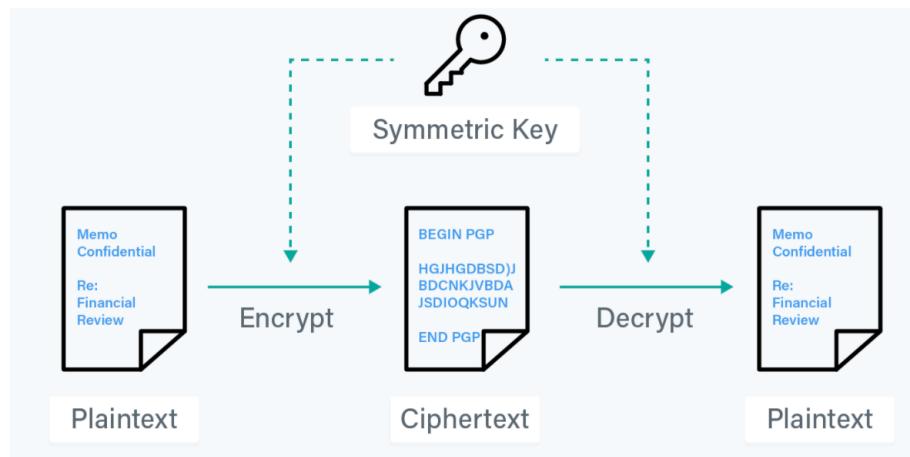


Figure 9.3: Symmetric encryption [15]

Symmetric key systems are faster and simpler (when compared to asymmetric key systems). However, they do not solve the key exchange problem between sender and receiver if the key is not known in advance [15].

With this form of cryptography, it is therefore obvious that the key must be known to both the sender and the receiver; that, in fact, is the secret. The biggest difficulty with this approach, of course, is the distribution of the key.

Secret key cryptography schemes are generally categorized as being either *stream ciphers* or *block ciphers*. Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism so that the key is constantly changing. A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block.

In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher.

With that, it is clearly that symmetric encryption is ideally suited to encrypting messages, since the sender can generate a session key on a per-message basis to encrypt the message; and the receiver, of course, needs the same session key to decrypt the message [42].

### Asymmetric Encryption

Asymmetric encryption uses two different keys for encryption and decryption. The key that will need to be kept secret is called the private key, while the key that does not is called the public key.

For instance, if A wants to send a message to B and ensure that B will be the only person able to comprehend the message, A can encrypt the message using the public key such that only B can decrypt the message using the private key [15].

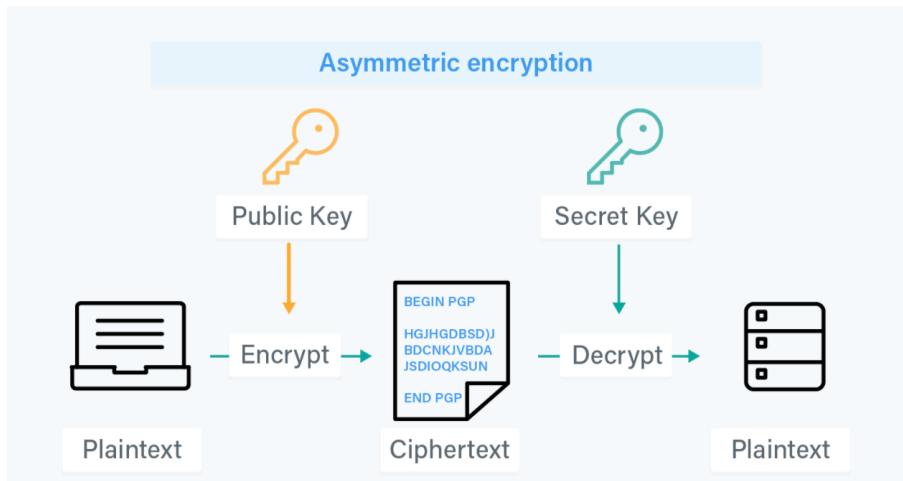


Figure 9.4: Asymmetric encryption [15]

While the public key and private key are different, they are mathematically related. But the mathematical relationship is only usable upon encryption and decryption, and others can never derive the private key even if they know the public key [15]. Specifically, it depends upon the existence of so-called one-way functions, or mathematical functions that are easy to compute whereas their inverse function is relatively difficult to compute [42].

Asymmetric encryption employs two keys that are mathematically related although knowledge of one key does not allow someone to easily determine the other key. One key is used to encrypt the plaintext (public key) and the other key is used to decrypt the ciphertext (private key), as can be seen in Fig. 9.4. The important point here is that it does not matter which key is applied first, but that both keys are required for the process to work [42].

Asymmetric cryptography algorithms are widely used in cryptocurrencies. For example, the wallet address is a public key, and only those who have the private key are able to use the money inside [15].

## Hash Functions

Hash functions, also called message digests and one-way encryption, are algorithms that, in some sense, use no key (Fig. 9.5). Instead, a fixed-length hash value is computed based upon the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered [42].

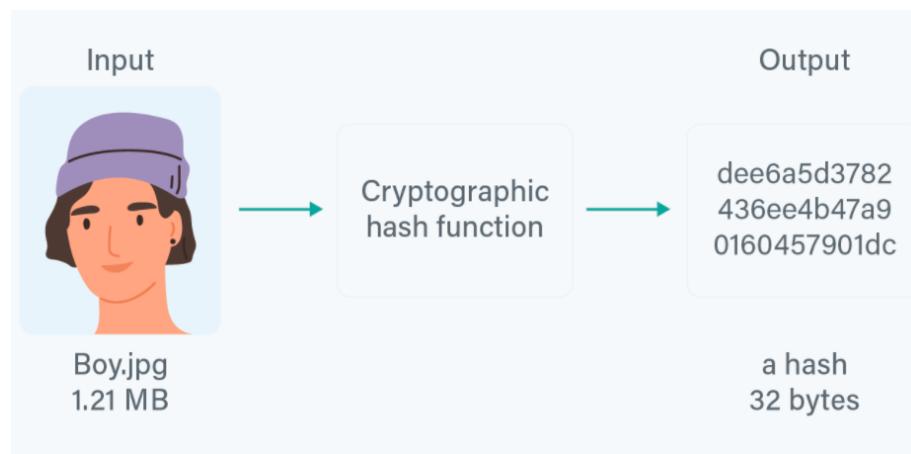


Figure 9.5: Hash functions [15]

Hash algorithms are typically used to provide a digital fingerprint of the contents of a file, often used to ensure that the file has not been altered by an intruder or virus. Hash functions are also commonly employed by many operating systems to encrypt passwords. Hash functions, then, provide a measure of the integrity of a file [42].

Bearing the above in mind, it is essential to take into account that a hash function should fulfill three security properties [15]:

- **Collision Resistant:** Nobody can find two different input strings so that if you apply the hash function to them, they have the same output. Mathematically, it can be expressed as:  $x$  and  $y$  do not exist, where  $H(x) = H(y)$  and  $x$  are not equal to  $y$ .
- **Hidden:** Given  $H(x)$ , it is infeasible to find  $x$ . This can be explained as if you are given the hashed version of  $x$ , you will not find  $x$ .
- **Puzzle-Friendly:** If someone wants to target a certain hash function, get some value of  $y$ . If part of the input is chosen randomly, it is difficult to find another value to target the hash function value. Given an output  $y$  of the hash function, if  $k$  is chosen from a random distribution, there is no way to find an  $x$ , such that the  $k|x$  hash to  $y$ :  $H(k|x)=y$ .

Hash functions are well-suited for ensuring data integrity because any change made to the contents of a message will result in the receiver calculating a different hash value than the one placed in the transmission by the sender. Since it is highly unlikely that two different messages will yield the same hash value, data integrity is ensured to a high degree of confidence [42].

#### 9.1.4 Cryptocurrency Wallet

Once cryptography has been understood, it is time to comprehend how one can make transactions in a specific blockchain: by means of a cryptocurrency wallet.

Cryptocurrency wallets (crypto-wallets) are key to use blockchain. Every user, who intend to use blockchain platform for any transaction, has to use crypto-wallet. Unlike traditional pocket wallets, cryptocurrencies are not stored in crypto-wallets. In fact, cryptocurrencies are neither stored in any single area nor exist anywhere in any bodily form, but it exists as data of transactions stored on blockchain [52].

Therefore, wallets are software applications which can be used to view cryptocurrency balances and make transactions. Each wallet type is a piece of different software, which provides access to blockchain data and operate on blockchain data. In general, any given wallet can work with one or more cryptocurrencies and features.

Public addresses are like cryptocurrency precise account numbers, they are used to acquire a particular kind of cryptocurrency and can be shared publicly. For example, to access Bitcoin blockchain one needs Bitcoin address, similarly to access Ethereum blockchain one needs Ethereum address. Thus, a wallet helps to access all transactions associated with that address on respective blockchain [52].

As commented, wallets store one or more cryptocurrency-unique public addresses. A public address is a hexadecimal string with a combination of numbers and letters, using both upper case and lower case. It needs to be shared publicly to receive cryptocurrency. The software within the cryptocurrency wallet is connected directly to the blockchain. The connected wallet allows user to read transactions and to submit transactions to the blockchain. More specifically, wallet is client software to blockchain platform to facilitate operations on the blockchain.

A secret key associated with a public address known as private key is also stored in the wallet. A private address is also a hexadecimal string. It is difficult to remember by the human beings, therefore it is store in wallet software or with third party vendors in case of online wallets. A user need to know a pin associated with each private key to access the wallet. A pin is easy to remember small string similar to ATM pin. In general, a wallet is like an online bank account platform, the public address is similar to an account number, the blockchain is like the bank's ledger, and with custodial wallets, the custodian is a bit like a banker.

In a nutshell, a crypto-wallet interacts with the blockchain with stored public address; and transactions on blockchain are signed with the private key associated with the public key [52].

### 9.1.4.1 Types of Cryptocurrency Wallets

According to user needs, cryptocurrencies are handled using different type of crypto-wallets [52]:

- **Desktop wallet:** It is a software that can be downloaded and installed on a PC or a laptop. It is easy to use from a single computer in which it is installed. Desktop wallets offer one of the maximum tiers of security. However, if computer is hacked or receives a virus, there is possibility of loosing keys and thereby funds.
- **Online wallet:** It is a software that runs on the cloud and is theoretically easy to use from any location. Although it is handy to access, such wallets save your private and public keys on-line and are controlled by a third party. It subjects to on-line risk of hacking or compromising third party.
- **Mobile wallet:** It is a software that runs as a mobile application on your phone. It is more convenient and user friendly to use for transactions and payments like retail stores. Mobile wallet software has a simple user interface than desktop wallets.
- **Hardware wallet:** These wallets differ from software wallets in terms of key storage and management as keys are saved on a device like the one shown in Fig. 9.6. Although hardware wallets make transactions online, public and private keys are saved offline. Hence, it delivers immoderate security, since it is absolutely a physical copy or printout of public and private keys of the user. It also refers to a bit of software that is used to securely generate multiple keys which is then subsequently printed on paper. Use of a paper wallet is relatively trustworthy and it provides the highest stage of security as compared to above wallets, despite compromising or loss of paper wallet is still risky.



Figure 9.6: KeepKey Hardware Wallet [2]

Digital wallets are not registered under the law of any country, so it is important to cautiously choose the cryptocurrency wallet for digitized transactions. It is always better to choose a universal cryptocurrency wallet that supports more than one cryptocurrency.

Thus, when choosing the best cryptocurrency wallets to operate with a bot, it is highly recommended to make use of two types of wallets: a web-based wallet and a hardware wallet. On the web-based wallet, the immediate cash that is used to operate is stored. Whereas on the hardware wallet, it is stored the bulk of cryptocurrency volume. Hence, via the web-based wallet, it is possible to trade to and from other exchange based wallets, and then transferring trading profits from the web-wallet onto the hardware wallet for long term safekeeping [52].

### 9.1.5 Proof-of-Work against Proof-of-Stake

Proof-of-Work (PoW) and Proof-of-Stake (PoS) are both cryptocurrency consensus mechanisms<sup>1</sup> used to confirm transactions and create new blocks in a blockchain<sup>2</sup>, so that it is secured. Their main difference is how they choose and qualify users to add transactions.

#### 9.1.5.1 Proof-of-Work

PoW is the original crypto consensus mechanism. Proof-of-Work and mining are closely related ideas. The reason why it is called “Proof-of-Work” is because the network requires a huge amount of processing power [11]. The work, in this case, consists of generating a hash (a long string of characters) that matches the target hash for the current block. The crypto miner who does this wins the right to add that block to the blockchain and receive rewards [55], thus making the blockchain secure and causing it to be verified.

To accomplish this, miners use mining devices that quickly generate computations. The aim is to be the first miner with the target hash because that miner is the one who can update the blockchain and receive crypto rewards. Hence, the reason why PoW in cryptocurrency works satisfactorily is because finding the target hash is challenging enough to prevent the manipulation of transaction records, but at the same time, verifying it is not, since once a target hash is found, it is straightforward for other miners to check it [55].

Proof-of-Work has some powerful advantages. Firstly, it is a proven, robust way of maintaining a secure decentralized blockchain. Additionally, as the value of a cryptocurrency grows, more miners are incentivized to join the network, increasing its power and security.

By contrast, it is an energy-intensive process that can have trouble scaling to accommodate the vast number of transactions that huge blockchains can generate. Also, because of the amount of processing power involved, it becomes impractical for any individual or group to meddle with a valuable cryptocurrency’s blockchain [11]. For that reason, alternatives have been developed, the most popular of which is the aforementioned Proof-of-Stake.

---

<sup>1</sup>A consensus mechanism is a method for validating entries into a distributed database and keeping the database secure [41]

<sup>2</sup>Please read Sec. 9.1.2.1 Transaction Process for more detailed information about the transaction process of cryptocurrency in blockchain.

### 9.1.5.2 Proof-of-Stake

Proof-of-Stake is also a cryptocurrency consensus mechanism for processing transactions and creating new blocks in a blockchain. It was created as an alternative to PoW, since it reduces the amount of computational work needed to verify blocks and transactions. Under PoW, blockchain is kept blockchain, while PoS changes the way blocks are verified using the machines of coin owners, so there does not need to be as much computational work done. Instead, the owners offer their coins as collateral (staking) for the chance to validate blocks and then become validators [41].

Validators are selected randomly to confirm transactions and validate block information. This system randomizes who gets to collect fees rather than using a competitive rewards-based mechanism like proof-of-work. Thus, to become a validator, a coin owner must "stake" a specific amount of coins [41].

Staking is the process of locking up crypto holdings in order to obtain rewards or earn interest. Staking is another way to describe validating those transactions on a blockchain. But achieving that consensus requires participants. That's what staking is—investors who actively hold onto, or lock up their crypto holdings in their crypto wallet are participating in these networks' consensus-taking processes. Stakers are, in essence, approving and verifying transactions on the blockchain [50].

Hence, each blockchain has their specific minimum number value of its cryptocurrency to hold on staking, so as to be able to become a validator. Furthermore, it is relevant to note that blocks are validated by more than one validator, and when a specific number of the validators verify that the block is accurate, it is finalized and closed.

With all of the above, it is evidently that PoS is seen as less risky regarding the potential for an attack on the network, as it structures compensation in a way that makes an attack less advantageous. Furthermore, cryptocurrency owners validate block transactions based on the number of staked coins, being the next block writer on the blockchain selected at random, with higher odds being assigned to nodes with larger stake positions [41].

### 9.1.5.3 PoW & PoS Comparison

Both consensus mechanisms help blockchains synchronize data, validate information, and process transactions. Each method has proven to be successful at maintaining a blockchain, although each has pros and cons. However, the two algorithms have very differing approaches.

Under PoS, block creators are called validators. A validator checks transactions, verifies activity, votes on outcomes, and maintains records. Under PoW, block creators are called miners. Miners work to solve for the hash, a cryptographic number, to verify transactions. In return for solving the hash, they are rewarded with a coin.

To "buy into" the position of becoming a block creator, you need only own enough coins or tokens to become a validator on a PoS blockchain. For PoW, miners must

invest in processing equipment (Fig. 9.7) and incur hefty energy charges to power the machines attempting to solve the computations.



Figure 9.7: Cryptocurrency miners [32]

The equipment and energy costs under PoW mechanisms are expensive, limiting access to mining and strengthening the security of the blockchain. PoS blockchains reduce the amount of processing power needed to validate block information and transactions. The mechanism also lowers network congestion and removes the rewards-based incentive PoW blockchains have [11].

Taking all of the above into account, Table (9.1) compares the two main consensus mechanisms used in blockchains.

Table 9.1: PoW & PoS comparison [11]

Proof-of-Work	Proof-of-Stake
Block creators are called miners	Block creators are called validators
Participants must buy equipment and energy to become a miner	Participants must own coins or tokens to become a validator
Not energy efficient	Energy efficient
Robust security due to expensive upfront requirement	Security through community control
Miners receive block rewards	Validators receive transactions fees as rewards

### 9.1.6 History

Although relatively short, cryptocurrencies have a history of a few decades, which is worth knowing in order to contextualize them. Hence, the history presented below is a summary taken from [16].

### 9.1.6.1 The Idea for Cryptocurrency

The idea for cryptocurrency first emerged in 1983, when American cryptographer David Chaum published a conference paper outlining an early form of anonymous cryptographic electronic money. The concept was for a currency that could be sent untraceably and in a manner that did not require centralized entities (i.e. banks). In 1995, Chaum built on his early ideas and developed a proto-cryptocurrency called DigiCash. It required user software to withdraw funds from a bank and required specific encrypted keys before said funds could be sent to a recipient.

Bit Gold, often deemed a direct precursor to Bitcoin, was designed in 1998 by Nick Szabo. It required a participant to dedicate computer power to solving cryptographic puzzles, and those who solved the puzzle received a reward. Combined with Chaum's work, it results in something that comes very close to resembling Bitcoin.

But Szabo could not solve the infamous double-spending problem (digital data can be copied and pasted) without the use of a central authority. As such, it was not until a decade later when a mysterious person or group, using the pseudonym Satoshi Nakamoto, set the history of Bitcoin and later cryptocurrencies in motion, by publishing a white paper called "Bitcoin – A Peer to Peer Electronic Cash System" [47].

### 9.1.6.2 The Beginning (2008-2010)

On 31<sup>st</sup> October 2008, Satoshi Nakamoto published the Bitcoin (BTC) white paper, describing the functionality of the Bitcoin blockchain network. Satoshi formally began work on the bitcoin project on 18<sup>th</sup> August 2008, when they purchased *Bitcoin.org*.

The history of Bitcoin was underway. Satoshi Nakamoto mined the first block of the Bitcoin network on 3<sup>rd</sup> January 2009. They embedded a headline from *The Times* newspaper in this initial block, making a permanent reference to the economic conditions - involving bank bailouts and a centralized financial system - that Bitcoin was partly a reaction against.

This first block - which resulted in 50 Bitcoins being mined - is now referred to as the Genesis Block. Bitcoin had virtually no value at this time, as well as for the first few months of its existence. Six months after Bitcoin became tradeable, in April 2010, the value of one BTC was just under 14 cents. By early November, the price surged to 36 cents before settling at around 29 cents.

### 9.1.6.3 The Market Begins to Form (2010-2014)

While it was not yet worth much, Bitcoin was showing it had real world value. In February 2011, it rose to \$1.06 before coming back down to roughly 87 cents. In the spring, in part due to a Forbes story on the new "Cryptocurrency," the price took off. From early April to the end of May, the value of BTC rose from 86 cents to \$8.89.

On 1<sup>st</sup> June, after Gawker published a story about the appeal of currency in the online drug dealing community, the price more than tripled in a week, to about \$27.

The market value of BTCs in circulation approached nearly \$130 million. However, by September 2011, the value had dropped back down to around \$4.77.

In October of that same year, Litecoin (LTC) appeared, one of many forks (i.e. updated versions) of Bitcoin. Litecoin soon became the second-biggest cryptocurrency by market cap, with the earliest archive of CoinMarketCap [14] (from May 2013) showing PPCoin, Namecoin and 10 others trailing in the distance. Such cryptocurrencies were quickly named as altcoins, with some forked from Bitcoin and others based on new code. Further information on the types of cryptocurrencies is detailed in Sec. 9.1.7 Types of cryptocurrency.

In 2012, Bitcoin prices grew steadily, and in September of that year the Bitcoin Foundation was established to promote the development and uptake of Bitcoin. Then known as OpenCoin, Ripple (XRP) was also launched that year, with the project attracting venture capital the next year.

In 2013, amid federal, criminal, regulatory, and software related issues, the price of BTC constantly rose and crashed. On 19<sup>th</sup> November, its price reached \$755, only to crash down to \$378 the same day. By 30<sup>th</sup> November, it was all the way up to \$1,163 again. This was, however, the beginning of another long-term crash that ended with Bitcoin dropping back down to \$152 by January 2015.

#### 9.1.6.4 Scams Dominate Headlines (2014-2016)

Though intentional, anonymity and lack of centralized control make digital currency highly attractive to criminals. In January 2014, Mt.Gox - then the largest BTC exchange in the world - collapsed and declared bankruptcy, having lost 850,000 BTC. While it is not known what exactly happened, it is likely that the missing BTCs were stolen gradually over time, beginning in 2011, and resold on various exchanges for cash (including Mt.Gox), until one day Mt.Gox checked their wallets and found they were empty. Its CEO Mark Karpeles was charged with embezzlement in 2017, but acquitted in 2019, so the destination of the missing BTC remains a mystery.

While the hack was not a singular event, it has served as a cautionary tale, and security on exchanges is much improved. Though smaller exchanges continue to be hacked even today, larger platforms now provide more guarantees on their reserve holdings in case of breaches. This includes the Secure Asset Fund for Users on Binance, for example, which is an emergency insurance fund.

Crypto traders are advised to use a hardware or software wallet<sup>3</sup> to safely store their cryptocurrency rather than storing them on an exchange. Wallets such as these were not as accessible during this early period in the history of cryptocurrency.

#### 9.1.6.5 Bitcoin Ascends to Worldwide Phenomenon (2016-2018)

Bitcoin prices rose steadily year over year, going from \$434 in January 2016, to \$998 in January 2017. In July 2017, a software upgrade to Bitcoin was approved, with

---

<sup>3</sup>Readers interested in further information on hardware wallets are referred to Sec. 9.1.4.1 Types of Cryptocurrency Wallets.

the aim being to support development of the Lightning Network (a layer-two scaling solution) as well as improve security.

A week after the upgrade was activated in August, Bitcoin was trading at around \$2,700. By 17<sup>th</sup> December 2017, Bitcoin reached an astronomical all-time high (ATH) of just under \$20,000.

During this same time, a new blockchain project called Ethereum (ETH) was making noise in the cryptocurrency sphere, having quickly become the number two cryptocurrency by market cap since launching in July 2015. It brought smart contracts<sup>4</sup> to cryptocurrency, opening a wide array of potential use cases and generating over 200,000 different projects (and counting). In contrast to Bitcoin, Ethereum enables additional platforms to launch and operate on its own chain, each with their own cryptocurrencies and their own use cases. This was a model widely copied by other new blockchains, with Cardano (ADA) or Tezos (XTZ) launched during this period.

#### 9.1.6.6 Bust and Recovery, and Bust and Recovery (2018-Present)

Bitcoin was not able to sustain its all-time high of \$19,783. Likewise, Ethereum, which reached its own ATH in January 2018 of around \$1,400, was also not able to maintain its newfound level for long. Financial regulations and security concerns (due to semi-regular exchange hacks) contributed to the market-wide decline, and by the end of 2018, BTC had dropped down to around \$3,700.

However, prices did not stay down for too long. Starting from late 2020, BTC enjoyed something of a renaissance, beginning with the announcement of Microstrategy in August that it had bought BTC worth \$250 million. This kicked off a bull market that was joined by the rest of the market, with prices boosted further by the purchase of Tesla of \$1.5 billion in BTC in early 2021. It was in November of that year that BTC reached its current record high of \$69,000, as it is displayed in Fig. 9.8.

