Ca' Foscari
University
of Venice

Master's Degree programme in
Economics and Finance
curriculum: Finance

Final Thesis

Reinforcement Learning for financial trading: an application to the cryptocurrency market

Supervisor
Ch. Prof. Marco Corazza

Assistant supervisor
Ch. Prof. Silvia Faggian

Graduand
Edoardo Comis
Matriculation Number 974604

Academic Year
2020 / 2021

Acknowledgements

Before proceeding with the discussion, I would like to dedicate a few lines to all the people who have accompanied me on my course of study in the last years. You know I've never been good with acknowledgements so do not expect anything big!

This is the most difficult part of the thesis because it is the part that concludes everything. My university career ends with this work; many years in which I was able to discover the world of statistics first and then that of finance, making me appreciate them and become passionate about many aspects of these two interesting disciplines. Many years in which I met a lot of people, cultivated new friendships and visited new places. Many wonderful years in which I grew both as a person and as a professional, specializing in what I hope will soon be my job.

Moving on to acknowledgements, first of all, a special thanks goes to my supervisor, Professor Corazza, for his immense patience, for his indispensable advice and for the knowledge transmitted throughout the drafting process. I also thank my assistant supervisor, Professor Faggian, who contributed to the drafting of this thesis, as well as for the pleasure and involvement I felt in listening to each of her lessons.

A dutiful thanks obviously goes to my family, Stefano, Cathia and Lorenza, without whom I would not have been able to begin this university career. Now mum I promise you I will find a nice job in Boston, "massaciussessts" so you can come and see me and, most importantly, you can put the walk-in closet in my room. I absolutely thank my cat Micia and a little less the dog Tea because, after all, they are also part of the family.

I thank my grandparents Livio and Lina very much for always thinking of me and for whom I feel endless love. Unfortunately, due to this pandemic, we were no longer able to have the big lunches and dinners with the whole family that we had always organized. You will see that we will recover soon to celebrate all together. I take this opportunity to also thank the rest of my family, both near and far, the relatives that I see often and those who, also because of the distance, unfortunately I see less. I hope to celebrate this milestone with you soon.

I thank all the friends, the FEC and all my companions of adventures: thank you for having cheered me during these years of university with the outings, the happy hours, the dinners, the parties and all the trips we have done together. Without you I would surely have gone mad!

Last but not least I would like to say a huge thank you to my girlfriend Giuditta. I kept you for last so I can give you the space you deserve. After five years of relationship you understand how fundamental a person is in your life. Although with our ups and downs due to our difficult characters, I am very happy that you are by my side because you make me feel good. Thank you for all the good times we had together and for the support and trust you give me for whatever I do. Thank you for always rooting for me. And let's face it, I would not have been able to deliver the thesis without your help with English! Thanks again!

Contents

INTRODUCTION 7

1. CRYPTOCURRENCIES, DEFINITION AND RISKS 10
1.1 Main features 13
1.2 The reasons for success 16
1.3 The cryptocurrencies. 19
Bitcoin (BTC) 19
Ethereum (ETH) 20
Litecoin (LTC) 21
Bitcoin Cash (BCH) 21
Ripple (XRP) 22
Dash (Digital) Coin (DASH) 22
Monero (XMR) 23
1.4 Trading 24
Short Story 24
The current era 25
Importance of historical data 26

2. MACHINE LEARNING AND REINFORCEMENT LEARNING 28
2.1 Machine Learning: a brief introduction 28
2.2 Sequential Decision Processes 30
2.2.1 Markov Decision Process 31
2.3 Basic concepts 34
2.3.1 The Agent-Environment model 34
2.3.2 Estimation of the optimality of the policies 37
2.3.3 Optimal Value Function 39
2.3.4 Exploration vs Exploitation 40
2.4 Fundamental algorithms 42
2.4.1 Dynamic Programming 43
2.4.2 Monte Carlo methods 48
2.4.3 Temporal Difference methods 51