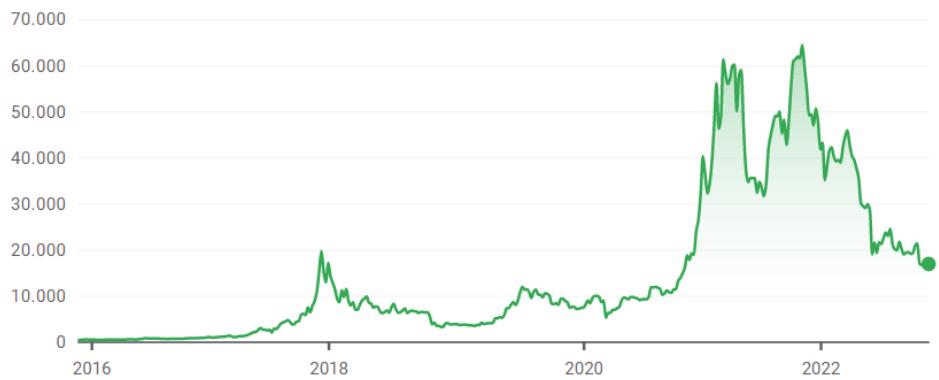


Figure 9.8: Evolution of BTC-USD price [28]

---

<sup>4</sup>A smart contract, like any contract, establishes the terms of an agreement. But unlike a traditional contract, terms of a smart contract are executed as code running on a blockchain, and allow developers to build apps that take advantage of blockchain security, reliability, and accessibility while offering sophisticated peer-to-peer functionality - everything from loans and insurance to logistics and gaming [12].

The market has fallen once again since this high, dragged down by macroeconomic concerns resulting from high inflation, rising interest rates and the specter of war. That said, with global stock markets also falling in late 2021 and 2022, parallel fall of cryptocurrencies shows that the sector is becoming increasingly entwined with traditional financial markets.

And while the volatility of cryptocurrencies is both attractive and potentially devastating, the underlying technology behind them all, blockchain, has the power to change many sectors of our society. Whether it is providing accessible and affordable financial exchange options, securing your own funds so that no one but you can access them, or providing accurate data for your insurance quote, blockchain technology has the potential to be used in almost every area of the economy.

As the market becomes more stable with increased knowledge, and with the introduction of new areas such as stablecoins and decentralized finance (DeFi), it is simple to get excited about cryptocurrency and its potential from an investment and technological perspective.

### 9.1.7 Types of cryptocurrency

The term *cryptocurrency* is often used interchangeably with *token* and *coin*, regardless of not being the same. Thus, cryptocurrencies generally fall into one of these two categories [51]:

- **Coins:** Programmable assets that live within the blockchain of a their own platform.
- **Tokens:** Programmable assets that live within the blockchain of a given platform.

#### 9.1.7.1 Coins

Crypto coins are strings of computer code that can represent an asset, concept, or project - whether tangible, virtual, or digital - intended for various uses and with varying valuations. Originally, these coins were meant to function as a type of currency [51].

Coins are built on their own blockchain and were originally intended as a form of currency. Generally, any blockchain-based cryptocurrency that is not Bitcoin is referred to as an altcoin. Thus, an altcoin is an alternative digital currency to Bitcoin.

A digital coin is created on its own blockchain and acts much like fiat (traditional money). Coins can be used to store value and as a means of exchange between two parties doing business with each other [51]. Examples of coins include Bitcoin and Litecoin, which is an altcoin, among some of those shown in Fig. 9.9.

In addition to altcoins, it is worth mentioning stablecoins. Stablecoins are coins whose value is pegged, or tied, to that of another currency, commodity, or financial instrument. They aim to provide an alternative to the high volatility of the most

popular coins, which has made crypto investments less suitable for common transactions [39]. Most common stablecoins are Tether (USDT) and USD Coin (USDC), which both replicate to the real USD.



Figure 9.9: Most common coin symbols [17]

### 9.1.7.2 Tokens

Tokens are the digital representations of a particular asset or utility in a blockchain. All tokens can be termed altcoins, but they are differentiated by residing on top of another blockchain and not being native to the blockchain on which they reside.

Tokens are usually created and distributed through an initial coin offering (ICO), much like an initial public offering (IPO) for stock. They can be represented as value tokens, security tokens or utility tokens.

Therefore, tokens are created on an existing blockchain (not their own), and can function in many more ways than acting as currency. Instead of representing an exchange of value, tokens are considered programmable assets on which you may create and execute unique smart contracts. These contracts can establish ownership of assets outside the blockchain network.

Tokens can represent units of value - including real-world items like electricity, money, points, coins, digital assets, and more - and can be sent and received. They can be used as part of a software application - such as granting access to an app, verifying identity, or tracking products moving through a supply chain. They can also represent digital art - as with non-fungible tokens<sup>5</sup> (NFTs) [51].

---

<sup>5</sup>Non-fungible tokens (NFTs) are cryptographic assets on a blockchain with unique identification codes and metadata that distinguish them from each other. Unlike cryptocurrencies, they cannot be traded or exchanged at equivalency. This differs from fungible tokens like cryptocurrencies, which are identical to each other and, therefore, can serve as a medium for commercial transactions [38].

### 9.1.7.3 Largest Cryptocurrencies by Market Cap

Once understood the existing types of cryptocurrency, Table (9.2) is presented so as to classify the 15 first cryptocurrencies by more market cap.

Table 9.2: Cryptocurrency with more market cap on 09/12/2022 at 11 am [14]

#	Name	Price	Market Cap	Circulating Supply
1	Bitcoin (BTC)	\$17,208.72	\$331.11B	1,213,714 BTC
2	Ethereum (ETH)	\$1,281.16	\$156.78B	5,002,449 ETH
3	Tether (USDT)	\$1.00	\$65.71B	26,103,185,150 USDT
4	Binance Coin (BNB)	\$290.49	\$46.47B	2,133,753 BNB
5	USD Coin (USDC)	\$1.00	\$42.79B	2,210,474,750 USDC
6	Binance USD (BUSD)	\$1.00	\$22.06B	5,887,924,841 BUSD
7	XRP (XRP)	\$0.3914	\$19.67B	1,930,489,911 XRP
8	Dogecoin (DOGE)	\$0.09826	\$13.04B	4,465,554,116 DOGE
9	Cardano (ADA)	\$0.3146	\$10.85B	459,814,823 ADA
10	Polygon (MATIC)	\$0.9238	\$8.07B	285,644,523 MATIC
11	Polkadot (DOT)	\$5.40	\$6.19B	27,277,787 DOT
12	Dai (DAI)	\$1.00	\$5.87B	157,076,833 DAI
13	Litecoin (LTC)	\$78.12	\$5.61B	6,960,013 LTC
14	Shiba Inu (SHIB)	\$0.000009312	\$5.11B	9,014,342,996,273 SHIB
15	Solana (SOL)	\$13.87	\$5.08B	13,756,547 SOL

At first glance, it is prominent that both Bitcoin and Ethereum encompass the largest amount of capital within the cryptocurrency market, with a 38.5% and 18.3% of the total market cap, respectively. This reflects the dominance of both cryptos, which are undoubtedly the most established ones.

Indeed, such is the importance of Bitcoin that traders used BTC dominance to figure out whether altcoins are trending bullish or bearish against Bitcoin. For example, one popular theory is that if altcoins are going up, the crypto market is headed for a bull market. In 2017, for example, a major drop in BTC dominance indicated a sharp rise in altcoin prices (not a fall in BTC price), which coincided with the entire market entering a bullish phase.

### Bitcoin

As commented, Bitcoin is the clear leader in the crypto sector. It is also the very first cryptocurrency. Bitcoin launched in 2009; created by a person (or possibly a group) that goes by the pseudonym Satoshi Nakamoto. As of June 2022, there are slightly more than 19 million Bitcoin tokens in circulation, against a capped limit of 21 million. Almost a thousand new bitcoins are mined each day, bringing Bitcoin ever closer to its maximum finite number [51].

### Ethereum

Like Bitcoin, Ethereum is a blockchain network. But Ethereum was designed as a programmable blockchain - meaning it was not created to support a currency,

but rather to enable the network's users to create, publish, monetize, and deploy decentralized applications (dApps). ETH was developed as a form of payment on the Ethereum platform. It might be helpful to think of ETH as a kind of fuel that powers the Ethereum blockchain. Ethereum has helped to launch many initial coin offerings because many ICOs are built on the Ethereum blockchain. Ethereum has also been the blockchain behind the boom in non-fungible tokens (NFTs).

As the two most widely known blockchains and cryptocurrencies, many people often directly compare Ethereum and Bitcoin against each other. In reality, Bitcoin and Ethereum are designed to achieve different goals, and in many ways can be regarded as complementary forces. Bitcoin is a peer-to-peer digital cash network, which facilitates transactions without the need for a central authority. At turn, Ethereum is often referred to as the world computer, since it iterates on the technology of Bitcoin while introducing smart contracts, which allow for building dApps that span a broad range of crowdfunding platforms, financial instruments, digital games and collectibles, and decentralized marketplaces [51].

### Tether

Tether was the first cryptocurrency marketed as a stablecoin - a breed of crypto known as fiat-collateralized stablecoins. The value of the Tether is pegged to a fiat currency - in this case, the U.S. dollar. Tether is the largest stablecoin in the world; in 2022, the majority of cryptocurrencies are traded using Tether.

Like other stablecoins, Tether is designed to offer stability, transparency, and lower transaction fees to users. In fact, Tether was originally used to avoid the extreme volatility of the crypto market [51].

### Binance Coin

Binance is one of the biggest cryptocurrency exchanges in the world<sup>6</sup>. The Binance Coin (BNB) was created as a utility token for use as a medium of exchange on Binance. It was initially built on the Ethereum blockchain, but now lives on Binance's own blockchain platform. Originally, BNB allowed traders to get discounts on trading fees on Binance, but now it also can be used for payments, to book travel, for entertainment, online services, and financial services.

As one of the top five cryptocurrencies by market cap in 2022, BNB has developed a wide range of use cases and real-world applications. But, as with other digital assets, this crypto platform has also faced regulatory hurdles here and abroad.

BNB was created with a maximum of 200 million tokens, about half of which were made available to investors during its ICO. Every quarter, to drive demand, Binance buys back and then burns - permanently destroys, or removes from circulation - some of the coins it holds, aiming at reducing the overall supply and thus increasing the value of the remaining tokens, as assets tend to rise in price whenever the circulating supply falls, and they become more scarce [51].

---

<sup>6</sup>Please read Sec. 9.1.8 Cryptocurrency Exchange for more information about Binance.

### 9.1.8 Cryptocurrency Exchange

Cryptocurrency exchanges are privately-owned platforms that facilitate the trading of cryptocurrencies for other crypto assets, including digital and fiat currencies and NFTs [9].

Basically, there are two kind of exchanges: Centralized Cryptocurrency Exchanges (CEX) and Decentralized Cryptocurrency Exchanges (DEX), each of which offers certain advantages and disadvantages.

#### 9.1.8.1 Centralized Cryptocurrency Exchange

Centralized cryptocurrency exchanges act as an intermediary between a buyer and a seller and make money through commissions and transaction fees. A CEX can be imagined to be similar to a stock exchange but for digital assets [9].

Most popular crypto exchanges are Binance, Coinbase, Kraken and KuCoin. Much like stock trading websites or apps, these exchanges allow cryptocurrency investors to buy and sell digital assets at the prevailing price, called spot, or to leave orders that get executed when the asset gets to the desired price target, called limit orders.

CEXs operate using an order book system, which means that buy and sell orders are listed and sorted by the intended buy or sell price. The matching engine of the exchange then matches buyers and sellers based on the best executable price given the desired lot size. Hence, the price of a digital asset will depend on the supply and demand of that asset versus another, whether it be fiat currency or cryptocurrency.

Hence, CEXs decide which digital asset is allowed trading in, which provides a small measure of comfort that unscrupulous digital assets may be excluded from the CEX.

#### Binance

According to traffic, liquidity<sup>7</sup> and trading volumes, Binance is the best CEX as of December 2022 [14]. It was founded in 2017 in Hong Kong, and features a strong focus on altcoin trading, offering crypto-to-crypto trading in more than 600 cryptocurrencies and virtual tokens.

The Binance exchange has among the lowest transaction fees for cryptocurrency exchanges. It also has high liquidity and offers discounts to users who pay in the native BNB cryptocurrency tokens. Moreover, with high standards of safety and security and multi-tier and multi-clustered architecture, Binance delivers high processing throughput with the capacity to process around 1.4 million orders per second.

To start trading, users have to complete the necessary KYC requirements. Upon successful trading account creation, users can add cryptocurrency funds to their public wallet address, provided by Binance, to start trading [35].

---

<sup>7</sup>Liquidity is the efficiency or ease with which an asset or security can be converted into ready cash without affecting its market price [40].

### 9.1.8.2 Decentralized Cryptocurrency Exchange

A decentralized exchange is another type of exchange that allows peer-to-peer transactions directly from your digital wallet without going through an intermediary. Examples of DEXs include Uniswap, PancakeSwap, dYdX, and Kyber.

These decentralized exchanges rely on smart contracts, self-executing pieces of code on a blockchain. These smart contracts allow for more privacy and less slippage (another term for transaction costs) than a centralized cryptocurrency exchange.

On the other hand, even though smart contracts are rules-based, the lack of an intermediary third party means that the user is left to their own, so DEXs are meant for sophisticated investors [9].

### 9.1.8.3 CEX against DEX

Once understood the difference between CEXs and DEXs, it is pertinent to analyze the advantages and disadvantages of each type of crypto exchange [9].

#### Advantages of CEX

- **User-friendly:** Centralized exchanges offer beginner investors a familiar, friendly way of trading and investing in cryptocurrencies. As opposed to using crypto wallets and peer-to-peer transactions, which can be complex, users of centralized exchanges can log into their accounts, view their account balances, and make transactions through applications and websites.
- **Reliable:** Centralized exchanges offer an extra layer of security and reliability when it comes to transactions and trading. By facilitating the transaction through a developed, centralized platform, centralized exchanges offer higher levels of comfort.
- **Leverage:** One of the other benefits of certain CEXs is the option to leverage your investments using borrowed money from the exchange, called margin trading. It allows investors to reap higher returns, but losses can also be amplified.

#### Disadvantages of CEX

- **Hacking risk:** Centralized exchanges are operated by companies that are responsible for the holdings of their customers. Large exchanges usually hold billions of dollars worth of Bitcoin, making them a target for hackers and theft.
- **Transaction fees:** Unlike peer-to-peer transactions, centralized exchanges often charge high transaction fees for their services and convenience, which can be especially high when trading in large amounts.
- **Custody of digital assets and risk of fraud:** Most CEXs will hold digital assets as a custodian in their own digital wallet rather than allow users to store their private keys on their own digital wallet. While more convenient

when wanting to trade, there are drawbacks, namely the risk of the centralized cryptocurrency exchange failing and fraud.

### Advantages of DEX

- **Custody:** Users of decentralized exchanges do not need to transfer their assets to a third party. Therefore, there is no risk of a company or organization being hacked, and users are assured of greater safety from hacking, failure, fraud, or theft.
- **Preventing market manipulation:** Due to their nature of allowing for the peer-to-peer exchange of cryptocurrencies, decentralized exchanges prevent market manipulation, protecting users from fake trading and wash trading.
- **Less censorship:** Decentralized exchanges do not require customers to fill out know-your-customer (KYC) forms, offering privacy and anonymity to users. Since DEXs do not exercise censorship, more cryptocurrencies and digital assets are available than through a CEX. As a matter of fact, many altcoins are only available on DEXs.

### Disadvantages of DEX

- **Complexity:** Users of decentralized exchanges must remember the keys and passwords to their crypto wallets, or their assets are lost forever and cannot be recovered. They require the user to learn and get familiar with the platform and the process, unlike centralized exchanges, which offer a more convenient and user-friendly process.
- **Lack of fiat payments:** DEXs are best for investors looking to switch from one digital asset to another and not well suited for someone looking to buy or sell digital assets with fiat currency, called on and off-ramping. It makes them less convenient for users that do not already hold cryptocurrencies.
- **Liquidity struggles:** Some 99% of crypto transactions are facilitated by centralized exchanges, which suggests that they are accountable for the majority of the trading volume. Due to the lack of volume, decentralized exchanges often lack liquidity, and it can be difficult to find buyers and sellers when trading volumes are low.

With all of the above, it is undoubtedly that both type of exchanges can perfectly coexist, and in fact they are complementary to each other, since they offer the user flexibility, as they adapt to the specific needs and knowledge of the user.

Thus, the exchange that is used will depend on the required application, and that is why it is vitally crucial to actively search for possible exchanges and select the one that best suits the way that the user wants to operate with cryptocurrencies.

## 9.2 Cryptocurrency Bot

A cryptocurrency bot is a piece of software that looks for opportunities in the market and capitalizes on them to generate profit. In other words, they allow automatically trading with cryptocurrency. Crypto bots are more nimble and agile than a person could ever be alone, and also can watch various different crypto pairs at once and spot different opportunities in an instant.

As commented, crypto bots are basically programs that are executed to yield specific results in trades. This implies that a technical analysis strategy is required to follow, in order to provide the bot with a set of rules to execute. At that point, it is mandatory to make sure that the theoretical strategy work in reality before deploying it in normal trading. This process is called *backtesting*, and is developed in depth in Sec. 9.2.1 Backtesting.

In addition to the above, it is essential to note that crypto bots usually relies on historical data in order to both operate and backtest. For that reason, it is crucial to have a financial database to obtain the appropriate values for the cryptocurrencies with which you intend to operate. With that in mind, Yahoo! Finance database is presented in Sec. 9.2.2 Yahoo! Finance.

### 9.2.1 Backtesting

Backtesting is the general method for seeing how well a strategy or model would have worked. Backtesting assesses the viability of a trading strategy by discovering how it would play out using historical data. If backtesting works, traders and analysts may have the confidence to employ it going forward [34].

Hence, it is clearly that backtesting allows a trader to simulate a trading strategy using historical data to generate results and analyze risk and profitability before risking any actual capital. A well-conducted backtest that yields positive results assures traders that the strategy is fundamentally sound and is likely to yield profits when implemented in reality. In contrast, a well-conducted backtest that yields suboptimal results will prompt traders to alter or reject the strategy.

As long as a trading idea can be quantified, it can be backtested. Some traders and investors may seek the expertise of a qualified programmer to develop the idea into a testable form. Typically, this involves a programmer coding the idea into the proprietary language hosted by the trading platform.

The ideal backtest chooses sample data from a relevant time period of a duration that reflects a variety of market conditions. In this way, one can better judge whether the results of the backtest represent a fluke or sound trading.

A backtest should consider all trading costs, however insignificant, as these can add up over the course of the backtesting period and drastically affect the appearance of a strategy's profitability. Traders should ensure that their backtesting software accounts for these costs.

Out-of-sample testing and forward performance testing provide further confirmation

regarding a system's effectiveness and can show a system's true colors before real cash is on the line. A strong correlation between backtesting, out-of-sample, and forward performance testing results is vital for determining the viability of a trading system [34].

In order the backtesting results to be analyzed, the use of reward and risk metrics arises. As reward metrics, it is common to work with the annualized return, the average take profit/stop-loss, which is the relationship between the average take profits divided by the average stop-losses obtained in the total of trades, and the profitability, which is the percentage of winning trades. On the other hand, risk metrics basically consist of the annualized volatility, understood as the standard deviation of daily returns of the model in a year, and the maximum drawdown, which is the maximum negative percentage difference in total portfolio value in a day.

Lastly, two more ratios are of interest when analyzing the results of backtesting. They combine view of the returns and risk in one metric. Specifically, they are the Sharpe and Sortino ratios, which are calculated as:

$$\text{Sharpe Ratio} = \frac{\text{Annualized Net Return}}{\text{Annualized Volatility}} \quad (9.1)$$

$$\text{Sortino Ratio} = \frac{\text{Annualized Net Return}}{\text{Annualized Volatility of Negative Returns}} \quad (9.2)$$

As expected, it is convenient that these ratios achieve the higher possible values, since a higher annualized return will mean earning more money, and less volatility will mean operating with less risk. Hence, when they exceed a value of 2, the strategy is considered to operate with a correct return/risk ratio. Additionally, as a rule of thumb, it is better to have a Sharpe ratio greater than the Sortino ratio, since it is important that the volatility of the winning trades be lower so as to ensure consistency when trading.

### 9.2.2 Yahoo! Finance

Yahoo! Finance is a media property that is part of the Yahoo! network. It provides financial news, data and commentary including stock quotes, press releases, financial reports, and original content. It also offers some online tools for personal finance management. In 2017, Yahoo! Finance added the feature to look at news surrounding cryptocurrency. It lists over 9,000 unique coins including Bitcoin and Ethereum [59].

# Chapter 10

## Machine Learning and Statistics

The emergent price prediction schemes for cryptocurrency include strategies that are based on statistical and ML technologies.

Generally, the statistical models use mathematical equations to encode information extracted from the data, being thus able to quickly provide adequate models to estimate and predict the values of various economic variables.

On the other side, with the advancement of big data technology and artificial intelligence, numerous research studies have applied ML models to classification and prediction problems. Hence, many researchers have focused their efforts on applying these new techniques on financial markets.

With that, the aim of this chapter is to present some basic theoretical foundations on Machine Learning and statistics, which serve to better understand the implementation of the bot.

### 10.1 Machine Learning

Before delving into the field of ML, it is essential to know what Artificial Intelligence is. Artificial Intelligence (AI) is the simulation of human intelligence processes by machines, especially computer systems, which perform complex tasks in a way that is similar to how humans solve problems.

Hence, Machine Learning is a subfield of AI. ML is broadly defined as the capability of a machine to imitate intelligent human behaviour. Indeed, it is the science that makes computers learn from data. Instead of programming, step by step, each specific solution for each need raised, as is done in the conventional programming approach, the ML area is dedicated to the development of generic algorithms that can extract patterns from different types of data [6].

In order to be able to address any specific task, there are different types of classification and models. ML is divided into mainly four types: supervised ML, unsupervised ML, semi-supervised ML, and reinforcement ML. Despite the relevance of these kind of methods, this section is intended to focus on neural networks, that precisely one of

its methods is the one on which the strategy implemented for the backtesting of this project is based.

### 10.1.1 Neural Networks

Neural networks, also known as Artificial Neural Networks (ANNs) or Simulated Neural Networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

ANNs are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

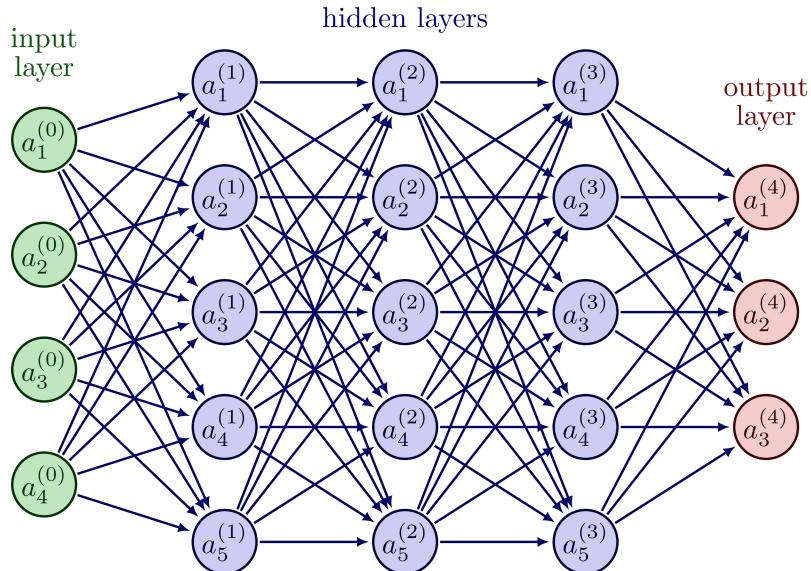


Figure 10.1: Neural networks layers' diagram [56]

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and AI, allowing us to classify and cluster data at a high velocity [30].

At a basic level, a neural network is comprised of four main components: inputs, weights, a bias or threshold, and an output, which must contain a non-linear activation function. Hence, the algebraic formula is:

$$\sum_{i=1}^m w_i x_i + \text{bias} = w_1 x_1 + w_2 x_2 + w_3 x_3 + \text{bias} \quad (10.1)$$

With that, neural networks are also divided into nine different types: Perceptron, Feed Forward Neural Network, Multilayer Perceptron, Convolutional Neural Network,

Radial Basis Functional Neural Network, Sequence to Sequence Models, Modular Neural Network, Long Short-Term Memory Network, and Recurrent Neural Network. Hence, only the last three are discussed below, since are the more relevant ones for trading bot applications.

### 10.1.2 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a type of ANN which uses sequential data or time series data to learn. They are distinguished by their “memory” as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of RNNs depend on the prior elements within the sequence, as can be seen in Fig. 10.1. While future events would also be helpful in determining the output of a given sequence, unidirectional RNNs cannot account for these events in their predictions.

Another distinguishing characteristic of recurrent networks is that they share parameters across each layer of the network. Specifically, they share the same weight parameter within each layer of the network. That said, these weights are still adjusted in the through the processes of backpropagation and gradient descent to facilitate Reinforcement Learning [31].

### 10.1.3 Long Short-Term Memory Network

Long Short-Term Memory (LSTM) networks are a type of RNN that uses special units in addition to standard units. LSTM units include a ‘memory cell’ that can maintain information in memory for long periods of time. A set of gates is used to control when information enters the memory when it is output, and when it is forgotten, as can be observed in Fig. 10.2.

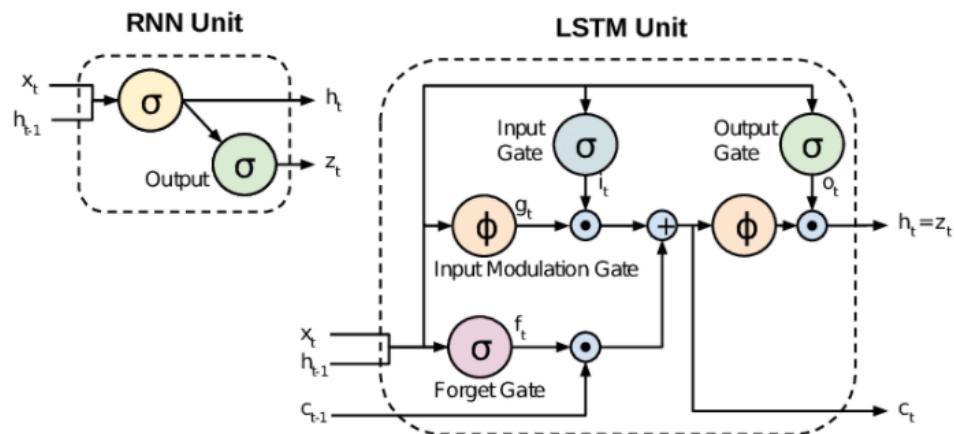


Figure 10.2: LSTM Network diagram [29]

There are three types of gates: input gate, output gate and forget gate. The input

gate decides how many information from the last sample will be kept in memory; the output gate regulates the amount of data passed to the next layer; and the forget gate controls the tearing rate of memory stored. This architecture allows them to learn longer-term dependencies.

#### 10.1.4 Modular Neural Network

A modular neural network has a number of different networks that function independently and perform sub-tasks. The different networks do not really interact with or signal each other during the computation process. They work independently towards achieving the output, as can be seen in Fig. 10.3.

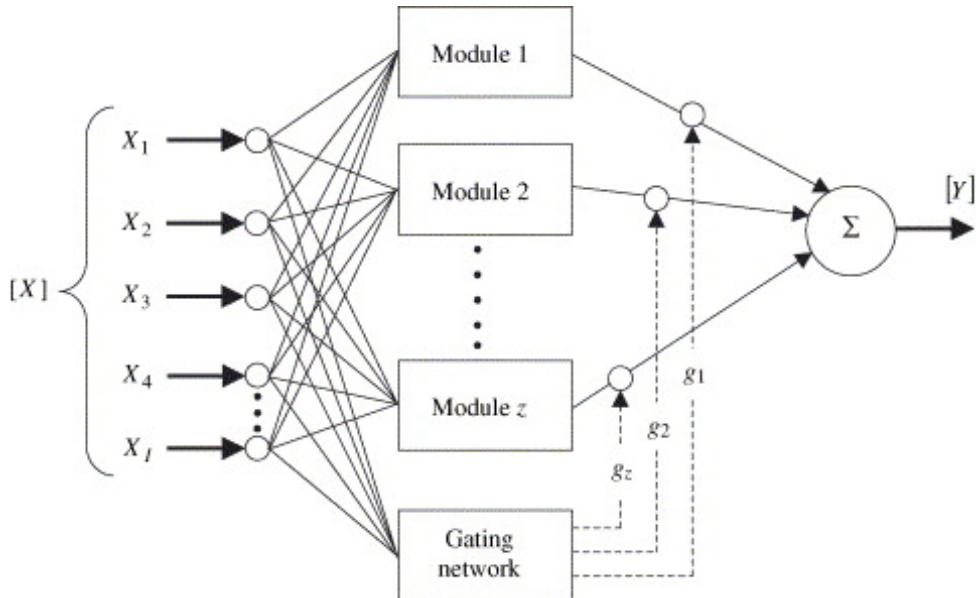


Figure 10.3: Modular Neural Network diagram [29]

As a result, a large and complex computational process are done significantly faster by breaking it down into independent components. The computation speed increases because the networks are not interacting with or even connected to each other [29].

## 10.2 Statistical Concepts

This section is intended to clarify some important statistical concepts for this project. It is assumed that the reader has an advanced level of statistics, and therefore basic concepts of statistics are not explained, but instead focuses on those that may be somewhat more specific.

Firstly, a statistical model used for time-series analysis is discussed. Later, a brief explanation of what is a combination without repetition is presented.

### 10.2.1 ARIMA model

Before explaining the model itself, it is mandatory to make it clear that a linear statistical-model-based approach evaluates the linear relationship between prices and an explanatory variable. Hence, if multiple explanatory variables exist, it is possible to model the linear relationship between explanatory variables (independent) and response variables (dependent) with the help of a multiple linear model [43].

The commonly used linear statistical model for time-series analysis is the AutoRegressive Integrated Moving-Average (ARIMA) model, which uses time series data to either better understand the data set or to predict future trends.

An ARIMA model is a form of regression analysis that gauges the strength of one dependent variable relative to other changing variables. In order to be understood, each of its components are outlined [33]:

- **AutoRegression (AR):** refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
- **Integrated (I):** represents the differencing<sup>1</sup> of raw observations to allow the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).
- **Moving Average (MA):** incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Therefore, each component in ARIMA functions has a parameter with a standard notation. For ARIMA models, a standard notation would be ARIMA with  $p$ ,  $d$ , and  $q$ , where integer values substitute for the parameters to indicate the type of ARIMA model used. The parameters can be defined as:

- **p:** the number of lag observations in the model, also known as the lag order.
- **d:** the number of times the raw observations are differenced; also known as the degree of differencing.
- **q:** the size of the moving average window, also known as the order of the moving average.

In an ARIMA model, the data is differenced in order to make it stationary. A model that shows stationarity<sup>2</sup> is one that shows there is constancy to the data over time. Most economic and market data show trends, so the purpose of differencing is to remove any trends or seasonal structures. At that point, it is also relevant to note that ARIMA models are complicated and work best on very large data sets, so that computer algorithms and ML techniques are used to compute them.

With all of the above, ARIMA models have strong points and are good at forecasting

---

<sup>1</sup>Differencing is a method of transforming a time series dataset. It can be used to remove the series dependence on time, so-called temporal dependence. This includes structures like trends and seasonality.

<sup>2</sup>Stationarity means that the statistical properties of a process generating a time series do not change over time.

based on past circumstances, but there are reasons to be cautious when using ARIMA. On the one hand, this model is good for short-term forecasting, and apart from only needing historical data, it models non-stationary data. On the other hand, it is not built for long-term forecasting, the prediction of turning points is poor, and additionally it is computationally expensive.

### 10.2.2 Combination Without Repetition

A combination is a method used in statistics, which consists of finding the possible ways a data set can be ordered. Hence, a combination without repetition of  $n$  elements taken  $k$  is the different groups of  $k$  elements that can be formed by these  $n$  elements, so that two groups differ only if they have different elements [49]. They are represented as  $C_{n,k}$ , and follow that:

$$C_{n,k} = \binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (10.2)$$

# Chapter 11

## Python Libraries

A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer, thus eliminating the need for writing codes from scratch. Hence, Python libraries play a very vital role in fields of Machine Learning, Data Science, and Data Visualization, among others.

Bearing that in mind, the aim of this chapter is to state constancy of the libraries used to develop the trading bot code, in order to provide the reader with their knowledge, and assume their full understanding in the description of the algorithm in Chap. 13.

Therefore, the libraries are divided into five different categories: general libraries; data analysis libraries; visualization libraries; trading libraries; and Machine Learning libraries. All the information detailed below related to these libraries is extracted from the Python Documentation website [48].

### 11.1 General Libraries

#### **csv**

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases.

The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats.

The csv module’s *reader* and *writer* objects read and write sequences. Programmers can also read and write data in dictionary form using the *DictReader* and *DictWriter* classes.

## math

This module provides access to the standard mathematical constants and functions defined by the C standard. Programmers can use that module to perform various mathematical calculations, such as numeric, trigonometric, logarithmic, and exponential calculations.

## os

This module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system. The OS comes under Python's standard utility modules.

## random

The random module is a built-in module to generate the pseudo-random variables. It can be used perform some action randomly such as to get a random number, selecting a random elements from a list, or shuffling elements randomly.

## sys

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available.

## 11.2 Data Analysis Libraries

### pandas

It is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way towards this goal.

Here are the main characteristics that pandas presents:

- Easy handling of **missing data** in both floating and non-floating point data.
- **Size mutability**: Columns can be inserted and deleted from DataFrame and higher dimensional objects.
- Automatic and explicit **data alignment**: Objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations.
- Powerful, flexible **group by** functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data.

- Make it **easy to convert** ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects.
- Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets.
- Intuitive **merging** and **joining** data sets.
- Flexible **reshaping** and **pivoting** of data sets.
- **Hierarchical** labeling of axes (possible to have multiple labels per tick).
- Robust IO tools for loading data from **flat files** (CSV and delimited), **Excel files**, **databases**, and saving/loading data from the ultrafast **HDF5 format**.
- **Time series**-specific functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging.

## NumPy

NumPy (Numerical Python) is the fundamental package for scientific computing with Python. It is used for working with arrays, and also has function for working in domain of linear algebra, Fourier transform, and matrices.

## datetime

The datetime module is one of the most common modules to manipulate date and time object data. It supplies classes for manipulating dates and times, looping through a range of dates, parsing and format date strings, and more.

While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

## 11.3 Visualization Libraries

### matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It makes easy things easier and hard things possible. For instance, it allows:

- Creating publication quality plots.
- Making interactive figures that can zoom, pan, update.
- Customizing visual style and layout.
- Exporting to many file formats.
- Using a rich array of third-party packages built on Matplotlib.

## 11.4 Trading Libraries

### **backtrader**

It is a feature-rich Python framework for backtesting and trading. It allows focusing on writing reusable trading strategies, indicators and analyzers instead of having to spend time building infrastructure. In general, every complex component of ordinary backtesting can be created with a single line of code by calling special functions.

### **talib**

TA-Lib is widely used by trading software developers requiring to perform technical analysis of financial market data. Among its main features, it stands out:

- Includes 150+ indicators such as ADX, MACD, RSI, Stochastic, Bollinger Bands, etc.
- Candlestick pattern recognition.
- Open-source API for C/C++, Java, Perl, and Python.

### **yfinance**

yfinance is a popular open source library as a means to access the financial data available on Yahoo! Finance, which offers an excellent range of market data on stocks, bonds, currencies and cryptocurrencies. For further information, reader is encouraged to read Sec. 9.2.2 Yahoo! Finance.

### **binance**

This package aims at connecting Python with Binance exchange, through the previous generation of an API key in that exchange. It allows obtaining and reading specific user data, in order the trading bot to operate in Binance.

## 11.5 Machine Learning Libraries

### **scikit-learn**

scikit-learn is probably the most useful library for ML in Python, since it contains a lot of efficient tools for ML and statistical modeling including classification, regression, clustering and dimensionality reduction.

### **keras & tensorflow**

keras is a high-level, deep learning API developed by Google for implementing neural networks, and belongs to TensorFlow library. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple back-end neural network computation.

# Chapter 12

## Studies

This chapter gathers studies related to this project that do not directly form its main body, but are a fundamental part of its understanding. Specifically, some studies to decide what percentage of the total capital is risked in each trade are carried out.

Apart from fixing a particular trading strategy, one of the most relevant features when implementing the bot is knowing how much money is bet on each trade. Obviously, that bet trade percentage will affect the Return Of Investment (ROI), also according to the percentage of winning trades that the bot itself has.

Bearing the above in mind, some ideas to analyze the relationship between these variables arise. First of all, a code based on repetition and on random trade closures is simulated. After that, a mathematical development on that percentage is carried out, focusing on some variables of interest.

### 12.1 Simulation on Random Trade Closures

Before mathematically analyzing the described problem, a Python code is developed in order to get a first insight in the expected results. Before proceeding with its explanation, it is worth mentioning that this code is presented in Sec. B.1 Simulation on Random Trade Closures.

The code consists of a trading simulator in which that closes trades randomly, obviously always within a certain specified limits. First of all, it can be defined both the minimum and the maximum percentage of winning trades that the bot executes. Also, it is notorious to remark that the simulation can be carried out by deciding if commissions are taken into account or not and, if proceed, select the percentage of that commission. Moreover, number of trades to perform in each iteration of the simulation and also the number of iterations can be chosen. Lastly, the limits for both take profit (for winning trades) and stop-loss (for losing trades) are defined.

Therefore, the main core of that code consists of a *for* structure in which the simulation is carried out for the entire range of percentage of winning trades. At

that point, a combination without repetition<sup>1</sup> function is also coded, as shown in Sec. B.2 Combination Without Repetition. This combination is used for generating all the possible index winning combinations for the given number of trades. In order the reader to better understand the latter, a specific example with four trades and a percentage winning trades of 50% is given:

$[W, W, L, L]$   
 $[W, L, W, L]$   
 $[W, L, L, W]$   
 $[L, W, W, L]$   
 $[L, W, L, W]$   
 $[L, L, W, W]$

where  $W$  refers to won trades and  $L$  to lost trades. For the above example, there are six different combinations, which is in accordance with Eq. (10.2). With  $n = 4$  and  $k = 0.5n = 2$ , Eq. (10.2) anticipates these six possible combinations.

The fact of performing different iterations for each combination is because either the take profit or the stop-loss (depending on whether it is a winning or losing trade, respectively) is a random number less than a specific chosen value. For instance, random values up to +10% for take profit and up to -10% for stop-loss.

With all of the above, the trades are simulated for every percentage of winning trades, traversing the specified range for the bet trade percentage. From that, some numerical results are extracted and graphically displayed. In this case, it is of interest to represent the ROI as a function of the bet trade percentage, for the whole range of percentage of winning trades.

Having understood the purpose of the study and the operation of the code, the results of a case with specific values of interest are shown and analyzed.

The percentage of winning trades is fixed from 48 to 52%. No commissions are applied to the trades<sup>2</sup>. The maximum value for take profit and stop-loss is 10%. The number of trades is set to 100, and a total of 10 iterations are performed for each winning combination. With that, the results shown in Fig. 12.1 are obtained.

At first instance, it can be observed that percentages of winning trades below 50% clearly lead to a negative ROI, while percentages above 50% lead to a positive ROI. At the midpoint, 50% assumes mainly negative ROI values, according to the average obtained for each bet trade percentage. It can be said, therefore, that the 50% line marks winning or losing money, as long as commissions are not charged on the trades. Without a doubt, it would be expected that the fact of applying commissions would

---

<sup>1</sup>The reader not familiar with combination without repetition is recommended to read Sec. 10.2.2 Combination Without Repetition.

<sup>2</sup>As will be seen later, the idea is to trade Bitcoin on Binance, an exchange that does not charge commissions for trading with said cryptocurrency.

lower the percentage of winning trades that would make money (positive ROI). In other words, the higher the commissions, the higher percentage of winning trades required to maintain the same ROI.

Regardless of the percentage, it is also concluded that as the bet trade percentage increases, the greater the dispersion of the ROI obtained. From a mathematical sense, the above makes perfect sense, since the take profit and stop loss are generated randomly within each of the combinations without repetition to iterate.

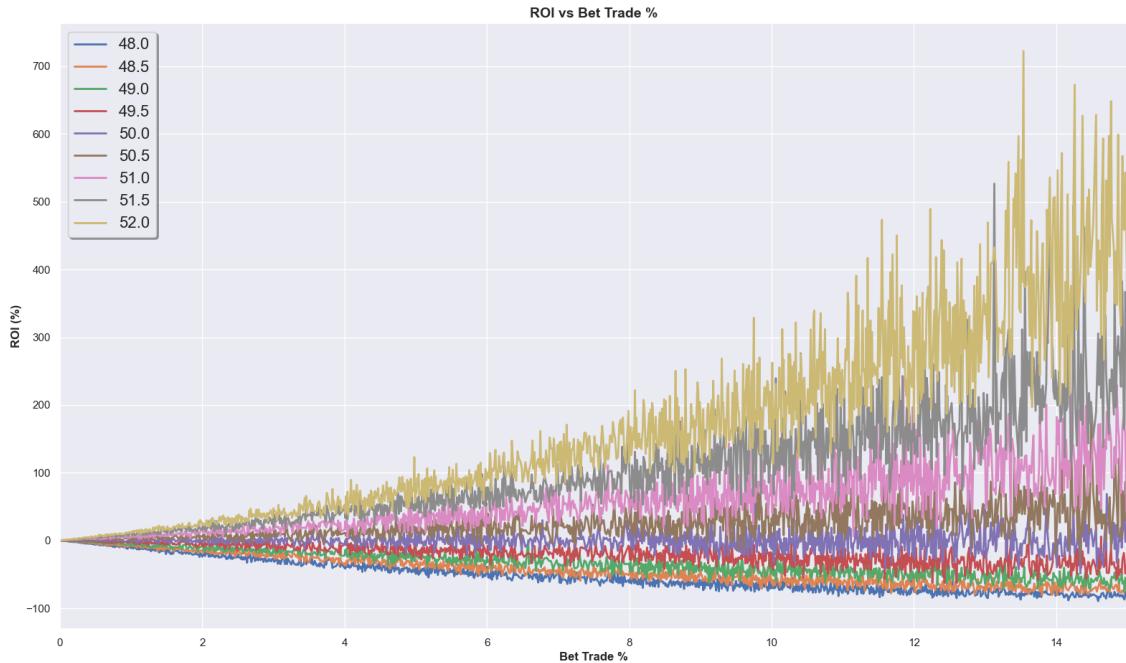


Figure 12.1: Simulation on random trade closures

Looking at Fig. 12.1, there is no doubt that as long as it is ensured that the bot operates with an effectiveness greater than 50%, or even 51% to be sure, the higher the bet trade percentage, the higher the ROI obtained. However, here one must take into account the fact that a losing streak can lead to the total bankruptcy of the estate. In view of this and the clear dispersion that is presented in Fig. 12.1, it is concluded that a good range in which to operate would be for a bet trade percentage of between 2% and 4%.

Furthermore, it is worth mentioning that the fact of randomizing the closing of the trades is not faithfully close to reality. Here, the trades would have to be closed according to a specific strategy. However, it is intended to carry out a generic study for the case presented.

Assessing all of the above, it was decided to mathematically analyze the above variables, to relate them to each other and to be able to draw more general conclusions.

## 12.2 Analysis of Strategy Variables

As commented, a mathematical relationship is found for the proposed variables. To do this, it is essential to make the definition of each variable clear:

- $b$ : bet trade percentage
- $t$ : take profit
- $s$ : stop-loss
- $p = b \cdot t$
- $g = b \cdot s$
- $A = \frac{g}{p} = \frac{s}{t}$  (A: stop-loss vs take profit factor)
- $n$ : number of trades
- $\omega$ : number of won trades
- $\omega_p = \frac{\omega}{n}$  ( $\omega_p$ : percentage of won trades)

With all these variables defined, the definition of winning and losing trades itself leads to:

$$1 + ROI = (1 + p)^\omega \cdot (1 - g)^{n-\omega} \quad (12.1)$$

In Eq. (12.1),  $(1 + p)^\omega$  represents the  $\omega$  won trades, while  $(1 - g)^{n-\omega}$  represents the  $n - \omega$  lost trades. To eliminate some variables, Eq. (12.1) can become:

$$1 + ROI = (1 + p)^\omega \cdot (1 - Ap)^{n(1-\omega_p)} \quad (12.2)$$

To eliminate the exponents, apply logarithms to each side of Eq. (12.2).

$$\log(1 + ROI) = n [\omega_p \cdot \log(1 + p) + (1 - \omega_p) \cdot \log(1 - Ap)] \quad (12.3)$$

To simplify Eq. (12.3), the terms are rearranged:

$$\frac{\log(1 + ROI)}{n} = \omega_p [\log(1 + p) - \log(1 - Ap)] + \log(1 - Ap) \quad (12.4)$$

And continuing with the simplification:

$$\omega_p = \frac{\frac{\log(1+ROI)}{n} - \log(1 - Ap)}{\log(1 + p) - \log(1 - Ap)} \quad (12.5)$$

Eq. (12.5) represents a relationship between five variables of interest: the percentage of winning trades  $\omega_p$ , the return of investment  $ROI$ , the number of trades  $n$ , the

stop-loss vs take profit factor  $A$ , and the variable  $p$ , which is no more than the bet trade percentage  $b$  multiplied by the take profit  $t$ .

As far as factor  $A$  is concerned, there is no doubt that it is a positive value. When it takes values less than one, take profit becomes greater than stop-loss. This does not only mean that take profit gives more money than the stop-loss removes, but also that it is more probable to close a trade because of the stop-loss, by clearly having less margin in which the price can fall. When  $A$  is exactly one, take profit and stop-loss values are equal. And in case of taking values greater than one, stop-loss becomes greater than take profit, and consequently, in this case it is more probable to close a trade because of the take profit.

As for variable  $p$ , it takes values from  $5 \cdot 10^{-4}$  to  $5 \cdot 10^{-3}$ . This is because it is assumed that  $b$  varies between 0.01 and 0.05, and that  $t$  comes from 0.05 to 0.1, all limits included.

Lastly,  $n$  is limited to 40,000 trades, assuming that the bot can operate with the daily number of trades it wishes, and therefore the time range of the study will depend on the frequency with which the bot operates, obviously according to the implemented strategy.

Once the variables are completely fixed, it is time to better analyze their relationship. It is decided to represent the ROI versus the number of trades by fixing two of the others three variables, and traversing a specific range for the other one. To achieve that, Matlab code is implemented, and the three plots in Fig. 12.2, Fig. 12.3 and Fig. 12.5 are obtained, from respectively the codes shown in Sec. B.3 ROI vs Number of Trades for  $\omega_p$  Range, Sec. B.4 ROI vs Number of Trades for  $p$  Range and Sec. B.5 ROI vs Number of Trades for  $A$  Range.

Firstly, both  $A$  and  $p$  are fixed.  $\omega_p$  is varied from 0.48 to 0.52, since a lower value than 0.48 would suppose that the strategy could be improved, and a higher value than 0.52 would almost ensure a positive ROI.

At that point, it is of vital relevance to assess that the latter is true as long as factor  $A$  takes values close to unity, since for values much greater than one, the benefit of each won trade would be much greater than the loss of a trade lost, and therefore more trades could be lost than won and still making solid money. And the same would happen the other way around, in case of having a factor  $A$  that is much smaller than unity, much more money would be lost with a stop-loss than the one that would be gained with a take profit, and consequently, money could be lost even having a  $\omega_p$  clearly higher than 50%.