3. PRACTICAL APPLICATION OF REINFORCEMENT LEARNING TO ONLINE CRYPTOCURRENCY TRADING..... 57
3.1 Explanation of the chosen parameters 57
3.2 Assets chosen with time span, short mention 58
3.3 Structure of s-states and logarithmic returns 66
3.4 Structure of the actions 67
3.5 Transaction costs 67
3.6 Reward functions 68
3.7 Squashing function 69
3.8 Phase definitions 70

4. APPLICATION OF THE SARSA AND Q-LEARNING METHODS TO THE CRYPTO-CURRENCY MARKET: Observations and results. 73
4.1 BITCOIN 73
4.2 BITCOIN CASH 81
4.3 DASH-COIN 87
4.4 ETHEREUM 92
4.5 LITECOIN 97
4.6 MONERO 102
4.7 RIPPLE 106

5. FINAL CONSIDERATIONS AND OBSERVATIONS 110
5.1 Observations and Final Results 110
5.2 Conclusions 116

BIBLIOGRAPHY 118

INTRODUCTION

The advent of the cryptocurrency market dates back to the end of 2008 / beginning of 2009 when Satoshi Nakamoto presented *Bitcoin*, the virtual currency for operating secure transactions on a decentralized peer-to-peer network that guarantees anonymity and transparency thanks to the use of cryptographic algorithms. Users who respect their privacy would appreciate their promise of protection, anonymity and the lack of a central controlling authority.

The market price of the first cryptocurrency had a stable trend until 2013 – always remaining below $ 30 – and then continued to rise to such an extent that it culminated in a clear speculative bubble in 2017, when the purchase price of a Bitcoin rose from $ 1.000 to $ 20.000 within a few months. In fact, in recent years, interest in the new technology has been continuously growing and several new cryptocurrencies have been launched on the market in order to ride the wave of the current trend.

The analogies of online cryptocurrency trading with the financial market are not lacking. Just as with investments in shares, the trader hopes to make a profit by reselling or repurchasing the assets in his possession at a more advantageous price than that of the previous operation. The investor's bet lies in guessing the forecast of the future trend and gaining both from the rise in price – using the long-selling (LONG) strategy – as well as from the descent – thanks to short-selling (SHORT) investments. In the first case, a positive profit is obtained by buying a security at a discount and reselling it once the price has risen: the gain is the difference between the selling price and the purchase price, net of taxes. On the other hand, the SHORT technique is purely speculative and consists of short selling: the trader sells the securities he has borrowed and bets on the decrease in price; this way he can make a profit when he returns the action back to the lender with the related interest. In the paper, together with these two market positions, I also consider the possibility of staying out of the market.

Just as the key to buying and selling financial instruments on the stock exchange is knowing how to study and exploit market trends in order to make profits, the choice of investing in cryptocurrencies for profit is also essentially a bet on the future price of the currency. The search for strategies to maximize earnings in the stock market have been probed for decades, but it is not immediate to understand whether these solutions are also valid for the cryptocurrency market where the variability of rates of return seems to have no apparent limit and the opportunity of a high gain is tangible, but it could equal the level of a large loss.

Machine Learning and related techniques have made significant progresses in recent years. Some of these technological breakthroughs have led to the creation and improvement of products used by billions of people around the world. Many researchers have based their efforts on applying machine learning and related technologies to financial markets since the beginning of machine learning research. Stock market forecasting and detecting manipulation are some examples of a large body of research in this field. Cryptocurrencies are also considered a financial asset by many users, and a significant amount of research applied to financial markets can be applied to this field as well.

Since the invention of Blockchain technology about a decade ago, most of the research published in this area has focused on the non-technological aspects of Blockchain technology such as legal issues and its role in criminal activities. Because of the novelty of Blockchain technology and the quick advancements in machine learning techniques, research on their intersection is less established and widespread than in many other fields. Consequently, existing articles in this area need to be revised to help researchers to understand better the current research landscape.