Despite taking into account the above reasoning, the analysis is also carried out in the indicated  $\omega_p$  range, obtaining then the plot in Fig. 12.2.

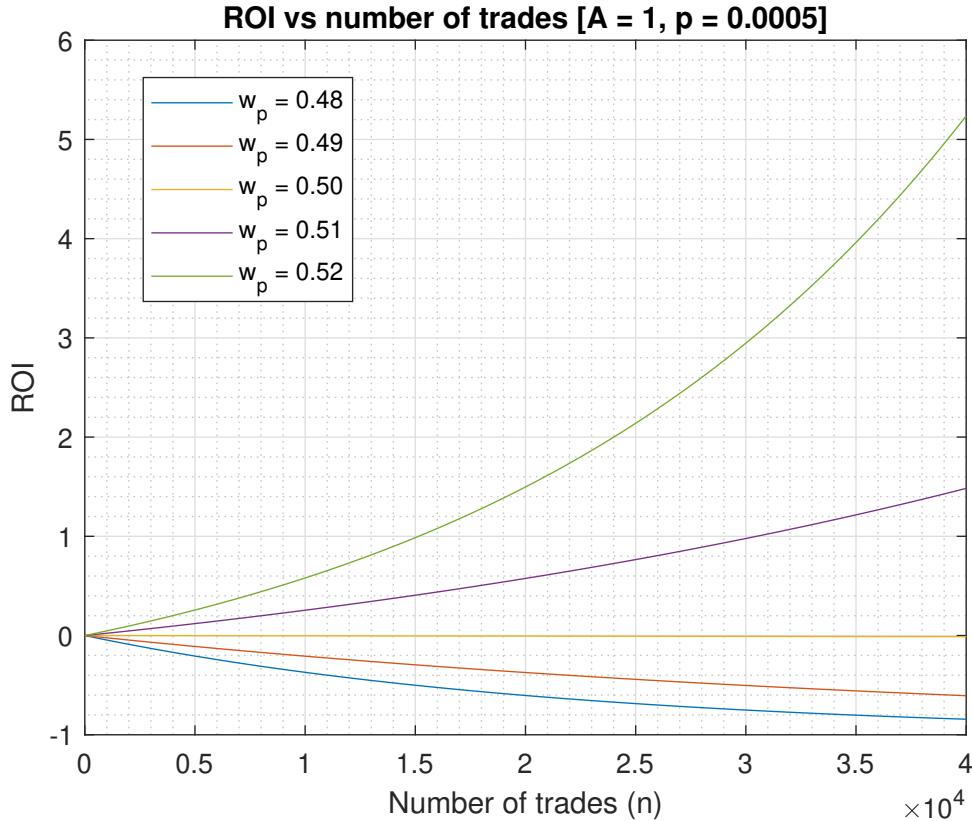


Figure 12.2: ROI vs number of trades for  $\omega_p$  range

With  $A$  factor set to one, the ROI becomes positive as long as  $\omega_p$  is above 50%, and turns negative when  $\omega_p$  is less than 50%. When that percentage is just that value, the ROI remains almost zero. Hence, it can be concluded that the greater  $\omega_p$ , the better the ROI is, always taking into account that the zero-ROI-line (where hardly any money is lost or earned) is affected by the  $A$  factor.

Now, it is intended to analyze what is the direct influence of variable  $p$ . It should be expected that the higher the  $p$  value, greater tendency exists to win or lose money. In other words, in a positive ROI tendency, a higher value of  $p$  is expected to lead to even a greater positive tendency, and the same with a negative ROI tendency. Indeed, the latter is displayed in Fig. 12.3. It is clearly observed that for a situation in which ROI has to be negative ( $A = 1$  and  $\omega_p = 0.498$ ), an increase in  $p$  leads to greater monetary losses.

Nevertheless, this does not happen for the entire ranges of  $\omega_p$ , as can be seen in Fig. 12.4, where not only the rule of operating with a higher  $p$  value leads to a greater tendency is broken, but also the tendency changes completely according to that value of  $p$ .

It is worth taking a look at the plots obtained. There are three which presents a positive ROI tendency, while one reverts to negative. The first three plots corresponds with  $p$  values of  $5 \cdot 10^{-4}$  (blue),  $1.5 \cdot 10^{-3}$  (orange) and  $2.5 \cdot 10^{-3}$  (yellow). The point

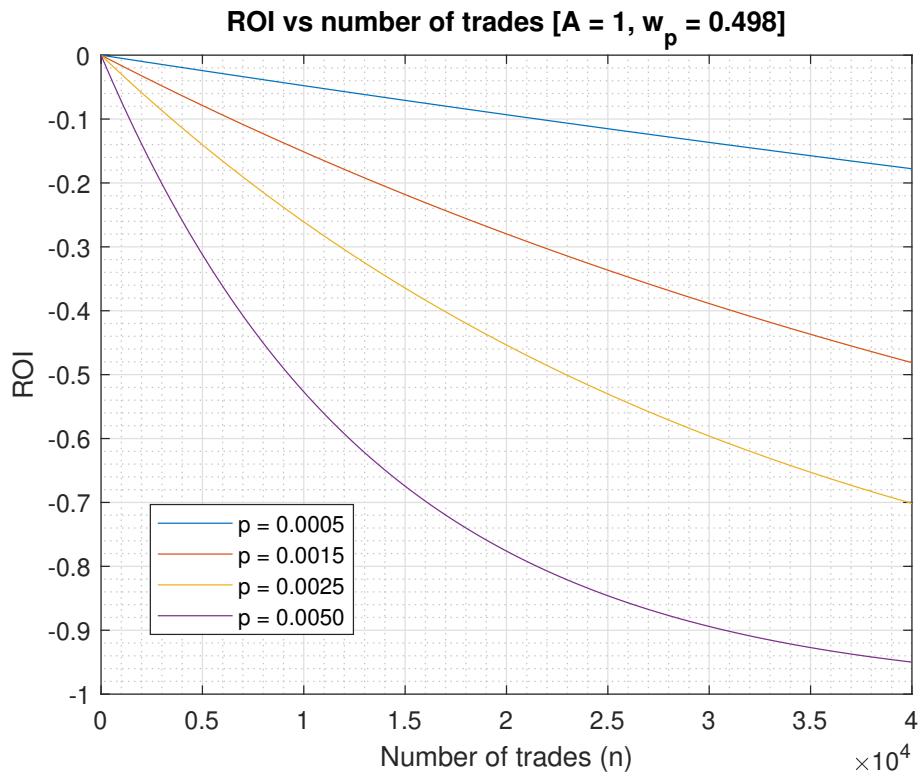


Figure 12.3: ROI vs number of trades for  $p$  range and  $\omega_p = 0.498$

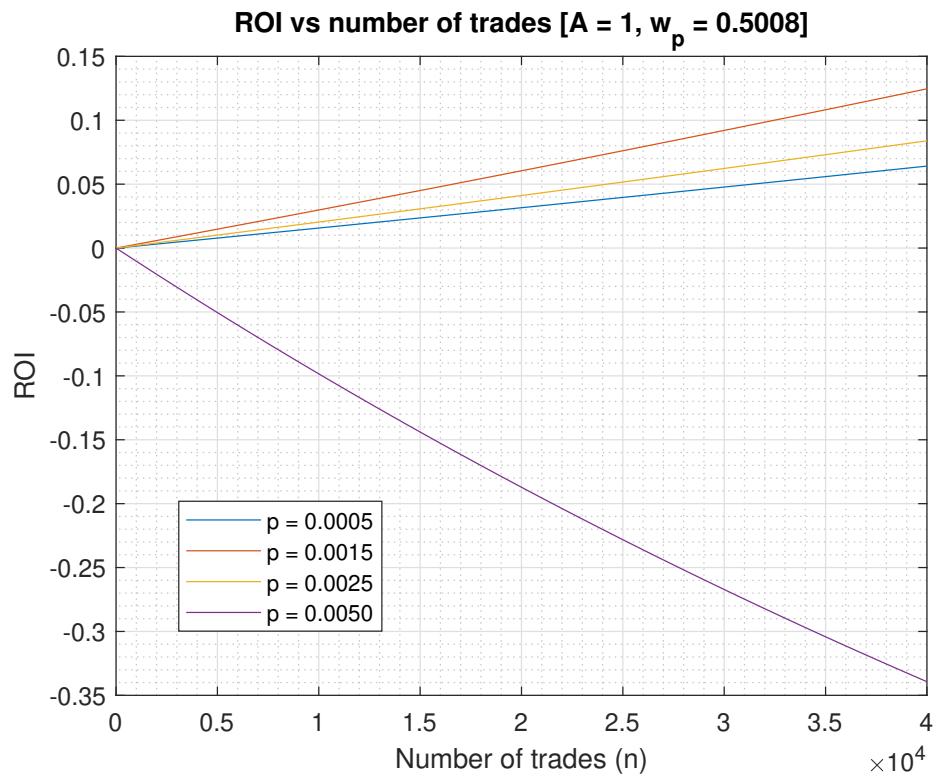


Figure 12.4: ROI vs number of trades for  $p$  range and  $\omega_p = 0.5008$

is that yellow and orange cases presents better results on ROI than blue case, but orange case is better than yellow one. This means that, with  $A = 1$  and the chosen  $\omega_p = 0.5008$ , more money is earned by following a strategy with a  $p$  value of  $1.5 \cdot 10^{-3}$  rather than a value of  $2.5 \cdot 10^{-3}$ .

The explanation behind this is based on the nature of Eq. (12.5). It consists of a fraction with different logarithms, some of whose arguments also contain more than one variable. Hence, it is in fact expected that some non-linear behaviour arises, at least in specific numerical ranges for any of the variables. Having observed the above, it is clear that said equation should be studied mathematically in depth, as it is commented in Chap. 17.

Lastly, it is intended to study the behavior of the ROI for different values of  $A$ , from 0.98 to 1.02, both included. With  $p = 5 \cdot 10^{-4}$  and  $\omega_p = 0.5$ , Fig. 12.5 shows that for values of  $A$  above 1, the ROI is expected to be negative, since the stop-loss is greater than the take profit, and consequently more money is lost than earned. On the contrary, for values of  $A$  below 1, with which take profit is greater than stop-loss value, a positive ROI is obtained.

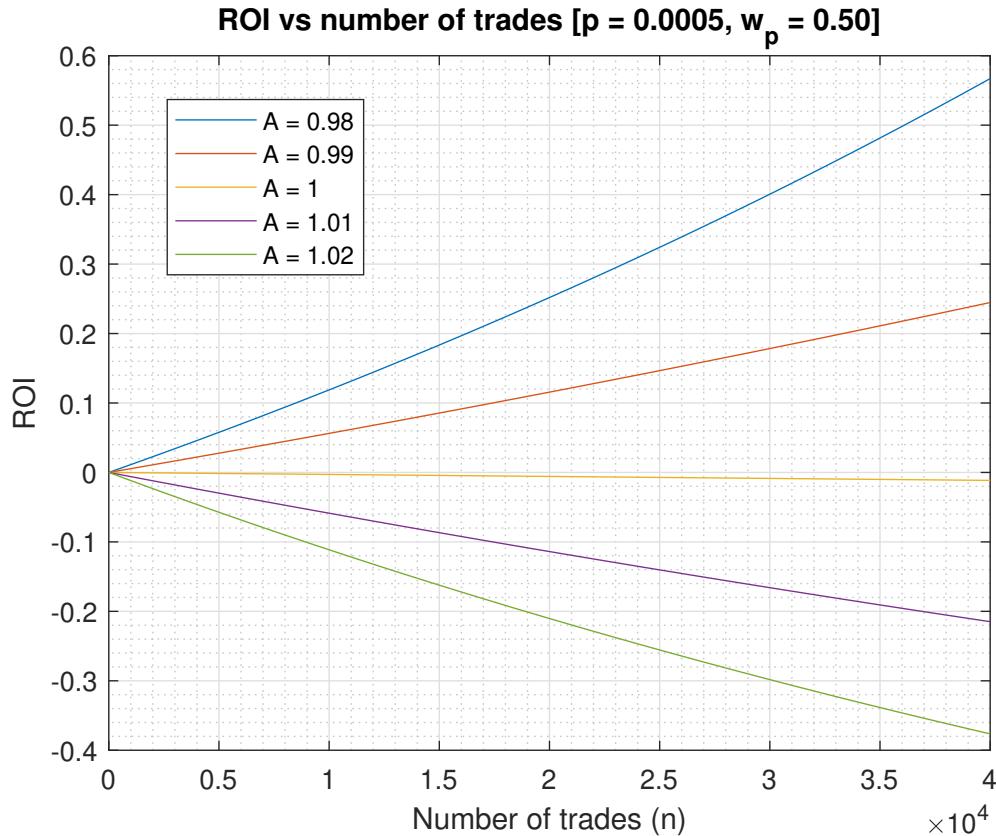


Figure 12.5: ROI vs number of trades for  $A$  range

In view of all the above, it is concluded that the most appropriate option is to perform that analysis with the specific trading strategy, which would help to fix some values and extract more particular conclusions.

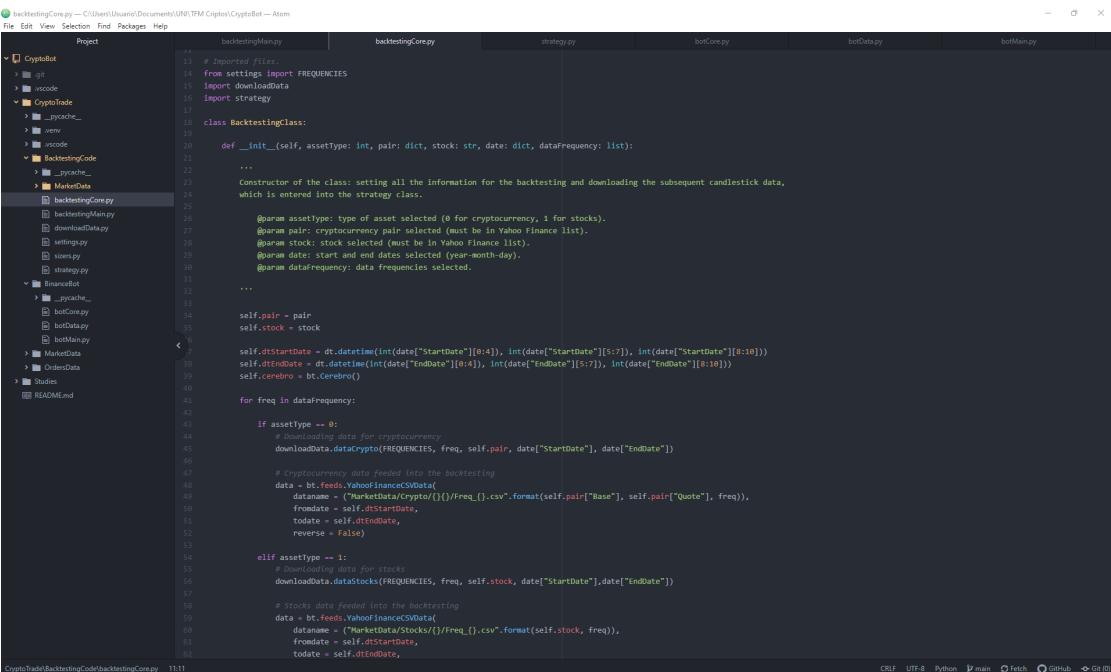
# Chapter 13

## Software Description

Software used for the development of the project is briefly described in this chapter. Specifically, the less common software is explained, and it is understood that programs such as Excel or Matlab are known to the reader.

### 13.1 Atom

Atom is a free and open-source text and source code editor for macOS, Linux, and Microsoft Windows (Fig. 13.1). It supports plug-ins written in JavaScript, and is embedded with GitHub. Atom enables users to install third-party packages to customize the features and looks of the editor. Moreover, Atom's default packages can apply syntax highlighting for multiple programming languages and file formats.



The screenshot shows the Atom text editor interface. The top menu bar includes File, Edit, View, Selection, Find, Packages, Help, and Atom. The left sidebar displays a project structure with several subfolders like CryptBot, CryptoTrade, and backtestingCode. The main editor area has five tabs open: backtestingCore.py, backtestingCore.py, strategy.py, testCore.py, and testMain.py. The code in the backtestingCore.py tab is as follows:

```
# backtestingCore.py --- C:\Users\Usuarin\Documents\UNI\TFM Cripto\CryptoTrade --- Atom
File Edit View Selection Find Packages Help
Project backtestingCore.py strategy.py testCore.py testMain.py
CryptBot
CryptoTrade
backtestingCode
MarketData
backtestingCore.py
backtestingMain.py
downloadData.py
settings.py
bt.py
strategy.py
btCoreBot
btCoreCryp
btCoreStock
btCoreCryp
btCoreStock
MarketData
Studies
README.md

# Imported files
from settings import FREQUENCIES
import downloadData
import strategy

class BacktestingClass:
    def __init__(self, assetType: int, pair: dict, stock: str, date: dict, dataFrequency: list):
        ...
        # Constructor of the class: setting all the information for the backtesting and downloading the subsequent candlestick data,
        # which is entered into the strategy class.

        #param assetType: type of asset selected (0 for cryptocurrency, 1 for stocks).
        #param pair: cryptocurrency pair selected (must be in Yahoo Finance list).
        #param stock: stock selected (must be in Yahoo Finance list).
        #param date: start and end dates selected (year-month-day).
        #param dataFrequency: data frequencies selected.

        self.pair = pair
        self.stock = stock
        self.date = date
        self.dataFrequency = dataFrequency

        self.dtStartDate = dt.datetime(int(date["StartDate"][:4]), int(date["StartDate"][5:7]), int(date["StartDate"][-2:10]))
        self.dtEndDate = dt.datetime(int(date["EndDate"][:4]), int(date["EndDate"][5:7]), int(date["EndDate"][-2:10]))
        self.cerebro = bt.Cerebro()

        for freq in dataFrequency:
            if assetType == 0:
                # Downloading data for cryptocurrency
                downloadData.downloadDataCrypto(FREQUENCIES, freq, self.pair, date["StartDate"], date["EndDate"])

            # Cryptocurrency data feeded into the backtesting
            data = bt.feeds.YahooFinanceCSVData(
                dataname = ('Marketdata/Crypto/{}//Freq_{}.csv'.format(self.pair["Base"], self.pair["Quote"], freq)),
                fromdate = self.dtStartDate,
                todate = self.dtEndDate,
                reverse = False)

        elif assetType == 1:
            # Downloading data for stocks
            downloadData.downloadDataStocks(FREQUENCIES, freq, self.stock, date["StartDate"], date["EndDate"])

            # Stocks data feeded into the backtesting
            data = bt.feeds.YahooFinanceCSVData(
                dataname = ('Marketdata/Stocks/{}//Freq_{}.csv'.format(self.stock, freq)),
                fromdate = self.dtStartDate,
                todate = self.dtEndDate,
```

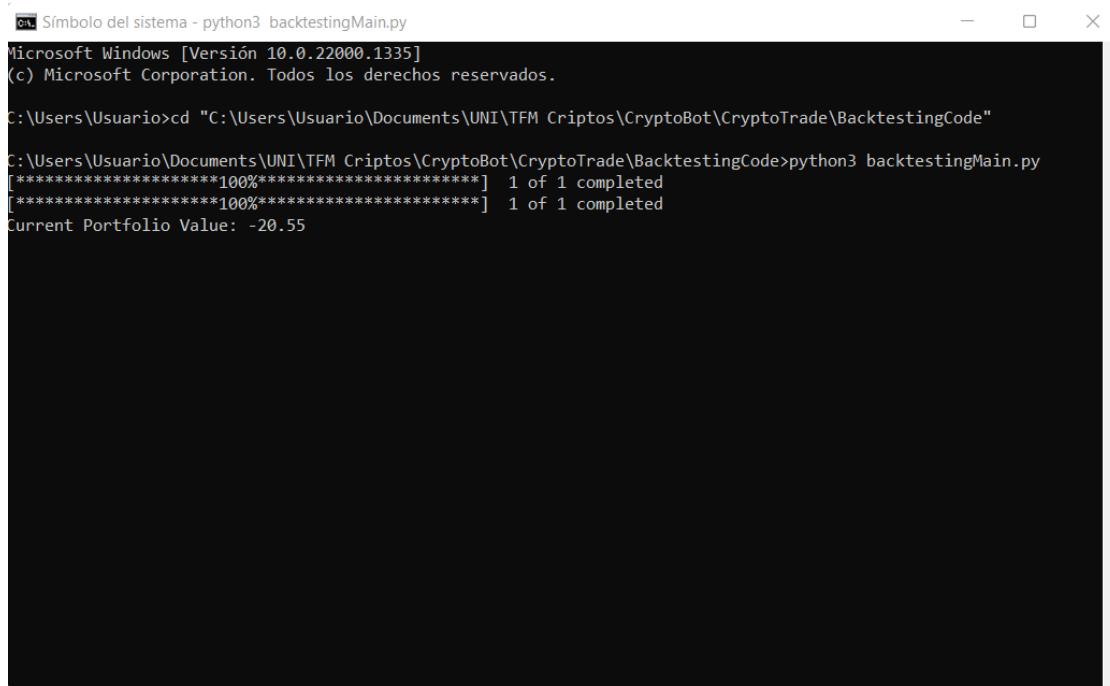
Figure 13.1: The bot has been programmed in Atom environment

## 13.2 GitHub Desktop

GitHub Desktop is an application that enables the interaction with GitHub using a Graphical User Interface (GUI) instead of the command line or a web browser. It allows completing most Git commands from the desktop with visual confirmation of changes. One can push to, pull from, and clone remote repositories, and uses collaborative tools such as attributing commits and creating pull requests.

## 13.3 Command Prompt

In Windows operating systems, the Command Prompt is a program that emulates the input field in a text-based user interface screen with the Windows GUI. It can be used to execute entered commands and perform advanced administrative functions. Despite not being a specific software, requires to be flagged for code development, as shown in Fig. 13.2.



The screenshot shows a Windows Command Prompt window titled "Símbolo del sistema - python3 backtestingMain.py". The window displays the following text:

```
Símbolo del sistema - python3 backtestingMain.py
Microsoft Windows [Versión 10.0.22000.1335]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Usuario>cd "C:\Users\Usuario\Documents\UNI\TFM Criptos\CryptoBot\CryptoTrade\BacktestingCode"

C:\Users\Usuario\Documents\UNI\TFM Criptos\CryptoBot\CryptoTrade\BacktestingCode>python3 backtestingMain.py
[******100%*****] 1 of 1 completed
[******100%*****] 1 of 1 completed
Current Portfolio Value: -20.55
```

Figure 13.2: Code has been run by using Command Prompt

# Chapter 14

## Software Implementation

The proposal of this chapter is to detail the development of the implementation of the algorithm proposed for the development of the trading bot. Firstly, its requirements are stated and justified. And then, a further description of its development is detailed. Please note that this chapter is not related to the description of neither the language programming nor the explanation of how using Atom.

### 14.1 Requirements of the Algorithm

Apart from the method design itself, an essential aspect to take into account for developing the algorithm is its requirements. The next list outlines them:

- The algorithm must be developed to trade through Binance exchange.
- It must use Yahoo! Finance data.
- It must be totally flexible. Variables must be easy to be changed. Different strategies could be used at the same time.
- It must ensure the backtesting of strategies.
- It must be delivered before January 12<sup>th</sup>, 2023.

The algorithm is thereby implemented according to the first four features, and taking into account the temporary imposition of the last requirement. At all times, it is a priority that the algorithm is properly integrated into Binance exchange, since it would not be useful from a practical point of view. Further, downloaded data must be obtained from Yahoo! Finance platform. The bot must allow the user to easily swap the value of the input variables, such as the bet percentage on each trade, the pair being traded, or the time frequency of the trade. Lastly, the backtesting of any developed strategy must be carried out before its own use in reality.

Studied the demanded requirements, the algorithm can already be implemented.

## 14.2 Algorithm

The implemented algorithm is clearly separated into two different parts: one for the trading bot itself; and another one for the backtesting. Moreover, each part has been implemented from the development of several files, which facilitate both the programming and the understanding of the algorithm.