The goal of this thesis is to understand if an efficient analysis method (*reinforcement learning*) for trading on the stock market can also be applied with good results on the cryptocurrency market. Most trading strategies have, as input, the calculation of technical indicators on the data of the historical series; the same type of study is applied to the digital currency sector.

In the first chapter of this thesis, we will first present cryptocurrencies, their history and their importance nowadays, then linking to the increasingly expanding phenomenon of online trading. In the second chapter we will address the theory behind machine learning by going into detail about reinforcement learning: we will analyse the main characteristics and we will begin to expose the techniques and the structure of the SARSA and Q-Learning algorithms. Both algorithms have been implemented in MATLAB environment and exposed in chapter three by presenting a practical application of the algorithm obtained from the proposed theory. In chapter four, the results will be presented and commented in detail. Finally, in the fifth and last chapter we will summarize the main observations providing conclusions and ideas for future research

Will the high volatility of crypto currencies – which scares investors so much – lead to obtain good results, and consequently a profit, or will it mislead the algorithm by sending the whole system at a loss?

1. CRYPTOCURRENCIES, DEFINITION AND RISKS

In this first chapter, I will introduce what cryptocurrencies are, starting from their history and their evolution up to the present day. I will focus on the importance they have nowadays, both for the large number of users who use them and for the digitalization process of the transactions they are favouring. I will explain their complicated structure: is true to say that cryptocurrencies make transactions really safe, but they also allow criminal companies to abuse them. Next, I will introduce the cryptocurrencies that I decided to bring and analyse in my thesis. Each crypto has its own characteristics and peculiarities that make it unique compared to the others. I chose the most famous ones but also the most different – which is why they are so successful. There are so many cryptocurrencies nowadays that would be worthy to analyse but, for a matter of time and amount of data, I decided not to select. Finally, I presented a bit of trading history to understand and grasp how such a digitalization of monetary exchanges has come about today.

The financial sector – with particular reference to the methods of assets exchange, services and any financial activity – is undergoing a radical change. As expected, it proceeds step by step with the progress of cryptography, that is the discipline that allows you to make a message intelligible only to authorized people.[1] Among the applications of digitalization – that have generated the greatest impact on the financial sector – the birth of "cryptocurrency" or "virtual currency" stands out. Rightfully entered among the new protagonists of the financial world, cryptocurrencies have revolutionized the Internet and the way of understanding money. The expression "cryptocurrency" – literally "hidden currency" – testifies and confirms the possibility of using it only if aware of a certain access code. Unlike how we are – or perhaps it would be better to say we were – used to identify and handle the currency in our daily life, cryptocurrency does not exist in physical form: they are exchanged only electronically. Many concepts traditionally used for legal tender currencies are adapted to the context: for virtual currency we no longer speak of "wallet", but of "e-wallet"[2].

The most distant ancestor of cryptocurrency is eCash, the brainchild of David Chaum[3], a payment system designed to send and receive electronic money anonymously and securely. Through this new software, the user can use money in digital format cryptographically signed by an American bank

[1] On this point, see Consob.it, *Le criptovalute*, www.consob.it/web/investor-education/criptovalute.

[2] A virtual wallet that allows users to make payments - online or in physical stores - using electronic devices. In other words, the e wallet is a secure tool that can store credit, debit, prepaid or bank account numbers to make payments quickly and easily.

[3] For further information, please refer D. Chaum, *Blind signature for untraceable payments*, in R. L. Rivest e A. T. Sherman (eds), *Advances in cryptology: proceedings of crypto 82,* Springer book archive, 1983, pp. 199 – 203. And for his life: https://academy.bit2me.com/en/who-is-david-chaum/.

– the “Mark Twain” of Saint Louis – which used this system for micropayments from 1995 to 1998, until the bank was bought by Merkantile Bank, a large issuer of credit cards – an instrument that effectively makes Chaum's software less useful. In fact, in Europe, where cash transactions were considerably preferred to credit cards, in my opinion establishing a digital micro-payment system would have made much more sense. In fact, it was in June 1998 that eCash became available in Switzerland through Credit Suisse, in Germany through Deutsche Bank, in Austria with UniCredit Bank Austria AG, in Sweden with the Swedish postal service Posten AB and in Norway at Den Norsken Bank.