Next, the description of the implementation of said files is presented, taking into account that the reader has perfectly understood what was detailed in the previous chapters. Before that, it is also notorious to remark that the code that is taught in this chapter is not complete, since only the essential content for its explanation and understanding is shown. The entire code is presented in Appendix A.

### 14.2.1 Bot

This part of the algorithm consists of the trading bot itself. Specifically, three different files are implemented: *botData*, *botCore*, and *botMain*. The relationship that these files have within the algorithm is the one observed in Fig. 14.1.

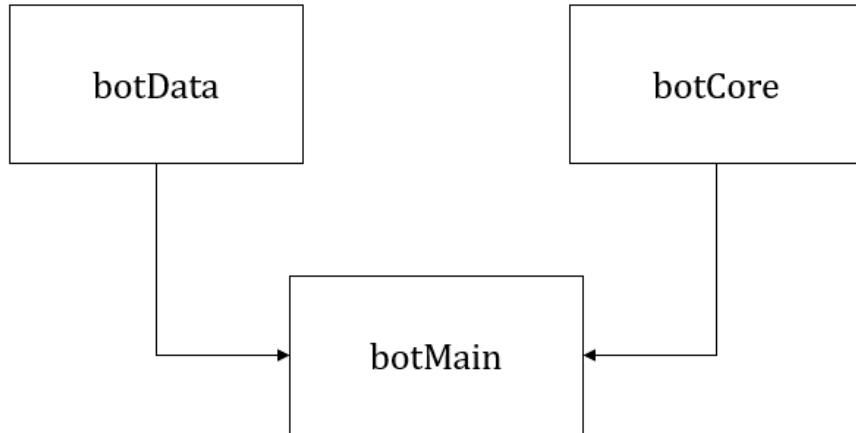


Figure 14.1: Diagram of the trading bot

Both *botData* and *botCore* files are implemented in order to ease the development of the *botMain* file. On the one hand, *botData* sets all the necessary data, which is chosen by the user, to make the bot run. On the other hand, *botCore* consists of a class in which functions to properly implement the bot are defined. These functions include from the data downloading from Binance and its treatment, to the creation of orders to execute the trading.

All the above functions are imported into the *botMain* file, in which the trading bot itself is implemented. The bot is thus properly connected with Binance, and some forward market orders has been created so as to verify the functionality<sup>1</sup>. However,

---

<sup>1</sup>More comments on that are presented in Sec. 14.2.1.3 *botMain*.

and as Chap. 2 progressed, the bot has not yet worked with any strategy in reality. In other words, it has not yet really faced the financial market of cryptocurrencies, since no real trade has been completed yet.

Therefore, it can be said that the bot has everything ready to work, despite the fact that it has not done so in reality. This topic is covered in depth in Chap. 17, where comments on the future work of this project are presented.

Having introduced the general structure of that part of the algorithm, it is time to analyze each file, in order to better understand the algorithm itself.

#### 14.2.1.1 botData

This class aims at setting the data to run the bot. The user should specify their public and secret keys of the API created in their own Binance account. Furthermore, the cryptocurrency pair being traded must also be selected. Also, start and end dates of the period during which it is intended to trade, and the frequency of the candlesticks to be traded.

Code Listing 14.1: The user can choose the settings to run the bot

```
# This class sets the data to run the bot.

# Python libraries.
import datetime as dt
from datetime import datetime

# Binance public and private keys.
BINANCE = {
    "APIKey": "XXXXXXX",
    "SecretKey": "XXXXXXX"
}

# Cryptocurrency pair.
COIN = [
    "Crypto": "BTC",
    "Fiat": "USDT"
]

# Start and end dates (year, month, day, hours, minutes, seconds).
DATE = [
    "StartDate": "2022-05-01 10:00:00",
    "EndDate": "2022-07-15 00:00:00"
    #datetime.now()
]

# Selected frequency.
FREQUENCY = "15m"
```

Please note that both the real API and secret keys are not displayed in Code Listing 14.1 (XXXXXXX is shown) because of privacy reasons, as the author does not intend to share his passwords with anyone else.

### 14.2.1.2 botCore

This file consists of a class called *Bot\_BinanceClass*, which at turn contains different functions that are used to implement the bot. Despite the fact that are presented in separated code listings, all the functions are within the aforementioned class.

Firstly, the constructor of the class<sup>2</sup> is displayed. It assigns and stores all the information necessary to trade, such as public and private keys, cryptocurrency pair data, frequency to operate with, price data, commission, or balance.

Code Listing 14.2: Constructor of the class in *botCore*

```
# Class with Binance data and functions.
class Bot_BinanceClass:

    def __init__(self, APIKey, secretKey, crypto, fiat, frequency,
                 dataElements):

        self.APIKey = APIKey
        self.secretKey = SecretKey

        self.crypto = crypto
        self.fiat = fiat

        self.frequency = frequency
        self.dataElements = dataElements

        self.apiBinance = Client(self.APIKey, self.secretKey)

        # Variables on purchase and sale operations.
        self.BUY = SIDE_BUY
        self.SELL = SIDE_SELL

        self.pairInfo = self.apiBinance.get_symbol_info(self.crypto
                                                       +self.fiat)

        self.cryptoDecimals = int(self.pairInfo["baseAssetPrecision"
                                              ""])
        self.fiatDecimals = int(self.pairInfo["quotePrecision"])

        self.filtersData = self.pairInfo["filters"]

        self.PRICE_FILTER = self.filtersData[0]
        self.LOT_SIZE = self.filtersData[2]
        self.MIN_NOTIONAL = self.filtersData[3]

        self.minPrice = float(self.PRICE_FILTER["minPrice"])
        self.maxPrice = float(self.PRICE_FILTER["maxPrice"])
        self.tickSize = float(self.PRICE_FILTER["tickSize"])

        self.minQty = float(self.LOT_SIZE["minQty"])
        self.maxQty = float(self.LOT_SIZE["maxQty"])
```

---

<sup>2</sup>A constructor is a special method of a class or structure in object-oriented programming that initializes a newly created object of that type. Whenever an object is created, the constructor is called automatically [53].

```

        self.stepSize = float(self.LOT_SIZE["stepSize"])

        self.minNotional = float(self.MIN_NOTIONAL["minNotional"])

        self.pairFees = self.apiBinance.get_trade_fee(symbol = self.crypto+self.fiat)[0]

        self.makerCommission = float(self.pairFees["makerCommission"])
        self.takerCommission = float(self.pairFees["takerCommission"])

        self.cryptoBalance = self.apBinance.get_asset_balance(self.crypto)[ "free"]
        self.fiatBalance = self.apiBinance.get_asset_balance(self.fiat)[ "free"]
    
```

The function *dataFunction* is used for filtering the required data. Pandas library is used to transform the input data into a data frame, which is properly distributed into the necessary columns: time, price related items, volume and trades. Then, this data frame is transformed into date reference time, to finally being the output of the function itself.

Code Listing 14.3: The data downloaded from Binance is filtered

```

def dataFilter(Data):
    df = pd.DataFrame(Data)
    df = df.drop([6, 7, 9, 10, 11], axis=1)

    columns = ["time", "open", "high", "low", "close", "volume",
               "trades"]
    df.columns = columns

    for i in columns:
        df[i] = df[i].astype(float)

    df["start"] = pd.to_datetime(df["time"]*1000000)

    return df
    
```

The function *updateData* aims at updating the data of the last candle available. To achieve that, the information of the new candle is got from Binance. Then, the data of the first candle stored (in index 0) is removed, and the rest are moved one index down, so that the new candle can be stored in the last position.

Code Listing 14.4: The data of the last candle registered by Binance is updated

```

def updateData(self):
    newCandle = self.apiBinance.get_klines(symbol=self.crypto+
                                             self.fiat, interval=self.
                                             frequency, limit=1)
    # limit = 1 -> the data of the last candle is downloaded
    # interval -> interval of that candle (frequency)
    self.df.drop(index = 0, inplace = True) # The first candle
                                             is removed and the rest
    
```

```
        are moved 1 index down
    self.df = self.df.append(self.dataFilter(newCandle),
                           ignore_index=True)
    self.df.index = list(range(self.DataElements))
```

The function *updateAccountBalance* is used for updating the crypto and fiat balances. The process is simple, since every time both balances are obtained from Binance.

Code Listing 14.5: Crypto and fiat balances are updated

```
def updateAccountBalance(self):
    self.cryptoBalance = self.apiBinance.get_asset_balance(self.
                                                             .crypto)[“free”]
    self.fiatBalance = self.apiBinance.get_asset_balance(self.
                                                             fiat)[“free”]
```

The function *marketOrder* aims at creating a market order to be executed. A try-except structure is used. The function always tries to create the order, which needs to know in which direction of the market is intended to trade (buy or sell) and the quantity to be risked. In case of not being possible of creating the order, a Binance API exception is printed.

Code Listing 14.6: Creation of a generic market order

```
def marketOrder(self, operationSide, quantity):
    try:
        self.OrderName = self.apiBinance.create_order(
            symbol = self.Crypto + self.Fiat,
            side = operationSide,
            type = “MARKET”,
            qty = quantity
        )

    except BinanceAPIException as e:
        print(e)
```

The function *limitOrder* works similar to the function *marketOrder* in terms of structure, but in that case time and price limit values are also demanded to create the order.

Code Listing 14.7: Creation of a limit market order

```
def limitOrder(self, operationSide, quantity, price):
    try:
        self.orderName = self.apiBinance.create_order(
            symbol = self.crypto + self.fiat,
            side = operationSide,
            type = “LIMIT”,
            timeInForce = TIME_IN_FORCE_GTC,
            qty = “{:.{}f}”.format(quantity, self.
                                  cryptoDecimals),
            price = “{:.{}f}”.format(price, self.cryptoDecimals
                                    )
        )
```

```

except BinanceAPIException as e:
    print(e)

```

The function *notifyOrder* is used for notifying orders that has been executed, taking into account that an order could be completed with more than one purchase. For instance, when a market order is created, the bot will try to buy at the lower possible price, and because of price and time delays, it is possible that all quantity may not be purchased at the same price. For that reason, a *for* structure which considers all the different possible purchases is needed. Here, the commissions are also calculated. Lastly, and by means of the function *orderRegister*, the data of the order is registered.

Code Listing 14.8: An order that hast just been filled is notified

```

def notifyOrder(self):
    totalPrice = 0
    totalQuantity = 0
    averagePrice = 0
    totalComissions = 0

    for i in self.newOrder["fills"]:
        # It is necessary to
        # calculate the mean of the
        # order
        totalComissions += float(i["commission"])
        totalQuantity += float(i["qty"])
        totalPrice += float(i["price"])*float(i["qty"])

    averagePrice = totalPrice/totalQuantity

    # Registering the order data
    self.orderRegister(self.newOrder["type"], self.newOrder["side"],
                       totalComissions,
                       self.newOrder["commissionAsset"],
                       averagePrice,
                       totalQuantity, totalPrice)

```

The function *orderRegister* aims at stating constancy of all the orders executed, both purchases and sales. All of them are written into a text file, following a specific structure for the data. It is relevant to note that in order to be able to write into the file, it firstly should be opened, and closed once the order has been registered.

Code Listing 14.9: All the purchase and sale orders are registered

```

def orderRegister(self, orderType, side, comission, coinComission,
                  avgPrice, qtyCrypto, qtyFiat):
    f1 = open("OrdersData/Registro_{}_{BOT.txt".format(self.
                                                       crypto+self.fiat, self.
                                                       frequency), "a+")

    if os.stat("OrdersData/Registro_{}_{BOT.txt".format(self.
                                                       crypto+self.fiat, self.
                                                       frequency)).st_size == 0:

```

The function `getCandleData` is used for reading the downloaded candlestick data from the `.csv` file.

Code Listing 14.10: Reading the downloaded candlestick data

```
def getCandleData(self):
    self.candleData = pd.read_csv("MarketData/{}{}/Freq_{}.csv"
                                  .format(self.crypto, self.fiat, self.frequency))
    self.dataCandles = len(self.candleData)
```

All the functions displayed in Code Listing 14.11 consists of the calculation of specific indicators that can be later used in the trading strategy implementation. Note that the numerical calculations are obtained from *talib* library and stored in the candlestick data.

Code Listing 14.11: Calculation of some different indicators

```

# Calculation of the SMA Indicator
def getSMA(self, days):
    self.candleData["SMA_{}".format(days)] = ta.SMA(self.candleData
                                                    ["close"], days)

# Calculation of the EMA Indicator
def getEMA(self, days):
    self.candleData["EMA_{}".format(days)] = ta.EMA(self.candleData
                                                    ["close"], days)

# Calculation of Stochastic Indicator
def getStochastic(self, days, pfast, pslow):
    STO = ta.stochastic(self.candleData, period = days, pfast =
                           pfast, pslow = pslow)
    self.candleData["STO_k_{}_{}_{}".format(days, pfast, pslow)] =
        STO.k
    self.candleData["STO_d_{}_{}_{}".format(days, pfast, pslow)] =
        STO.d

```

### 14.2.1.3 botMain

The trading bot itself is implemented in this file. As has been commented, the bot is ready to work in reality, but has not done it yet. It is clearly that this is one of the next imminent steps to work on, as it is discussed in Chap. 17.

Anyway, the bot is connected with Binance and is able to operate, since it downloads data price, updates it, and creates orders to start the trading. Obviously, these orders have to be automatically created from the based on a trading strategy previously validated in backtesting.

Having clarified the above, it is time to understand the code thoroughly. Firstly, the correspondent libraries, options and data are imported. Binance client library is imported to get and store both API and secret keys. Also, *botData* and *botCore* files are imported, and data related to available frequencies to trade with in Binance is generated.

Code Listing 14.12: Importing libraries and data

```
# Binance libraries.
from binance.client import Client

# Imported files.
sys.path.insert(0, os.path.abspath('..../..'))
from BinanceKeys import BINANCE_KEYS
from botData import BINANCE, COIN, DATE, FREQUENCY
import botCore

# API data.
apiBinance = Client(BINANCE["APIKey"], BINANCE["SecretKey"])

# Available frequency in Binance.
frequencyAvailable = ["1m", "3m", "5m", "15m", "30m", "1h", "2h", "4h",
                      "6h", "8h", "12h", "1d", "3d",
                      "1w", "1M"]
```

The function *downloadCandlesData* aims at downloading the candlestick data from Binance. According to the selected time range in which to trade (start and end dates from *botData*), the total time range in seconds is obtained. Then, as selected frequency to trade with gives the time each candle lasts (for that frequency), the total number of candles is calculated as the division of the total time by the time of each candle. At that point, it is remarkable to note that the time duration of the candle is obtained by an *if-elif* structure in which all the frequency cases are considered, and for each one the correspondent time is calculated.

Before returning the number of candles to download, the subsequent data is already downloaded from Binance, taking the frequency as the interval and this number of candles as limit to stop downloading. This data is filtered and stored in a *csv* file.

Code Listing 14.13: Downloading candlestick data from Binance

```
def downloadCandlesData(Coin, Fiat, Frequency, StartDate, EndDate):
```

```
# Calculating the total number of candles to download according
# to the selected time range
FI = dt.datetime(int(StartDate[0:4]), int(StartDate[5:7]), int(
    StartDate[8:10]), int(
    StartDate[11:13]),
    int(StartDate[14:16]), int(StartDate[17:19]))
FF = dt.datetime(int(EndDate[0:4]), int(EndDate[5:7]), int(
    EndDate[8:10]), int(EndDate[
    11:13]),
    int(EndDate[14:16]), int(EndDate[17:19]))

# EndDate
Total_Time = math.floor((FF - FI).total_seconds())

if "m" in Frequency:
    if(len(Frequency)) == 2:
        CandleSeconds = 60*float(Frequency[0])
        CandleMinutes = int(Frequency[0])
    else:
        CandleSeconds = 60*float(Frequency[:2])
        CandleMinutes = int(Frequency[:2])
elif "h" in Frequency:
    if (len(Frequency)) == 2:
        CandleSeconds = 3600*float(Frequency[0])
        CandleMinutes = 60*int(Frequency[0])
    else:
        CandleSeconds = 3600*float(Frequency[:2])
        CandleMinutes = 60*int(Frequency[:2])
elif "d" in Frequency:
    CandleSeconds = 86400*float(Frequency[0])
    CandleMinutes = 24*60*int(Frequency[0])
elif "w" in Frequency:
    CandleSeconds = 7*86400*float(Frequency[0])
    CandleMinutes = 7*24*60*int(Frequency[0])
elif "M" in Frequency:
    CandleSeconds = 30*86400*float(Frequency[0])
    CandleMinutes = 30*24*60*int(Frequency[0])

DataElements = math.floor(Total_Time/CandleSeconds)

Header = ["time", "open", "high", "low", "close", "volume", "
trades", "start"]
Data = apiBinance.get_klines(symbol = Coin+Fiat, interval =
Frequency, limit =
DataElements)
Data = initialDataFilter(Data)

if not os.path.isdir("MarketData/{}/{}".format(Coin, Fiat)):
    os.mkdir("MarketData/{}/{}".format(Coin, Fiat))

if os.path.exists("MarketData/{}/{}Freq_{}.csv".format(Coin,
Fiat, Frequency)):
    os.remove("MarketData/{}/{}Freq_{}.csv".format(Coin, Fiat,
Frequency))

Data.to_csv("MarketData/{}/{}Freq_{}.csv".format(Coin, Fiat,
```

```

        Frequency))

return DataElements

```

Code Listing 14.14 consists of the code lines which finally implements the bot. First of all, it is checked if the selected frequency to trade with is available in Binance, and if so, the data is downloaded, filtered and stored, according to all the parameters defined by the user in *botData*. In case of the frequency not being available, an error message is printed and the bot does nothing. Secondly, the bot variable is created from the class of *botCore* file, by simply calling it.

At that point, the strategy would have to be called and from there start trading from the creation of orders that appear thanks to the intelligence developed for the strategy.

Code Listing 14.14: Trading bot implementation

```

if checkFrequency(frequencyAvailable, FREQUENCY):
    DataElements = downloadCandlesData(COIN["Crypto"], COIN["Fiat"]
                                         , FREQUENCY, DATE["StartDate"]
                                         , DATE["EndDate"])

else:
    print("Error for the selected frequency {}".format(
          FREQUENCY))

Bot = botCore.Bot_BinanceClass(BINANCE["APIKey"], BINANCE[
                               "SecretKey"], COIN["Crypto"], COIN
                               ["Fiat"], FREQUENCY, DataElements)

```

## 14.2.2 Backtesting

This part of the algorithm aims at validating the implemented trading strategies before using them in reality by the bot algorithm. Hence, it is a necessary prior step to verify that the strategy to follow is really going to generate a positive ROI. As progressed before, this validation is carried out by using cryptocurrency data from Yahoo! Finance, since this platform allows downloading data for a lot of coins and for their entire time range.

Here, six different files are developed: *settings*, *sizers*, *downloadData*, *strategy*, *backtestingCore*, *backtestingMain*. Their relationship within the algorithm is shown in Fig. 14.2.

The *backtestingMain* file collects the functions of all the other files to end up implementing the backtesting. Both *settings* and *sizers* files aims at setting specific data and variables. Later, the *downloadData* file is used to download the correspondent market information data from Yahoo. The trading strategy to be backtested is defined in the *strategy* file. And lastly, the *backtestingCore* aims at defining functions to be used for the backtesting. It is relevant to remark that this file also calls three other files: *settings*, *strategy* and *downloadData*, as represented in Fig. 14.2.

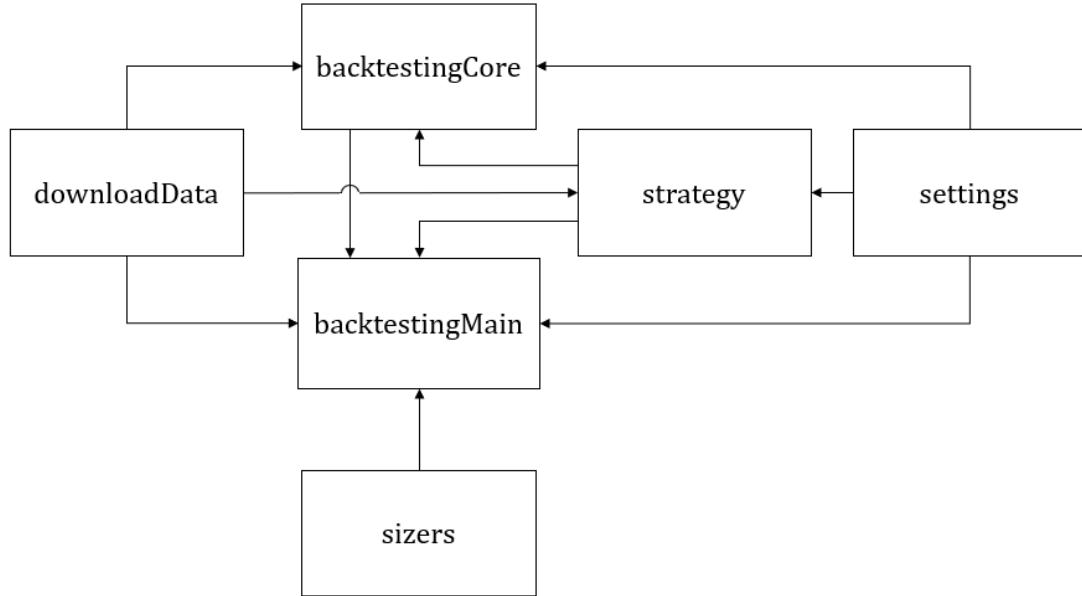


Figure 14.2: Diagram of the backtesting

Having introduced the general structure of that part of the algorithm, it is time to analyze each file, in order to better understand the algorithm itself.

#### 14.2.2.1 settings

This class aims at setting the data for the backtesting. The user must specify the cryptocurrency pair (present on the Yahoo! Finance list), the start and end dates and the frequencies which with the backtesting is intended to be performed.

Please note that this trading bot is intended to not only trade cryptocurrencies, but also stocks in the future. For this reason, a stock (present on the Yahoo! Finance list) must also be indicated and the type of asset to be tested: 0 for cryptocurrencies, 1 for stocks.

Code Listing 14.15: The user can choose the settings to run the backtesting

```
# Python libraries
import datetime as dt
from datetime import datetime

# Cryptocurrency pair (must be in Yahoo Finance list).
PAIR = {
    "Base": "BTC",
    "Quote": "USD"
}

# Stocks (must be in Yahoo Finance list).
STOCK = {
    "Ticker": "TSLA"
}
```

```

# Defining asset type: 0 for cryptocurrency, 1 for stocks.
ASSET = {
    "AssetType": 0
}

# Start and end dates (year-month-day) for the backtesting (
#           datetime.now() for current date).
DATE = {
    "StartDate": "2022-01-01",
    "EndDate": "2022-07-15"
}

# Frequencies available for the candlestick data (must be in Yahoo
#           Finance list).
FREQUENCIES = ["1m", "2m", "5m", "15m", "30m", "60m", "90m", "1h",
               "1d", "5d", "1wk", "1mo", "3mo"]

```

#### 14.2.2.2 sizers

The sizer of the bet trade percentage is set in this class, by importing the percent sizer from the backtrader library. Although this is a simple setting, it is separated into a specific code file because of the possibility of adding new sizers in the future.

Code Listing 14.16: Sizers for the backtesting are defined

```

# Python libraries
from backtrader.sizers import PercentSizer

class FullMoney(PercentSizer):

    params = (
        ("percents", 3),
    )

```

From the analysis developed in Chap. 12, it has been decided to use a bet trade percentage of 3%, since with a take profit/stop-loss of a 5%, it is in a range for  $p$  without expected high dispersion and with a value that seems to maximize the return for a strategy that wins a bit more than half the total trades, which is the aim to reach.

#### 14.2.2.3 downloadData

This file defined the functions for extraction the datasets containing the information of the market (candles) from Yahoo! Finance. Specifically, two different functions are presented<sup>3</sup>. The first one checks if the frequency selected is on Yahoo, and the second one downloads the requested data.