Despite the great potential of this new tool, DigiCash – the company started by Chaum in 1990 with eCash as a trademark – went bankrupt in 1998: the credit card remains the "*currency of choice*", the preferred one. «*As the Web grew, the average level of sophistication of users dropped. It was hard to explain the importance of privacy to them*»[4]. For its failure, the father of eCash blames the failure of users to give privacy the importance it deserves, underestimating a payment system that would allow them to make purchases without necessarily having to open an account with the supplier or transmit credit card numbers.

In august 2009, the one who prefers to be called Satoshi Nakamoto[5] – but whose identity is still unknown – published a new protocol in which he illustrated a system of *peer-to-peer* electronic payment through the first cryptocurrency, as of today the most famous in the world: bitcoin[6]. It is a payment system that guarantees anonymity and transparency thanks to the use of cryptographic algorithms that make transactions and user wallets safe. The term “peer-to-peer” – which stands for “peer network” – means a network of personal computers – called “nodes” – which interact directly by sharing multimedia resources in the absence of intermediary servers. The architecture of the bitcoin’s software is opposed for the first time to client-service services, where communications from various customers are addressed to a single central entity.

[4] Cf. J. Pitta, *Requiem for a bright idea,* 1999, www.forbes.com

[5] Who is Satoshi Nakamoto, and what is his story? The founders of the Bitcoin cryptocurrency go by the pseudonym Satoshi Nakamoto. Although the name Satoshi Nakamoto is often associated with Bitcoin, the real person who bears the name has never been identified, leading many to speculate that it is a pseudonym for a person or group of persons with a different identity.

[6] For further information see S. Nakamoto, *Bitcoin: a peer-to-peer electronic cash system*, October 2008, www.cryptovest.co.uk.

The main functions (peer to peer) of this new decentralized mechanism are three:

- *Discovering*: i.e., the direct discovery of new peers;
- *Sharing*: direct sharing of files within a network;
- *Querying*: direct request for content from other peers.[7]

"The peer-to-peer network potentially allows any user to transfer money at almost instantaneous speed at very low or no cost (...)"[8]. A practical example of a transaction that uses the peer-to-peer method was recently adopted by Poste Italiane, a symptom that the absence of intermediaries can undoubtedly attract the attention of users: for just over three years it has been possible to transfer money from a postepay card to another one simply through the application that can be downloaded on your smartphone. The only data required is the telephone number of the user to whom you want to send or ask for the amount; the transfer does not require commissions for a sum less than or equal to € 25. This method greatly reduces times and costs, and also allows users to conclude a transaction without necessarily having to know personal data or the counterparty's card number. Undoubtedly, this is a successful system that represents a milestone in financial innovation.

The new Nakamoto protocol, which responds to its own rules[9], makes use of an unprecedented technology called *blockchain technology*, which does not require the intervention of intermediaries – banks or other financial institutions – to ensure that users can exchange money safely and directly at the same time. A "*distributed ledger*" or "*blockchain*" is an open, distributed ledger that can store transactions between two parties in a secure, verifiable and permanent way. The participants in the system are defined as '*nodes'* and are connected to each other in a distributed manner. It is an ever-growing list of records called *blocks*, which are linked together and secured through the use of cryptography. The data in a block are inherently immutable and for this reason they cannot be retroactively altered without all subsequent blocks being modified; to do this, it would be required the consent of the majority of the network. Each block is identified by a code that contains both the information of a series of transactions and the code of the previous block, so to make it possible to retrace the chain back to the original block – a sort of transaction DNA. All the nodes of the network store all the blocks and, therefore, the whole Blockchain. The underlying technology of a great number of cryptocurrencies is called the Blockchain.