The function *checkFrequency* initializes a boolean named *check* being false. The list of frequency available in Yahoo is traversed by means of a *for* instruction. Only in

---

<sup>3</sup>In fact, three different functions are coded in the file, but the third one is the same as the one for downloading data for crypto, but for stocks. The entire code is presented in Appendix A.

case that the selected frequency is found in that list, the boolean *check* becomes true. After that, in case of that boolean is still false, a warning message is printed to the user, and the available frequencies are displayed. The function returns the value of the boolean *check*.

Code Listing 14.17: Checking if the selected frequency is on Yahoo

```
def checkFrequency(FrequencyAva: list, Frequency: str):

    check = False
    for i in range(len(FrequencyAva)):
        if Frequency == FrequencyAva[i]:
            check = True

    if check == False:
        print("The selected frequency of {} is not available.
              Select one of the
              following frequencies: \n".format(Frequency))
        for i in range(len(FrequencyAva)):
            print("{} \t".format(FrequencyAva[i]))
    exit()

    return check
```

As commented, the second function is for downloading the candlestick data into .csv files. Firstly, note that the download is performed only if the selected frequency exists (if not, again an error message appears). Obviously, this checking is carried out by means of the *checkFrequency* function detailed above.

Code Listing 14.18: Downloading candlestick data

```
def dataCrypto(FrequencyAva: list, Frequency: str, Pair: tuple,
               StartDate: str, EndDate: str):

    # Downloading in case the frequency exists
    if checkFrequency(FrequencyAva, Frequency):

        # Downloading data from Yahoo Finance
        Data = yf.download('{}-{}'.format(Pair["Base"], Pair["Quote"]),
                           start = StartDate,
                           end = EndDate,
                           index_as_date = False,
                           interval = Frequency)

        Data['time'] = Data.index
        Data['time'] = Data['time'].dt.strftime('%Y-%m-%d %H:%M:%S')

        # Generating the directory of the data
        if not os.path.isdir("MarketData/Crypto/{}{}".format(Pair[
                               "Base"], Pair["Quote"])):
            os.mkdir("MarketData/Crypto/{}{}".format(Pair["Base"],
                                                     Pair["Quote"]))

        # Deleting the name in case it already exists
```

```

if os.path.exists("MarketData/Crypto/{}{}/Freq_{}.csv".
                    format(Pair["Base"], Pair
                           ["Quote"], Frequency)):
    os.remove("MarketData/Crypto/{}{}/Freq_{}.csv".format(
                    Pair["Base"], Pair[""
                           Quote"], Frequency))

# Generating .csv file from DataFrame
Data.to_csv("MarketData/Crypto/{}{}/Freq_{}.csv".format(
                    Pair["Base"], Pair["Quote"]
                           ], Frequency))

else:
    print("Error for the selected frequency {}".format(
                    Frequency))

```

#### 14.2.2.4 strategy

This file aims at creating a model on which the trading operations are based. Specifically, a Recurrent Neural Network through an ARIMA model<sup>4</sup> is computed. Hence, it is essential to note that here only the operation of the algorithm is described, and that no reference is made to the parameters required by the strategy to carry out the backtesting. All this is developed in Chap. 15, where the results obtained by applying the entire backtesting algorithm are also discussed.

The file basically consists of a class called *Strategy*, whose constructor develops the aforementioned neural network.

Firstly, the data used for training the neural network is obtained by means of *downloadData* file. This data is preprocessed from *numpy* and *keras* libraries. Specifically, the data is converted into a training set, which is adequately reshaped according to its own length, to later be scaled using the *MinMaxScaler()* function. Lastly, the set is divided into a dependent and a independent variable to follow the implementation of an ARIMA model.

With that, the RNN is initialized, and the subsequent layers are added. This is done by using *Sequential()* class, which is a framework for creating neural network models. That model is fitted with the data sets from the ARIMA model. This means that the model itself learns how to operate from the provided historical data.

Once the model is trained, a test model is created so as to be able to backtest the effectiveness of the neural network. Obviously, the dates with which the model is tested cannot coincide with the period during the model has been trained. More comments on that are detailed in Chap. 15.

Code Listing 14.19: Constructor of the *strategy* file

```

class Strategy(bt.Strategy):

    def __init__(self):

```

<sup>4</sup>Please read Chap. 10 carefully, where it is detailed what a RNN is and the ARIMA model.

```
# Downloading training data
self.trainingData = downloadData.dataCrypto(FREQUENCIES,
                                             dataFrequency, PAIR, DATE
                                             ["StartDate"], DATE[""
                                               EndDate"])

# Data preprocessing
self.trainingSet = self.trainingData.values
self.trainingSet = np.reshape(self.trainingSet, (len(self.
                                                 trainingSet), 1))
self.trainingSet = MinMaxScaler().fit_transform(self.
                                                 trainingSet)
self.lengthTS = len(self.trainingSet)
self.trainingX = self.trainingSet[0:self.lengthTS - 1]
self.trainingX = np.reshape(self.trainingX, (len(self.
                                                 trainingX), 1, 1))
self.trainingY = self.trainingSet[1:self.lengthTS]

# Initializing the RNN
self.RNN = Sequential()

# Output layer is added
self.RNN.add(Dense(units = 1))

# Input layer and LSTM layer are added
self.RNN.add(LSTM(units = 4, activation = 'sigmoid',
                  input_shape = (None, 1)))

# Compiling the RNN
self.RNN.compile(optimizer = 'adam', loss =
                  mean_squared_error)

# Fitting the RNN to the training data set
self.RNN.fit(self.trainingX, self.trainingY, batch_size = 5
             , epochs = 100)

# Creating a test model
self.testingData = downloadData.dataCrypto(FREQUENCIES,
                                             dataFrequency, PAIR, DATE
                                             ["StartDate"], DATE[""
                                               EndDate"])
self.testSet = self.testingData.values
self.input = np.reshape(self.testSet, len(self.testSet), 1)
self.input = MinMaxScaler().transform(self.input)
self.input = np.reshape(self.input, (len(self.input), 1, 1))
self.testModel = self.RNN.predict(self.input)
self.testModel = MinMaxScaler().inverse_transform(self.
                                                 testModel)
```

#### 14.2.2.5 backtestingCore

This file consists of a class called *BacktestingClass*, which at turn contains different functions that are used to develop the backtesting. Despite the fact that are presented

in separated code listings, all the functions are within the aforementioned class.

Firstly, the constructor of the class is displayed. It sets all the information for the backtesting and downloads the subsequent candlestick data, which at turn is entered into the strategy class.

The most relevant feature is the import of the *Cerebro* class<sup>5</sup>, which allows gathering all the data, the strategy and the analyzers while ensuring a correct functioning at any moment. Furthermore, it allows the backtesting to be executed, returning the results and giving access to the plotting facilities.

It is also outstanding to note that this constructor is designed to operate with both cryptocurrencies and stocks, despite the fact that this project is focused exclusively on the first type of asset.

Code Listing 14.20: Constructor of the *BacktestingClass*

```
class BacktestingClass:

    def __init__(self, assetType: int, pair: dict, stock: str, date
                 : dict, dataFrequency: list):

        self.pair = pair
        self.stock = stock

        self.dtStartDate = dt.datetime(int(date["StartDate"][:4]),
                                       int(date["StartDate"][5:7]),
                                       int(date["StartDate"][:8:10]))
        self.dtEndDate = dt.datetime(int(date["EndDate"][:4]), int(
            date["EndDate"][5:7]),
            int(date["EndDate"][:8:10]))
        self.cerebro = bt.Cerebro()

        for freq in dataFrequency:

            if assetType == 0:
                # Downloading data for cryptocurrency
                downloadData.dataCrypto(FREQUENCIES, freq, self.
                                         pair, date[""
                                         StartDate"], date
                                         ["EndDate"])

                # Cryptocurrency data feeded into the backtesting
                data = bt.feeds.YahooFinanceCSVData(
                    dataname = ("MarketData/Crypto/{}{}/Freq_{}.csv"
                               .format(self
                                       .pair["Base"],
                                       self.pair[""
                                       Quote"], freq
                                       )),

                    fromdate = self.dtStartDate,
```

<sup>5</sup>The *Cerebro* class is an open class which belongs to the *backtrader* library. Please read Sec. 11.4 Trading Libraries to know more about this library.

```
        todate = self.dtEndDate,
        reverse = False)

    elif assetType == 1:
        # Downloading data for stocks
        downloadData.dataStocks(FREQUENCIES, freq, self.
            stock, date[""
            "StartDate"], date["
            "EndDate"])

        # Stocks data feeded into the backtesting
        data = bt.feeds.YahooFinanceCSVData(
            dataname = ("MarketData/Stocks/{}/Freq_{}.csv".
                format(self.
                    stock, freq))
            ,
            fromdate = self.dtStartDate,
            todate = self.dtEndDate,
            reverse = False)

        self.cerebro.adddata(data)
```

The function *setInitialMoney* is used for setting the initial cash with which is intended to trade.

Code Listing 14.21: Setting initial cash of the portfolio

```
def setInitialMoney(self, money):

    self.cerebro.broker.setcash(money)
    self.InitialMoney = money
```

The function *printCurrentMoney* displays on screen which is the portfolio value at the requested time.

Code Listing 14.22: Printing the current portfolio value

```
def printCurrentMoney(self):

    print("Current Portfolio Value: %.2f" % self.cerebro.broker
        .getvalue())
```

The function *setCommissions* aims at setting the commission applied in each trade of the specific cryptocurrency which is intended to trade with on Binance.

Code Listing 14.23: Setting Binance commissions

```
def setComissions(self, comissions):

    self.cerebro.broker.setcommission(commission = comissions)
```

The function *setSizers* is used for setting the correspondent sizers for trading. In this case, the bet percentage for each trade is the only sizer which is set.

Code Listing 14.24: Setting sizers

```
def setSizers(self, sizer):
    self.cerebro.addsizer(sizer)
```

The function *setStrategy* simply sets the strategy to be backtesting, adding it to *cerebro*.

Code Listing 14.25: Setting backtesting strategy

```
def setStrategy(self, strategy):
    self.cerebro.addstrategy(strategy)
```

The function *runStrategy* runs the implemented strategy by means of *cerebro*. Additionally, the results of that strategy are analyzed by the *talib* library.

Code Listing 14.26: Running the implemented strategy

```
def runStrategy(self):
    self.strats = self.cerebro.run()
    self.strat = self.strats[0]
    self.results = self.strat.analyzers.ta.get_analysis()
```

The function *plotBacktestingResults* is used for plotting backtesting results. Specifically, it displays the market analyzers and backtesting plots by calling the *printMarketAnalyzers*, which will be explained later, and the plot functionality of *cerebro*.

Code Listing 14.27: Plotting backtesting results

```
def plotBacktestingResults(self):
    self.printMarketAnalyzers()
    self.cerebro.plot()
```

The function *setBotAnalyzers* sets the bot analyzers. Hence, the family of *Analyzer* objects aims at provide an analysis of what has happened or even of what is actually happening with the implemented trading system, both in terms of profit and risk. To deepen the understanding and use of these analyzers, the reading of [3] is highly recommended.

Code Listing 14.28: Setting bot analyzers

```
def setBotAnalyzers(self):
    self.cerebro.addanalyzer(bt.analyzers.TradeAnalyzer, _name=
                           "ta")
```

The function *printMarketAnalyzers* is used for displaying the market analyzers plots. If there are no trades, this function warns the user.

Code Listing 14.29: Printing market analyzers plot

```
def printMarketAnalyzers(self):
```

```
results = self.strat.analyzers.ta.get_analysis()

if results.total.total==0:
    print("There have been no trades for the current
          backtesting .")
    return

else:
    closedTrades = results.total.closed
    wonTrades = results.won.total
    winRatio = wonTrades/closedTrades
    netProfit = results.pnl.net.total
    totalReturn = (netProfit/self.InitialMoney)*100

    totalDays = (self.dtEndDate - self.dtStartDate).days
    monthlyReturn = totalReturn*30.0/totalDays

    fig, ax = plt.subplots(1,1)

    column = ["Results"]

    rows = ["Closed trades",
            "Won trades",
            "Win ratio",
            "Net profit",
            "Total return",
            "Start period",
            "End period",
            "Monthly return"]

    results = [[ "{}".format(closedTrades),
                "{}".format(wonTrades),
                "{} %".format(100*winRatio),
                "{} {}".format(netProfit, self.pair[
                    "Quote"]),
                "{} %".format(totalReturn),
                "{}".format(self.dtStartDate),
                "{}".format(self.dtEndDate),
                "{} %".format(monthlyReturn)]]

    df = pd.DataFrame(results, columns = column)
    ax.axis('tight')
    ax.axis('off')
    ax.table = plt.table(cellText = df.values,
                         rowLabels = rows,
                         colLabels = column,
                         colWidths = None,
                         loc = "center")
    ax.table.set_fontsize(20)
    ax.table.scale(1.8, 1.8)

    for (row, col), cell in ax.table.get_celld().items():
        if (row == 0) or (col == -1):
```

```

        cell.set_text_props(fontproperties=
                             FontProperties(
                                 weight='bold
                             )))

    ax.table.auto_set_column_width(col=list(range(len(df.
columns))))
    plt.title("Backtesting Results", fontsize = 30,
              fontweight="bold")

```

#### 14.2.2.6 backtestingMain

This file is formed by calling functions of all the others files, causing the backtesting to be carried out. Firstly, the backtesting strategy variable is created from calling the class of the *backtestingCore* file, which needs some data from *settings* and *strategy* as input variables. With that, several functions from the *backtestingCore* file are properly called. The initial money is arbitrarily set to \$1,000, and the bet trade percentage from *sizers* is loaded to the backtesting data information. Then, the analyzers to trade and the strategy are also set. And lastly, the strategy is run to be backtested, and the subsequent results are displayed.

It is also notorious that although there is a line which sets the commissions that apply when trading, it is indeed commented. This is why the strategy to be backtested trades with Bitcoin, cryptocurrency whose trading is commission free on Binance. When trading with commissions, this line would have to be uncommented and would simply subtract the corresponding commissions from the total capital. In this case, since there are no commissions, it is decided to eliminate the line, since despite the fact that the code would work the same and would not take into account any commission (applying a 0), thus the algorithm does not have to directly go through a line that does nothing for this specific case.

Code Listing 14.30: Strategy is backtested

```

backStrategy = backtestingCore.BacktestingClass(ASSET ["AssetType"],
                                              PAIR, STOCK ["Ticker"], DATE,
                                              strat.dataFrequency)
#Bot_BackTest.setComissions(Bot.makerCommission)
backStrategy.setInitialMoney(1000.0)
backStrategy.setSizers(sizers.FullMoney)
backStrategy.setBotAnalyzers()
backStrategy.setStrategy(strat.Strategy)
backStrategy.runStrategy()
backStrategy.printCurrentMoney()
backStrategy.plotBacktestingResults()

```

As mentioned, backtesting results are obtained for the implemented strategy. This is undoubtedly the next step, which is performed in Chap. 15.



# Chapter 15

## Results

This chapter aims to state constancy of the results obtained for the backtesting of the implemented strategy, which serves to verify that the bot can actually generate money. Specifically, numerical values of the parameters that shape the neural network of the trading strategy are discussed, in addition to the results when said strategy is backtested.

Before commenting on all of the above, it is relevant to note that compliance with all algorithm requirements is achieved. These requirements are detailed in Sec. 14.1 Requirements of the Algorithm, and are justified throughout the explanation of the algorithm in Sec. 14.2 Algorithm.

As previously explained in Sec. 14.2.2.4 strategy, the trading strategy consists of a RNN through an ARIMA model, which has been trained to trade Bitcoin. This training has been carried out based on BTC data from September 17, 2014 to December 31, 2022, both included<sup>1</sup>. The training data has been limited to 2020 to have data with which to backtest the model, since it is obvious that the backtesting data must be different from the training data, otherwise the bot would not be encountering a real trading situation, but rather it would face already known operations.

Hence, in order the backtesting to be performed, data from January 1, 2021 to November 30, 2022 is selected. Furthermore, some other relevant parameters should also be defined. As concluded from the studies developed in Chap. 12, a good combination of parameters would be to set the bet trade percentage to 3% and the take profit and stop-loss values to 5%. Obviously, as Bitcoin is the cryptocurrency to be traded, the pair is BTC/USDT and the asset type<sup>2</sup> is cryptocurrency.

As far as the frequency of the candlesticks is concerned, in this case it is convenient to select a frequency of 15 minutes, since it is intended to make dozens of trades a day, so fast intraday trading is required. For this reason, it is also decided to train the model and test it for frequencies of 5 and 30 minutes.

---

<sup>1</sup>BTC data is available in Yahoo! Finance from September 17, 2014

<sup>2</sup>Remember that the algorithm is designed to also operate in stocks in the future, so asset type is also necessary to be fixed.

With all of the above, the backtesting is run, obtaining the results shown below. At that point, it should be remembered that no commissions are applied in Binance when trading with BTC, and also that backtesting parameters are explained in Sec. 9.2.1 Backtesting.

## 15.1 Backtesting for a Frequency of 15 Minutes

The implemented trading strategy is firstly backtested for a frequency range of 15 minutes. The results with that frequency are then shown in Table (15.1).

Table 15.1: Backtesting results for a frequency of 15 minutes

Metric	Value
Annualized Net Return	7.3675%
Profitability	50.3621%
Average Take Profit/Stop-Loss	1.2397
Maximum Drawdown	1.1838%
Annualized Volatility	0.5876%
Annualized Volatility Negative Returns	0.7664%
Sharpe Ratio	12.5383
Sortino Ratio	9.6131

From Table (15.1), it is undoubtedly that the implemented trading strategy is really sound and ready to be executed in reality with a candlestick frequency of 15 minutes. Firstly, the net return is above 7%, due to a profitability slightly higher than 50% for the total of 7,236 trades during the two-year period. This also implies operating with a positive average return in percentage per trade, since the total take profit is higher than the total stop-loss.

As for the risk metrics, despite having a considerable higher maximum drawdown (in comparison with the net return) the analyzed volatilities are low enough to ensure that the risk offered is medium. In line with that, the return-risk ratios are clearly above 2, which indicates that the expected return clearly compensates for the risk taken.

## 15.2 Backtesting for a Frequency of 5 Minutes

Table (15.2) presents the obtained results for the backtesting for a frequency of 5 minutes.

Despite the fact that all the metrics worsen compared to operating with a frequency of 15 minutes, it is clear that operating for a frequency of 5 minutes would not be a mistake, as consistent profits are also generated.

In fact, the return is quite similar, only 0.14% lower. However, it should be considered that the decrease of the profitability from 50.3621% to 50.2144% would have implied

a lower return than that 7.2223% obtained. The point here is that by decreasing the frequency range, the number of total trades has increased to 13,146 during the two years, which obviously allows obtaining a higher return, as more winning trades are closed.

In view of the above, it is mandatory to comment on commissions when trading, since it is obvious that if trading with commissions, the strategy would be clearly diminished when operating with a frequency of 5 minutes, because the number of trades made is clearly greater. Undoubtedly, this must be considered when designing a trading strategy, and without a doubt, a way to reduce commissions should always be tried to find, since commissions is return that is lost directly.

Table 15.2: Backtesting results for a frequency of 5 minutes

Metric	Value
Annualized Net Return	7.2223%
Profitability	50.2144%
Average Take Profit/Stop-Loss	1.1387
Maximum Drawdown	1.3704%
Annualized Volatility	0.6854%
Annualized Volatility Negative Returns	0.8559%
Sharpe Ratio	10.5373
Sortino Ratio	8.4372

The analysis of the metrics can be extrapolated to that carried out for the frequency of 15 minutes, since the values obtained are very similar. In a generic way, it can be concluded that the return metrics are once again acceptable and the return-risk ratio is still high.

### 15.3 Backtesting for a Frequency of 30 Minutes

Lastly, the results for a frequency of 30 minutes are displayed in Table (15.3).

Table 15.3: Backtesting results for a frequency of 30 minutes

Metric	Value
Annualized Net Return	2.6817%
Profitability	50.2389%
Average Take Profit/Stop-Loss	1.0533
Maximum Drawdown	2.5566%
Annualized Volatility	0.9276%
Annualized Volatility Negative Returns	1.0157%
Sharpe Ratio	2.8910
Sortino Ratio	2.6403

Even though the profitability when operating with a frequency of 30 minutes is almost the same than that for the frequency of 5 minutes, again the total number of trades takes a vital role for the return of the strategy. Now, 4,378 trades has been performed during the two-year period, which involves in closing less winning trades and, consequently, in reducing benefits.

Apart from that, the maximum drawdown has increased considerably, becoming almost as large as the expected return, a fact that certainly weakens the strategy. Moreover, and in accordance with the above, the return-risk ratios have decreased to just above 2.

It cannot be said that trading with this 30 minute candles is not a winning strategy, but it is clearly worse than trading with 5 or 15 minute candles. In addition, it would be necessary to see if the fact of operating with commissions would mean ending up losing money with this frequency.

## 15.4 Strategy Analysis

The trading strategy implemented from the RNN has passed the backtesting, and it clearly works better when operation for a frequency of 15 minutes, despite the fact that a similar profitability is obtained for all the candlestick frequencies studied. Hence, the strategy offers a clearly positive return, with not a high risk, and above all, with a high return-risk ratio. Without a doubt, the model is ready to operate in reality.

However, perhaps it would be convenient to evaluate it with more data (from different years), but because the cryptocurrency market is so recent (data from around a decade is only available), there is not enough data available to train the model and be able to test it over a long period of time.

At that point, the idea of operating in the stock market reappears<sup>3</sup>, since a quantity of data from decades would be available, which would allow both training the model for a longer period of time, and also making a more reliable backtesting.

---

<sup>3</sup>It is recalled that the bot is designed to be able to operate in the stock market, and it would only be necessary to connect to a platform that operates with stocks, since Binance does not.

# Chapter 16

## Conclusions

In this project, a cryptocurrency trading bot has been developed. Its algorithm is divided into two differentiated parts: one for the bot itself, and the other for the backtesting of trading strategies.

The part for the bot itself is only coded, and the work here consisted of verifying that the connectivity with Binance is correct and the algorithm is capable of executing automatic trades. Hence, all the functions to run the strategy have been developed and their proper functioning verified.

As far as the backtesting part is concerned, its subsequent algorithm is also developed, obtaining the cryptocurrency data from Yahoo! and ensuring that any strategy could be backtested by simply calling its specific strategy file.

For this project, one trading strategy has been implemented and backtested, achieving results with a consistent positive return of 7.36% and an acceptable risk to operate. This means that this strategy can be coupled to the algorithm of the bot and forms the basis for trading on Binance.

A fundamental fact to keep in mind when trading is that the number of trades and, therefore, the frequency of the candlesticks totally affect the return obtained, obviously in addition to the profitability and possible commissions. It has been possible to see that for a strategy without commissions, for similar profitabilities, as the number of trades increases, the return increases. This agrees with the results obtained in the studies of Chap. 12.

Thus, it makes sense that the number of trades is greater for frequencies of minor candles. Here, therefore, it must be assessed that the fact of changing the frequency also affects profitability and, consequently, depending on the strategy followed, it will not always be better to operate with lower frequencies, not only because of the return, but also for the risk.

It is also outstanding to note that the backtesting part has been designed in different files to be able to backtest different strategies with some ease, simply by changing settings and sizers parameters. In this sense, the bot algorithm would also allow

trading with more than one strategy at the same time by simply creating another class of bot with a different strategy.

With all of the above, this project can be considered an absolute success, since the entire base of an algorithm for a trading bot has been developed, guaranteeing compliance with all the initial requirements. The results achieved are really promising, and this leads to the expectation of generating money automatically once the bot operates in reality.

Finally, these last lines are intended to record that this Master's Degree Thesis is much more than a college work. It has meant getting fully into what could be a large-scale engineering project, since not only has the world of cryptocurrencies been deepened, but it has also gone through the technical analysis of the market, the powerful field of Machine Learning, and object-oriented programming through Python. Undoubtedly, specializing in these fields provides the author with a multitude of new tools with great practical-technical application.

# Chapter 17

## Future Work

Despite the fact that good results have been achieved, this project needs to continue to be developed. This chapter is then related to expose what should be done in the near future, basing on everything stated in the previous chapters.

After verifying through backtesting that the implemented strategy achieves a good percentage of winning trades and that, therefore, everything seems to indicate that the bot can generate money, the first imminent step to develop is to make the bot trade in reality through Binance. To do this, the strategy file should be simply called within the main of the bot, implementing the relevant buy and sell orders.

Without a doubt, this will be a critical step in determining the success of the bot, not only because of the effectiveness that the strategy in question can offer, but also because of making sure that the bot operates as expected. Any glitches, no matter how minuscule, will be revealed when the trades are executed in reality.

At this point, it will be necessary to carry out an analysis of the bot's operation, with the aim of improving it. For instance, optimizing its response time to a possible trade, or even modifying its algorithmic structure for better performance.

Once there is no doubt that the bot works optimally, another possible improvement would be the implementation of new trading strategies. For this, it would be necessary to deepen both Machine Learning and time series analysis techniques, which allow a more severe bot training. In turn, the fundamentals in technical analysis could be of great help to finish setting a more robust strategy in terms of reliability.

Parallel to the above, it is essential to place more emphasis on the study of the bet trade percentage. It would be necessary to analyze the proposed variables in greater depth from a mathematical point of view, taking into account that each specific strategy can work in certain ranges for the relevant variables.

Despite the fact that not paying commissions when trading with Bitcoin is really advantageous, it would be convenient to check the effectiveness of the strategies for other cryptocurrencies, since perhaps a higher return can be achieved.

In this sense, the development of the bot could also be extended to the stock market,

also taking advantage of the fact that the algorithm has already been programmed with this intention. Here, however, the bot would have to be connected to some stock platform. Clearly, it would be ideal to separate this part of connecting with the exchange, so as not to overlap lines of code and simplify the development of the algorithm.

Lastly, and as a more long-term step, it could be considered using a personal cloud, in which the large amount of data generated could be stored, and all files would remain protected.

# Appendix A

## Bot Code

In this appendix, it is presented the full developed code for the trading bot algorithm explained in Chap. 14, both the bot and backtesting part.

### A.1 botData

Code Listing A.1: *botData* file code

```
# This class sets the data to run the bot.

# Python libraries.
import datetime as dt
from datetime import datetime

# Binance public and private keys.
BINANCE = {
    "APIKey": "XXXXXXX",
    "SecretKey": "XXXXXXX"
}

# Cryptocurrency pair.
COIN = {
    "Crypto": "BTC",
    "Fiat": "USDT"
}

# Start and end dates (year, month, day, hours, minutes, seconds).
DATE = {
    "StartDate": "2022-05-01 10:00:00",
    "EndDate": "2022-07-15 00:00:00"
    #datetime.now()
}

# Selected frequency.
FREQUENCY = "15m"
```

## A.2 botCore

Code Listing A.2: *botCore* file code

```
# TRADING BOT CLASS FILE: functions defined to implement the bot.

# Python libraries.
import pandas as pd
import numpy as np
import datetime as dt
import math
import os
import talib as ta

# Binance libraries.
from binance.client import Client
from binance.exceptions import BinanceAPIException
from binance.enums import *

# Class with Binance data and functions.
class Bot_BinanceClass:

    def __init__(self, APIKey, secretKey, crypto, fiat, frequency,
                 dataElements):

        self.APIKey = APIKey
        self.secretKey = SecretKey

        self.crypto = crypto
        self.fiat = fiat

        self.frequency = frequency
        self.dataElements = dataElements

        self.apiBinance = Client(self.APIKey, self.secretKey)

    # Variables on purchase and sale operations.
    self.BUY = SIDE_BUY
    self.SELL = SIDE_SELL

    self.pairInfo = self.apiBinance.get_symbol_info(self.crypto
                                                   +self.fiat)

    self.cryptoDecimals = int(self.pairInfo["baseAssetPrecision"]
                           ""])
    self.fiatDecimals = int(self.pairInfo["quotePrecision"])

    self.filtersData = self.pairInfo["filters"]

    self.PRICE_FILTER = self.filtersData[0]
    self.LOT_SIZE = self.filtersData[2]
    self.MIN_NOTIONAL = self.filtersData[3]

    self.minPrice = float(self.PRICE_FILTER["minPrice"])
    self.maxPrice = float(self.PRICE_FILTER["maxPrice"])
    self.tickSize = float(self.PRICE_FILTER["tickSize"])
```

```
self.minQty = float(self.LOT_SIZE["minQty"])
self.maxQty = float(self.LOT_SIZE["maxQty"])
self.stepSize = float(self.LOT_SIZE["stepSize"])

self.minNotional = float(self.MIN_NOTIONAL["minNotional"])

self.pairFees = self.apiBinance.get_trade_fee(symbol = self.crypto+self.fiat)[0]

self.makerCommission = float(self.pairFees["makerCommission"])
self.takerCommission = float(self.pairFees["takerCommission"])

self.cryptoBalance = self.apBinance.get_asset_balance(self.crypto)["free"]
self.fiatBalance = self.apBinance.get_asset_balance(self.fiat)["free"]

# Filtering of data downloaded from Binance
def dataFilter(Data):
    df = pd.DataFrame(Data)
    df = df.drop([6, 7, 9, 10, 11], axis=1)

    columns = ["time", "open", "high", "low", "close", "volume",
               "trades"]
    df.columns = columns

    for i in columns:
        df[i] = df[i].astype(float)

    df["start"] = pd.to_datetime(df["time"]*1000000)

    return df

# Updating the data of the last candle registered by Binance
def updateData(self):
    newCandle = self.apiBinance.get_klines(symbol=self.crypto+
                                             self.fiat, interval=self.frequency,
                                             limit=1)
    # limit = 1 -> the data of the last candle is downloaded
    # interval -> interval of that candle (frequency)
    self.df.drop(index = 0, inplace = True) # The first candle
                                            # is removed and the rest
                                            # are moved 1 index down
    self.df = self.df.append(self.dataFilter(newCandle),
                            ignore_index=True)
    self.df.index = list(range(self.DataElements))

# Updating the data on the number of coins of the selected asset
def updateAccountBalance(self):
```

```
self.cryptoBalance = self.apiBinance.get_asset_balance(self.crypto)["free"]
self.fiatBalance = self.apiBinance.get_asset_balance(self.fiat)["free"]

# Creating a generic market order (usually Market)
def marketOrder(self, operationSide, quantity):
    try:
        self.OrderName = self.apiBinance.create_order(
            symbol = self.Crypto + self.Fiat,
            side = operationSide,
            type = "MARKET",
            qty = quantity
    )

    except BinanceAPIException as e:
        print(e)

# Creating a market LIMIT order
def limitOrder(self, operationSide, quantity, price):
    try:
        self.orderName = self.apiBinance.create_order(
            symbol = self.crypto + self.fiat,
            side = operationSide,
            type = "LIMIT",
            timeInForce = TIME_IN_FORCE_GTC,
            qty = "{:.{}f}".format(quantity, self.cryptoDecimals),
            price = "{:.{}f}".format(price, self.cryptoDecimals)
    )

    except BinanceAPIException as e:
        print(e)

# Creating a test order to verify that it works
def testOrder(self, operationSide, quantity):
    try:
        self.newOrder = self.apiBinance.create_test_order(
            symbol = self.crypto + self.fiat,
            side = operationSide,
            type = "MARKET",
            qty = "{:.{}f}".format(quantity, self.cryptoDecimals),
        )

    except BinanceAPIException as e:
        print(e)

# Notifying that an order hast just been filled
def notifyOrder(self):
    totalPrice = 0
    totalQuantity = 0
    averagePrice = 0
    totalComissions = 0
```

```

        for i in self.newOrder["fills"]:
            totalComissions += float(i["commission"])
            totalQuantity += float(i["qty"])
            totalPrice += float(i["price"])*float(i["qty"])

        averagePrice = totalPrice/totalQuantity

        # Registering the order data
        self.orderRegister(self.newOrder["type"], self.newOrder["side"],
                            totalComissions,
                            self.newOrder["commissionAsset"],
                            averagePrice,
                            totalQuantity, totalPrice)

    # Registering all the purchase and sale orders made by the bot
    def orderRegister(self, orderType, side, comission,
                      coinComission, avgPrice,
                      qtyCrypto, qtyFiat):
        f1 = open("OrdersData/Registro_{0}_{1}_BOT.txt".format(self.crypto+self.fiat, self.frequency), "a+")

        if os.stat("OrdersData/Registro_{0}_{1}_BOT.txt".format(self.crypto+self.fiat, self.frequency)).st_size == 0:
            f1.write("DATE \t\t\t ORDER_TYPE \t PRICE \t
                     CRYPTO_QUANTITY \t
                     FIAT_QUANTITY \t
                     COMISSION \t
                     COIN_COMISSION \n")

        f1.write("{} \t{} \t {} \t\t {} \t\t\t {} \n".
                 format(str(dt.datetime.now()), orderType, side, str(avgPrice), str(qtyCrypto),
                        str(qtyFiat), comission, coinComission))

    f1.close()

    # Reading the downloaded candlestick data
    def getCandleData(self):
        self.candleData = pd.read_csv("MarketData/{0}/{1}/Freq_{2}.csv".
                                      format(self.crypto, self.fiat, self.frequency))
        self.dataCandles = len(self.candleData)

    # Calculation of the SMA Indicator
    def getsMA(self, days):

```

```
    self.candleData["SMA_{}".format(days)] = ta.SMA(self.
                                                candleData["close"], days
                                                )

    # Calculation of the EMA Indicator
def getEMA(self, days):
    self.candleData["EMA_{}".format(days)] = ta.EMA(self.
                                                candleData["close"], days
                                                )

    # Calculation of Stochastic Indicator
def getStochastic(self, days, pfast, pslow):
    STO = ta.stochastic(self.candleData, period = days, pfast =
                         pfast, pslow = pslow)
    self.candleData["STO_k_{}_{ }_{}_{ }".format(days, pfast, pslow)
                    ] = STO.k
    self.candleData["STO_d_{}_{ }_{}_{ }".format(days, pfast, pslow)
                    ] = STO.d
```

### A.3 botMain

Code Listing A.3: *botMain* file code

```
# TRADING BOT MAIN CODE: Binance trading bot code.

# Python libraries.
import pandas as pd
import numpy as np
import csv
import datetime as dt
import math
import os
import random
import sys

# Python libraries options.
pd.options.mode.chained_assignment = None # default='warn'

# Binance libraries.
from binance.client import Client

# Imported files.
sys.path.insert(0, os.path.abspath('../..'))
from BinanceKeys import BINANCE_KEYS
from botData import BINANCE, COIN, DATE, FREQUENCY
import botCore

# API data.
apiBinance = Client(BINANCE["APIKey"], BINANCE["SecretKey"])

# Available frequency in Binance.
frequencyAvailable = ["1m", "3m", "5m", "15m", "30m", "1h", "2h", "4h",
                      "6h", "8h", "12h", "1d", "3d", "1w", "1M"]
```

```

# Checking for a correct frequency.
def checkFrequency(freqAva: list, Frequency: str):
    check = False
    for i in range(len(freqAva)):
        if Frequency == freqAva[i]:
            check = True

    if check == True:
        print("The selected frequency of {} is correct. \n".format(
            Frequency))
    else:
        print("The selected frequency of {} is NOT correct. \n".
              format(Frequency))

    return check

# Filtering of downloaded data.
def initialDataFilter(Data):
    df = pd.DataFrame(Data)
    df = df.drop([6, 7, 9, 10, 11], axis=1)

    columns = ["time", "open", "high", "low", "close", "volume", "trades"]
    df.columns = columns

    for i in columns:
        df[i] = df[i].astype(float)

    df["start"] = pd.to_datetime(df["time"] * 1000000, format=' %Y-%m-%d %H:%M:%S' )

    return df

def downloadCandlesData(Coin, Fiat, Frequency, StartDate, EndDate):

    # Calculating the total number of candles to download according
    # to the selected time range
    FI = dt.datetime(int(StartDate[0:4]), int(StartDate[5:7]), int(
        StartDate[8:10]), int(
        StartDate[11:13]),
        int(StartDate[14:16]), int(StartDate[17:19]))
    FF = dt.datetime(int(EndDate[0:4]), int(EndDate[5:7]), int(
        EndDate[8:10]), int(EndDate[
        11:13]),
        int(EndDate[14:16]), int(EndDate[17:19]))

    # EndDate
    Total_Time = math.floor((FF - FI).total_seconds())

    if "m" in Frequency:
        if(len(Frequency)) == 2:
            CandleSeconds = 60*float(Frequency[0])
            CandleMinutes = int(Frequency[0])
        else:
            CandleSeconds = 60*float(Frequency[:2])

```

```
CandleMinutes = int(Frequency[:2])
elif "h" in Frequency:
    if (len(Frequency)) == 2:
        CandleSeconds = 3600*float(Frequency[0])
        CandleMinutes = 60*int(Frequency[0])
    else:
        CandleSeconds = 3600*float(Frequency[:2])
        CandleMinutes = 60*int(Frequency[:2])
elif "d" in Frequency:
    CandleSeconds = 86400*float(Frequency[0])
    CandleMinutes = 24*60*int(Frequency[0])
elif "w" in Frequency:
    CandleSeconds = 7*86400*float(Frequency[0])
    CandleMinutes = 7*24*60*int(Frequency[0])
elif "M" in Frequency:
    CandleSeconds = 30*86400*float(Frequency[0])
    CandleMinutes = 30*24*60*int(Frequency[0])

DataElements = math.floor(Total_Time/CandleSeconds)

Header = ["time", "open", "high", "low", "close", "volume", "trades", "start"]
Data = apiBinance.get_klines(symbol = Coin+Fiat, interval =
                               Frequency, limit =
                               DataElements)
Data = initialDataFilter(Data)

if not os.path.isdir("MarketData/{}/{}".format(Coin, Fiat)):
    os.mkdir("MarketData/{}/{}".format(Coin, Fiat))

if os.path.exists("MarketData/{}/{}Freq_{}.csv".format(Coin,
                                                       Fiat, Frequency)):
    os.remove("MarketData/{}/{}Freq_{}.csv".format(Coin, Fiat,
                                                    Frequency))

Data.to_csv("MarketData/{}/{}Freq_{}.csv".format(Coin, Fiat,
                                                 Frequency))

return DataElements

# TRADING BOT IMPLEMENTATION
if checkFrequency(frequencyAvailable, FREQUENCY):
    DataElements = downloadCandlesData(COIN["Crypto"], COIN["Fiat"],
                                       FREQUENCY, DATE["StartDate"],
                                       DATE["EndDate"])
else:
    print("Error for the selected frequency {}".format(
        FREQUENCY))

Bot = botCore.Bot_BinanceClass(BINANCE["APIKey"], BINANCE["SecretKey"],
                               COIN["Crypto"], COIN["Fiat"], FREQUENCY, DataElements)
```

## A.4 settings

Code Listing A.4: *settings* file code

```
# This class sets the data for the backtesting.

# Python libraries
import datetime as dt
from datetime import datetime

# Cryptocurrency pair (must be in Yahoo Finance list).
PAIR = {
    "Base": "BTC",
    "Quote": "USD"
}

# Stocks (must be in Yahoo Finance list).
STOCK = {
    "Ticker": "TSLA"
}

# Defining asset type: 0 for cryptocurrency, 1 for stocks.
ASSET = {
    "AssetType": 0
}

# Start and end dates (year-month-day) for the backtesting (
#                         datetime.now() for current date).
DATE = {
    "StartDate": "2022-01-01",
    "EndDate": "2022-07-15"
}

# Frequencies available for the candlestick data (must be in Yahoo
# Finance list).
FREQUENCIES = ["1m", "2m", "5m", "15m", "30m", "60m", "90m", "1h",
               "1d", "5d", "1wk", "1mo", "3mo"]
```

## A.5 sizers

Code Listing A.5: *sizers* file code

```
# This class defines the sizers for the backtesting.

# Python libraries
from backtrader.sizers import PercentSizer

class FullMoney(PercentSizer):
    """
    Setting the percentage of money to invest in a trade.

    @param PercentSizer: percentage of money to be invested.
    """


```

```
    @return none
    ,,
params = (
    ("percents", 3),
)
```

## A.6 downloadData

Code Listing A.6: *downloadData* file code

```
# BACKTESTING DATA EXTRACTION: User defined functions for
#                               extracting the datasets
#                               containing the information of the
#                               market (candles) from Yahoo
#                               Finance.

# Python Libraries
import pandas as pd
import numpy as np
import csv
import datetime as dt
import math
import os
import random
import yfinance as yf

def checkFrequency(FrequencyAva: list, Frequency: str):
    ,
    Checking for a correct frequency: Check if the frequency of the
    data selected by the user
    exists on the repository of
    Yahoo Finance.
    Otherwise, prints an error message suggesting the available
    frequencies.

    @param FrequencyAva: List of frequencies available from
                          Yahoo Finance.
    @param Frequency: User desired frequency.

    @return check: Boolean variable that is true if Frequency
                  exists in FrequencyAva.
    ,

check = False
for i in range(len(FrequencyAva)):
    if Frequency == FrequencyAva[i]:
        check = True

if check == False:
    print("The selected frequency of {} is not available.
          Select one of the
          following frequencies: \\"
```

```

                n".format(Frequency))
for i in range(len(FrequencyAva)):
    print("{} \t ".format(FrequencyAva[i]))
exit()

return check

def dataCrypto(FrequencyAva: list, Frequency: str, Pair: tuple,
               StartDate: str, EndDate: str):
    """
    Downloading the candlestick data into .csv files for
    cryptocurrencies.

    @param FrequencyAva: List of frequencies available from
                          Yahoo Finance.
    @param Frequency: User desired frequency.
    @param Pair: Coin pair operated.
    @param StartDate: Starting date of the downloaded data.
    @param EndDate: Ending date of the downloaded data.

    @return None
    """

# Downloading in case the frequency exists
if checkFrequency(FrequencyAva, Frequency):

    # Downloading data from Yahoo Finance
    Data = yf.download('{}-{}'.format(Pair["Base"], Pair["Quote"]),
                       start = StartDate,
                       end = EndDate,
                       index_as_date = False,
                       interval = Frequency)
    Data['time'] = Data.index
    Data['time'] = Data['time'].dt.strftime('%Y-%m-%d %H:%M:%S')

    # Generating the directory of the data
    if not os.path.isdir("MarketData/Crypto/{}{}".format(Pair[
                           "Base"], Pair["Quote"])):
        os.mkdir("MarketData/Crypto/{}{}".format(Pair["Base"],
                                                Pair["Quote"]))

    # Deleting the name in case it already exists
    if os.path.exists("MarketData/Crypto/{}{}Freq_{}.csv".
                      format(Pair["Base"], Pair["Quote"], Frequency)):
        os.remove("MarketData/Crypto/{}{}Freq_{}.csv".format(
                  Pair["Base"], Pair["Quote"], Frequency))

    # Generating .csv file from DataFrame
    Data.to_csv("MarketData/Crypto/{}{}Freq_{}.csv".format(
                Pair["Base"], Pair["Quote"], Frequency))

```

```

else:
    print("Error for the selected frequency ({})\n ".format(
        Frequency))

def dataStocks(FrequencyAva: list, Frequency: str, Ticker: str,
               StartDate: str, EndDate: str):
    """
    Downloading the candlestick data into .csv files for stocks.

    @param FrequencyAva: List of frequencies available from
                          Yahoo Finance.
    @param Frequency: User desired frequency.
    @param Pair: Coin pair operated.
    @param StartDate: Starting date of the downloaded data.
    @param EndDate: Ending date of the downloaded data.

    @return None
    """

# Downloading in case the frequency exists
if checkFrequency(FrequencyAva, Frequency):

    # Downloading data from Yahoo Finance
    Data = yf.get_data(Ticker, start_date = StartDate, end_date
                       = EndDate, index_as_date
                       = True, interval =
                       Frequency)
    Data['time'] = Data.index
    Data['time'] = Data['time'].dt.strftime('%Y-%m-%d %H:%M:%S'
                                             )

    # Generating the directory of the data
    if not os.path.isdir("MarketData/Stocks/{}".format(Ticker)):
        os.mkdir("MarketData/Stocks/{}".format(Ticker))

    # Deleting the name in case it already exists
    if os.path.exists("MarketData/Stocks/{}/Freq_{}.csv".format(
        Ticker, Frequency)):
        os.remove("MarketData/Stocks/{}/Freq_{}.csv".format(
            Ticker, Frequency))

    # Generating .csv file from DataFrame
    Data.to_csv("MarketData/Stocks/{}/Freq_{}.csv".format(
        Ticker, Frequency))

else:
    print("Error for the selected frequency ({})\n ".format(
        Frequency))

```

## A.7 strategy

Code Listing A.7: *strategy* file code

```

# Python libraries
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM

# Imported files
from settings import PAIR, STOCK, ASSET, DATE, FREQUENCIES
import downloadData

# Frequency of the data
dataFrequency = ['15m']

class Strategy(bt.Strategy):

    def __init__(self):

        # Downloading training data
        self.trainingData = downloadData.dataCrypto(FREQUENCIES,
                                                      dataFrequency, PAIR, DATE
                                                      ["StartDate"], DATE["EndDate"])

        # Data preprocessing
        self.trainingSet = self.trainingData.values
        self.trainingSet = np.reshape(self.trainingSet, (len(self.
                                                               trainingSet), 1))
        self.trainingSet = MinMaxScaler().fit_transform(self.
                                                       trainingSet)
        self.lengthTS = len(self.trainingSet)
        self.trainingX = self.trainingSet[0:self.lengthTS - 1]
        self.trainingX = np.reshape(self.trainingX, (len(self.
                                                       trainingX), 1, 1))
        self.trainingY = self.trainingSet[1:self.lengthTS]

        # Initializing the RNN
        self.RNN = Sequential()

        # Output layer is added
        self.RNN.add(Dense(units = 1))

        # Input layer and LSTM layer are added
        self.RNN.add(LSTM(units = 4, activation = 'sigmoid',
                          input_shape = (None, 1)))

        # Compiling the RNN
        self.RNN.compile(optimizer = 'adam', loss =
                         'mean_squared_error')

        # Fitting the RNN to the training data set

```

```
    self.RNN.fit(self.trainingX, self.trainingY, batch_size = 5
                 , epochs = 100)

    # Creating trained model
    self.trainedSet = self.trainingData.values
    self.input = np.reshape(self.trainedSet, len(self.
                                              trainedSet), 1)
    self.input = MinMaxScaler().transform(self.input)
    self.input = np.reshape(self.input, (len(self.input), 1, 1))
    self.trainedModel = self.RNN.predict(self.input)
    self.trainedModel = MinMaxScaler().inverse_transform(self.
                                                       trainedModel)
```

## A.8 backtestingCore

Code Listing A.8: *backtestingCore* file code

```
# BACKTESTING BOT CLASS FILE: functions defined to implement the
# backtesting.

# Python libraries.
import pandas as pd
import math
import backtrader as bt
import backtrader.analyzers as btanalyzers
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
import datetime as dt
import csv

# Imported files.
from settings import FREQUENCIES
import downloadData
import strategy

class BacktestingClass:

    def __init__(self, assetType: int, pair: dict, stock: str, date
                 : dict, dataFrequency: list):
        """
        Constructor of the class: setting all the information for
        the backtesting and
        downloading the
        subsequent candlestick
        data,
        which is entered into the strategy class.

        @param assetType: type of asset selected (0 for
                          cryptocurrency, 1 for
                          stocks).
        @param pair: cryptocurrency pair selected (must be in
                    Yahoo Finance list).
    
```

```
    @param stock: stock selected (must be in Yahoo Finance
list).
    @param date: start and end dates selected (year-month-
day).
    @param dataFrequency: data frequencies selected.

    ,

self.pair = pair
self.stock = stock

self.dtStartDate = dt.datetime(int(date["StartDate"][:4]),
                               int(date["StartDate"][5:7]),
                               int(date["StartDate"][:8:10]))
self.dtEndDate = dt.datetime(int(date["EndDate"][:4]), int(
                           date["EndDate"][5:7]),
                           int(date["EndDate"][:8:10]))
self.cerebro = bt.Cerebro()

for freq in dataFrequency:

    if assetType == 0:
        # Downloading data for cryptocurrency
        downloadData.dataCrypto(FREQUENCIES, freq, self.
                                 pair, date[""
                                 StartDate"], date
                                 ["EndDate"])

        # Cryptocurrency data feeded into the backtesting
        data = bt.feeds.YahooFinanceCSVData(
            dataname = ("MarketData/Crypto/{}/Freq_{}.csv"
                        ".format(self
                        .pair["Base"]
                        , self.pair[""
                        Quote"], freq
                        )),

            fromdate = self.dtStartDate,
            todate = self.dtEndDate,
            reverse = False)

    elif assetType == 1:
        # Downloading data for stocks
        downloadData.dataStocks(FREQUENCIES, freq, self.
                                 stock, date[""
                                 StartDate"], date
                                 ["EndDate"])

        # Stocks data feeded into the backtesting
        data = bt.feeds.YahooFinanceCSVData(
            dataname = ("MarketData/Stocks/{}/Freq_{}.csv".
                        format(self.
                        stock, freq))
            ,
```

```
        fromdate = self.dtStartDate,
        todate = self.dtEndDate,
        reverse = False)

    self.cerebro.adddata(data)

def setInitialMoney(self, money):
    """
    Setting initial cash of the portfolio.

    @param money: cash of the portfolio.

    @return none
    """

    self.cerebro.broker.setcash(money)
    self.InitialMoney = money

def printCurrentMoney(self):
    """
    Printing the current portfolio value.

    @return none
    """

    print("Current Portfolio Value: %.2f" % self.cerebro.broker
          .getvalue())

def setComissions(self, comissions):
    """
    Setting Binance commissions.

    @param comissions: broker comission for each trade.

    @return none
    """

    self.cerebro.broker.setcommission(commission = comissions)

def setSizers(self, sizer):
    """
    Setting portfolio percentage for each trade.

    @param sizer: sizers class.

    @return none
    """

    self.cerebro.addsizer(sizer)

def setStrategy(self, strategy):
```

```
    ,,,  
    Setting backtesting strategy.  
  
        @param strategy: strategy class.  
  
        @return none  
    ,,,  
  
    self.cerebro.addstrategy(strategy)  
  
# Run the implemented strategy  
def runStrategy(self):  
  
    ,,,  
    Running the implemented class.  
  
        @return none  
    ,,,  
  
    self.strats = self.cerebro.run()  
    self.strat = self.strats[0]  
    self.results = self.strat.analyzers.ta.get_analysis()  
  
def plotBacktestingResults(self):  
  
    ,,,  
    Plotting backtesting results: market analyzers and  
                                backtesting plots.  
  
        @return none  
    ,,,  
  
    self.printMarketAnalyzers()  
    self.cerebro.plot()  
  
def setBotAnalyzers(self):  
  
    ,,,  
    Setting bot analyzers (see documentation:  
    https://www.backtrader.com/docu/analyzers/analyzers/)  
  
        @return none  
    ,,,  
  
    self.cerebro.addanalyzer(bt.analyzers.TradeAnalyzer, _name=  
                            "ta")  
  
def printMarketAnalyzers(self):  
  
    ,,,  
    Printing market analyzers plot. If there are no trades,  
                                this function  
    warns the user.
```

```
    @return None
    ,,

    results = self.strat.analyzers.ta.get_analysis()

    if results.total.total==0:
        print("There have been no trades for the current
              backtesting.")

    return

else:
    closedTrades = results.total.closed
    wonTrades = results.won.total
    winRatio = wonTrades/closedTrades
    netProfit = results.pnl.net.total
    totalReturn = (netProfit/self.InitialMoney)*100

    totalDays = (self.dtEndDate - self.dtStartDate).days
    monthlyReturn = totalReturn*30.0/totalDays

    fig, ax = plt.subplots(1,1)

    column = ["Results"]

    rows = ["Closed trades",
            "Won trades",
            "Win ratio",
            "Net profit",
            "Total return",
            "Start period",
            "End period",
            "Monthly return"]

    results = [[ "{}".format(closedTrades)],
               [ "{}".format(wonTrades)],
               [ "{:.2f} {}".format(100*winRatio)],
               [ "{:.2f} {}".format(netProfit, self.pair[
                                     "Quote"])],
               [ "{:.2f} {}".format(totalReturn)],
               [ "{}".format(self.dtStartDate)],
               [ "{}".format(self.dtEndDate)],
               [ "{:.2f} {}".format(monthlyReturn)]]

    df = pd.DataFrame(results, columns = column)
    ax.axis('tight')
    ax.axis('off')
    ax.table = plt.table(cellText = df.values,
                         rowLabels = rows,
                         colLabels = column,
                         colWidths = None,
                         loc = "center")
    ax.table.set_fontsize(20)
    ax.table.scale(1.8, 1.8)
```

```
        for (row, col), cell in ax.table.get_celld().items():
            if (row == 0) or (col == -1):
                cell.set_text_props(fontproperties=
                                     FontProperties(
                                         weight='bold
                                         ,)))
        ax.table.auto_set_column_width(col=list(range(len(df.
                                                       columns))))
        plt.title("Backtesting Results", fontsize = 30,
                  fontweight="bold")
```

## A.9 backtestingMain

Code Listing A.9: *backtestingMain* file code

```
# BACKTESTING MAIN CODE: main file of the backtesting tool.

# Python libraries
import pandas as pd
import numpy as np
import csv
import datetime as dt
import math
import os
import random

# Python libraries options
pd.options.mode.chained_assignment = None # default='warn'

# Imported files
from settings import PAIR, STOCK, ASSET, DATE
import downloadData
import backtestingCore

# Import of strategy and sizers
import strategy as strat
import sizers

# Strategy is backtested.
backStrategy = backtestingCore.BacktestingClass(ASSET["AssetType"],
                                                PAIR, STOCK["Ticker"], DATE,
                                                strat.dataFrequency)
# Bot_BackTest.setComissions(Bot_1.makerCommission)
backStrategy.setInitialMoney(1000.0)
backStrategy.setSizers(sizers.FullMoney)
backStrategy.setBotAnalyzers()
backStrategy.setStrategy(strat.Strategy)
backStrategy.runStrategy()
backStrategy.printCurrentMoney()
backStrategy.plotBacktestingResults()
```



# Appendix B

## Studies Codes

This appendix aims to state constancy of all the code developed in order to perform the studies presented in Chap. 12. Please note that not all codes are written using the same programming language. Some have been implemented in Python, while others in Matlab, for convenience when coding.

### B.1 Simulation on Random Trade Closures

Code Listing B.1: File code for the simulation on random trade closures

```
import pandas as pd
import random
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import combinatorics

minWon = 48 #%
maxWon = 52 #%
if minWon > maxWon:
    minWon,maxWon = maxWon,minWon

div = 100
quan = (maxWon - minWon) * div + 1
percWonTrades = []
for i in range(quan):
    perc = minWon + i / div # %
    percWonTrades.append(perc)

initialInvestment = 15000 # u.m.
isComission = False
makerComission = 0.018 # %
maxTakeProfitPer = 10 # %

# LIMITS DEFINITION
maxPer = 6 # Maximum percentage
minPer = 5 # Minimum percentage
numDiv = 2 # Number of divisions in each % unit
```

```

perBetTrade = []
quantity = (maxPer - minPer) * numDiv

# In case of minimum percentage being null, the first trade
# percentage must be 0.5%
if minPer == 0:
    for i in range(quantity):
        percentage = minPer + (maxPer - minPer) * (i + 1) /
            quantity
        perBetTrade.append(percentage)
else:
    quantity2 = int(quantity + 1)
    for i in range(quantity2):
        percentage = minPer + (maxPer - minPer) * i / (quantity2 -
            1)
    perBetTrade.append(percentage)

numTrades = 40
numIt = 10
countPos = -1

for botPerc in percWonTrades:

    # Combination without repetition
    r = int(numTrades*botPerc/100)
    numIt = math.factorial(numTrades) / (math.factorial(r) * math.
        factorial(numTrades - r))
    print(len(combinaciones(comb, r)))

    for per in perBetTrade:
        per = per / 100 # unitary percentage

        for i in range(numIt):
            currentCapital = initialInvestment
            winCount = 0

            for j in range(numTrades):
                if isComission:
                    betTrade = currentCapital * per * (1 -
                        makerComission
                        /100)
                else:
                    betTrade = currentCapital * per;
                winLossNum = random.uniform(0, 100)
                if winLossNum > botPerc:
                    profitPer = - random.uniform(0,
                        maxTakeProfitPer
                        ) / 100
                else:
                    profitPer = random.uniform(0, maxTakeProfitPer)
                        / 100
                winCount += 1
                if isComission:
                    finalTrade = betTrade * profitPer * (1 -
                        makerComission

```

```

                /100)
        else:
            finalTrade = betTrade * profitPer
            currentCapital = currentCapital + finalTrade

        ROI = (currentCapital - initialInvestment) /
                           initialInvestment *
                           100
        winPer = winCount / numTrades * 100
        countPos += 1
        results[countPos] = [botPerc, per * 100, round(ROI, 2),
                             round(winPer,2)]

Results = pd.DataFrame(results, columns = ["Set Win Perc", "Bet
                                            Trade Perc", "ROI", "Real Win
                                            Perc"])
pd.set_option('display.max_rows', None, 'display.max_columns', None
              )

sns.set_theme(style="dark")
sns.color_palette("viridis", as_cmap=True)

figureResults, ax = plt.subplots()

for botPerc in percWonTrades:
    perBetTradeAnalysis = []
    ROIAnalysis = []
    ResultsFiltered = Results[ Results['Set Win Perc'] == botPerc ]

    for per in perBetTrade:
        ResultsFiltered2 = ResultsFiltered[ ResultsFiltered['Bet
                                                Trade Perc'] == per ]
        ROIMean = ResultsFiltered2["ROI"].mean()
        perBetTradeAnalysis.append(per)
        ROIAnalysis.append(ROIMean)

    sns.lineplot(perBetTradeAnalysis, ROIAnalysis, linewidth=2.0,
                 label="{}".format(botPerc))

ax.set_title("ROI vs Bet Trade %", size=14, fontweight="bold")
ax.set_ylabel("ROI (%)", size=12, fontweight='bold')
ax.set_xlabel("Bet Trade %", size=12, fontweight='bold')
ax.set_xlim(minPer, maxPer)
ax.grid()
ax.legend(shadow=True, fontsize=16)

plt.show()

```

## B.2 Combination Without Repetition

Code Listing B.2: File code for computing the combination without repetition

```
# Computes a combination without repetition
def combWithoutRep(n, r):

    c = [i for i in range(r)]
    cIni = c.copy()

    comb = [c.copy()]
    i = 0

    while i < len(c) - 1:

        if c[i] == c[i+1] - 1:

            i += 1

        if i == len(c) - 1:

            if c[i] < n - 1:

                c[i] += 1
                i -= 1

            while i > -1:

                c[i] = cIni[i]
                i -= 1

            comb.append(c.copy())
            i = 0

        else:
            c[i] += 1
            i -= 1

        while i > -1:

            c[i] = cIni[i]
            i -= 1

        comb.append(c.copy())
        i = 0

    return comb
```

### B.3 ROI vs Number of Trades for $\omega_p$ Range

Code Listing B.3: ROI vs Number of Trades for  $\omega_p$  Range

```

p = 0.0005;

A = 1;

wp = linspace(0.48,0.52,5);
wpLen = length(wp);

n = [1:40000];
nLen = length(n);

negLog = log(1-A*p);
posLog = log(1+p);

ROI = zeros(wpLen,nLen);

for j = 1:wpLen

    for i = 1:nLen
        res = n(i) * (wp(j) * (posLog - negLog) + negLog);

        ROI(j,i) = 10^res - 1;
    end
    i = 1;
end

plot(n,ROI(1,:))
title('ROI vs number of trades [A = 1, p = 0.0005]')
xlabel('Number of trades (n)')
ylabel('ROI')
grid on
grid minor
hold on
plot(n,ROI(2,:))
plot(n,ROI(3,:))
plot(n,ROI(4,:))
plot(n,ROI(5,:))
legend('wp = 0.48','wp = 0.49','wp = 0.50','wp = 0.51','wp = 0.52')
hold off

```

## B.4 ROI vs Number of Trades for $p$ Range

Code Listing B.4: ROI vs Number of Trades for  $p$  Range

```
p = [0.0005, 0.0015, 0.0025, 0.005];
pLen = length(p);

A = 1;

wp = 0.5008;

n = [1:40000];
nLen = length(n);

ROI = zeros(pLen,nLen);

for j = 1:pLen

    negLog = log(1-A*p(j));
    posLog = log(1+p(j));

    for i = 1:nLen
        res = n(i) * (wp * (posLog - negLog) + negLog);

        ROI(j,i) = 10^res - 1;
    end
    i = 1;
end

plot(n,ROI(1,:))
title('ROI vs number of trades [A = 1, w_p = 0.5008]')
xlabel('Number of trades (n)')
ylabel('ROI')
grid on
grid minor
hold on
plot(n,ROI(2,:))
plot(n,ROI(3,:))
plot(n,ROI(4,:))
legend('p = 0.0005', 'p = 0.0015', 'p = 0.0025', 'p = 0.0050')
hold off
```

## B.5 ROI vs Number of Trades for A Range

Code Listing B.5: ROI vs Number of Trades for A Range

```

A = [0.98, 0.99, 1, 1.01, 1.02];
ALen = length(A);

p = 0.0005;

wp = 0.50;

n = [1:40000];
nLen = length(n);

ROI = zeros(ALen,nLen);

posLog = log(1+p);

for j = 1:ALen

    negLog = log(1-A(j)*p);

    for i = 1:nLen
        res = n(i) * (wp * (posLog - negLog) + negLog);

        ROI(j,i) = 10^res - 1;
    end
    i = 1;
end

plot(n,ROI(1,:))
title('ROI vs number of trades [p = 0.0005, w_p = 0.50]')
xlabel('Number of trades (n)')
ylabel('ROI')
grid on
grid minor
hold on
plot(n,ROI(2,:))
plot(n,ROI(3,:))
plot(n,ROI(4,:))
plot(n,ROI(5,:))
legend('A = 0.98', 'A = 0.99', 'A = 1', 'A = 1.01', 'A = 1.02')
hold off

```



# Bibliography

- [1] *3 Commas - Crypto Trading Bot*. Visited on December 18, 2022. URL: <https://3commas.io/>.
- [2] *7 Best Bitcoin Hardware Wallet 2022 — Reviews And Comparisons*. Visited on December 6, 2022. URL: <https://medium.com/coinmonks/the-best-cryptocurrency-hardware-wallets-of-2020-e28b1c124069>.
- [3] *Backtrader - Analyzers*. Visited on October 16, 2022. URL: <https://www.backtrader.com/docu/analyzers/analyzers/>.
- [4] Fernando Blanco, Helena Matute, and Miguel A Vadillo. “Making the uncontrollable seem controllable: The role of action in the illusion of control”. In: *Quarterly Journal of Experimental Psychology* 64.7 (2011), pp. 1290–1304.
- [5] *Blockchains: The Technology of Transactions*. Visited on December 4, 2022. URL: <https://towardsdatascience.com/blockchains-the-technology-of-transactions-9d40e8e41216>.
- [6] Jesús Bobadilla. *Machine Learning y Deep Learning: Usando Python, Scikit y Keras*. Ediciones de la U, 2021.
- [7] Matthew Browne and Matthew J Rockloff. “Measuring behavioural dependence in gambling: A case for removing harmful consequences from the assessment of problem gambling pathology”. In: *Journal of Gambling Studies* 36.4 (2020), pp. 1027–1044.
- [8] *Capfolio*. Visited on December 18, 2022. URL: <https://www.capfol.io/>.
- [9] *CFI - Cryptocurrency Exchanges*. Visited on December 9, 2022. URL: <https://corporatefinanceinstitute.com/resources/cryptocurrency/cryptocurrency-exchanges/>.
- [10] CNMC. “Informe de Garantías y Etiquetado de la Electricidad”. In: *Cnmc* (2018), p. 29.
- [11] *Coinbase - What is "proof of work" or "proof of stake"?* Visited on December 7, 2022. URL: <https://www.coinbase.com/es/learn/crypto-basics/what-is-proof-of-work-or-proof-of-stake>.
- [12] *Coinbase - What is a smart contract?* Visited on December 8, 2022. URL: <https://www.coinbase.com/es/learn/crypto-basics/what-is-a-smart-contract>.
- [13] *Coinbase - What is mining?* Visited on December 7, 2022. URL: <https://www.coinbase.com/es/learn/crypto-basics/what-is-mining>.
- [14] *CoinMarketCup*. Visited on December 8, 2022. URL: <https://coinmarketcap.com/>.

- [15] *crypto.com - Cryptography and Cryptocurrencies – Putting the Crypto into Currency*. Visited on December 4, 2022. URL: <https://crypto.com/university/what-is-cryptography>.
- [16] *CryptoVantage - A Brief History of Cryptocurrency*. Visited on December 7, 2022. URL: <https://www.cryptovantage.com/guides/a-brief-history-of-cryptocurrency/>.
- [17] *EQUITY TRUST - Types of Cryptocurrency Explained*. Visited on December 8, 2022. URL: <https://www.trustetc.com/blog/cryptocurrency-types/>.
- [18] *EUROMONEY LEARNING - How does a transaction get into the blockchain?* Visited on November 13, 2022. URL: <https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain>.
- [19] Fan Fang et al. “Cryptocurrency trading: a comprehensive survey”. In: *Financial Innovation* 8.1 (2022), pp. 1–59.
- [20] *GitHub - blackbird*. Visited on December 18, 2022. URL: <https://github.com/butor/blackbird>.
- [21] *GitHub - catalyst*. Visited on December 18, 2022. URL: <https://github.com/scrlabs/catalyst>.
- [22] *GitHub - ccxt*. Visited on December 18, 2022. URL: <https://github.com/ccxt/ccxt>.
- [23] *GitHub - CryptoSignal*. Visited on December 18, 2022. URL: <https://github.com/CryptoSignal/crypto-signal>.
- [24] *GitHub - freqtrade*. Visited on December 18, 2022. URL: <https://github.com/freqtrade/freqtrade>.
- [25] *GitHub - golang-crypto-trading-bot*. Visited on December 18, 2022. URL: <https://github.com/saniales/golang-crypto-trading-bot>.
- [26] *GitHub - Krypto-trading-bot*. Visited on December 18, 2022. URL: <https://github.com/ctubio/Krypto-trading-bot>.
- [27] *GitHub - StockSharp*. Visited on December 18, 2022. URL: <https://github.com/StockSharp/StockSharp>.
- [28] *Google Finance*. Visited on December 8, 2022. URL: <https://www.google.com/finance>.
- [29] *Great Learning - Types of Neural Networks and Definition of Neural Network*. Visited on January 5, 2023. URL: <https://www.mygreatlearning.com/blog/types-of-neural-networks>.
- [30] *IBM - What are neural networks?* Visited on January 4, 2023. URL: <https://www.ibm.com/topics/neural-networks>.
- [31] *IBM - What are recurrent neural networks?* Visited on January 5, 2023. URL: <https://www.ibm.com/topics/recurrent-neural-networks>.
- [32] *INFOSEC INSIGHTS - What Is Crypto Mining? How Cryptocurrency Mining Works*. Visited on December 7, 2022. URL: <https://sectigostore.com/blog/what-is-crypto-mining-how-cryptocurrency-mining-works/>.
- [33] *Investopedia - ARIMA Model: Autoregressive Integrated Moving Average*. Visited on January 5, 2023. URL: <https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp>.

- [34] *Investopedia - Backtesting: Definition, How It Works, and Downsides*. Visited on December 24, 2022. URL: <https://www.investopedia.com/terms/b/backtesting.asp>.
- [35] *Investopedia - Binance Exchange*. Visited on December 15, 2022. URL: <https://www.investopedia.com/terms/b/binance-exchange.asp>.
- [36] *Investopedia - Blockchain Facts: Whay Is It, How It Works, and How It Can Be Used*. Visited on November 13, 2022. URL: <https://www.investopedia.com/terms/b/blockchain.asp>.
- [37] *Investopedia - How Does Bitcoin Mining Work?* Visited on December 4, 2022. URL: <https://www.investopedia.com/tech/how-does-bitcoin-mining-work/>.
- [38] *Investopedia - Non-Fungible Token (NFT): What It Means and How It Works*. Visited on December 8, 2022. URL: <https://www.investopedia.com/non-fungible-tokens-nft-5115211>.
- [39] *Investopedia - Stablecoins: Definition, How They Work, and Types*. Visited on December 8, 2022. URL: <https://www.investopedia.com/terms/s/stablecoin.asp>.
- [40] *Investopedia - Understanding Liquidity and How to Measure It*. Visited on December 9, 2022. URL: <https://www.investopedia.com/terms/l/liquidity.asp>.
- [41] *Investopedia - What Does Proof-of-Stake (PoS) Mean in Crypto?* Visited on December 7, 2022. URL: <https://www.investopedia.com/terms/p/proof-stake-pos.asp>.
- [42] Gary C Kessler. “An overview of cryptography”. In: (2003).
- [43] Ahmed M Khedr et al. “Cryptocurrency price prediction using traditional statistical and machine-learning techniques: A survey”. In: *Intelligent Systems in Accounting, Finance and Management* 28.1 (2021), pp. 3–34.
- [44] Ellen J Langer. “The illusion of control.” In: *Journal of personality and social psychology* 32.2 (1975), p. 311.
- [45] Dale T Miller and Brian R Taylor. “Counterfactual thought, regret, and superstition: How to avoid kicking yourself”. In: *What might have been*. Psychology Press, 2014, pp. 317–344.
- [46] John J Murphy. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.
- [47] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. In: *Decentralized Business Review* (2008), p. 21260.
- [48] *Python 3.11.1 documentation*. Visited on December 27, 2022. URL: <https://docs.python.org/3/>.
- [49] *Sangaku Maths - Combinations without repetition*. Visited on January 2, 2023. URL: <https://www.sangakoo.com/en/unit/combinations-without-repetition>.
- [50] *Sofi Learn - Guide to Crypto Staking: What It Is, How It Works, and How to Get Started*. Visited on December 7, 2022. URL: <https://www.sofi.com/learn/content/crypto-staking/>.

## BIBLIOGRAPHY

---

- [51] *SoFi Learn - Understanding the Different Types of Cryptocurrency*. Visited on December 8, 2022. URL: <https://www.sofi.com/learn/content/understanding-the-different-types-of-cryptocurrency/>.
- [52] Saurabh Suratkar, Mahesh Shirole, and Sunil Bhirud. “Cryptocurrency wallet: A review”. In: *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*. IEEE. 2020, pp. 1–7.
- [53] *techopedia - Constructor*. Visited on December 29, 2022. URL: <https://www.techopedia.com/definition/5656/constructor>.
- [54] *The Balance - How Does Blockchain Work For Small Business*. Visited on December 4, 2022. URL: <https://www.thebalancemoney.com/blockchain-explained-4773366>.
- [55] *The Motley Fool - What Is Proof of Work (PoW) in Crypto?* Visited on December 7, 2022. URL: <https://www.fool.com/investing/stock-market/market-sectors/financials/cryptocurrency-stocks/proof-of-work/>.
- [56] *TikZ.net - Neural networks?* Visited on January 4, 2023. URL: [https://tikz.net/neural\\_networks/](https://tikz.net/neural_networks/).
- [57] Jennifer Williams. “The psychology of cryptocurrency trading: Risk and protective factors”. In: (2021).
- [58] W Scott Wood and Maria M Clapham. “Development of the drake beliefs about chance inventory”. In: *Journal of gambling studies* 21.4 (2005), pp. 411–430.
- [59] *yahoo! finance*. Visited on December 24, 2022. URL: <https://finance.yahoo.com/>.