[7] On this point, see P. Cellini, *La rivoluzione digitale*, Luiss University Press, 2018, Rome, pp. 330 – 331.
[8] Cf. S. Capaccioli, *Criptovalute e bitcoin: un'analisi giuridica*, Giuffrè editore, Milan, 2015, quote at p. 254.
[9] So D. Knezevic, *Impact of blockchain technology in changing the financial sector and other industries*, March 2018, Montenegrin Journal of Economics.

"The most critical area where Blockchain helps is to guarantee the validity of a transaction by recording it not only on a main register but a connected distributed system of registers, all of which are connected through a secure validation mechanism." - Ian Khan[10]

1.1 Main features

Virtual currency has been defined by the European Central Bank as "*a representation of digital value which is neither issued by a central bank or public body nor is it necessarily linked to a legal currency but is accepted by individuals and legal entities as means of payment and can be transferred, stored or exchanged electronically"*[11]. In order to highlight the possible risks arising from the use of cryptocurrencies, the Bank of Italy identifies their main characteristics[12]:

- They are created by a private broadcaster according to their own rules to which the members of the community choose to adhere;
- They are not physically held by the user, who owns an electronic walled best known as e-wallet which can be accessed via a password; in fact, these wallets are software developed by qualified parties such as wallet providers. In some cases, at a later time it is possible to convert the digital currency into legal money and vice versa through exchange platforms;
- Holders of electronic wallets remain anonymous;
- Transactions involving the transfer of virtual money are technically irreversible, thereby it is never possible to request cancellation;
- The cryptocurrency is not legal tender, it can be used for the purchase of goods and services only if the accipients is available to accept it.

It is inevitable that these characteristics inherently involve risks. On the basis of their own rules, the fact that each user can create a new crypto currency leaves room for unbridled competition but, above all, a game without rules – in the sense that there are no strict rules in creating crypto – can easily mislead those investors who do not master the cryptocurrency market. Furthermore, being an entirely

[10] See Bhramrah Dipender, "Blockchain internet of transaction", June 2018, BPB Publications, p. 10.

[11] Cf. Banca Centrale Europea, *Parere della Banca Centrale Europea su una proposta di direttiva del Parlamento europeo e del Consiglio che modifica la Direttiva (UE) 2015/849 relativa alla prevenzione dell'uso del sistema finanziario a fini di riciclaggio o finanziamento del terrorismo e che modifica la Direttiva 2009/10/CE*, October 2016, Official Journal of the European Union, paragraph 1.1.3. The definition seems to be based on the one proposed in the paragraph 19 of «*Opinion on virtual currencies*» of the European Banking Authority of 4 July 2014 (EBA / Op / 2014/08) available on the EBA website at www.eba.europa.eu.

[12] For further information, please refer to A. Caponera e C. Gola, *Aspetti economici e regolamentari delle "cripto-attività"*, in the Bank of Italy (edited by), *Questioni di Economia e Finanza (occasional papers)*, March 2019, www.bancaditalia.it.

virtual system, the risk of delivering our entire wallet into the hands of potential hackers is crucial. However, an even more critical aspect lies in the anonymity of transactions, a feature that undoubtedly attracts the attention of criminal organizations[13].

It is also possible to divide virtual currencies into three macro-types, based on their interaction with legal money[14]:

1. Virtual currency that cannot be purchased, is "non-convertible," or is "closed": it is unrelated to the real economy and users typically pay a monthly subscription fee before earning virtual money based on their online performance. This money is so named because it does not have the ability to be converted into legal tender currency and also, is only useable within the virtual community; World of Warcraft (WoW) Gold is an example of a virtual money used in this well-known online role-playing game in which a player has many options and subscription fees for making an account and beginning to play;
2. Currency with “limited convertibility”: it can be bought with real money at a set exchange rate, but it cannot be turned back into legal tender money. The scheme owner determines the conversion circumstances. You can always buy virtual products and services but some currencies – such as Facebook Credits – may also let you buy actual things and services (FB);[15]
3. “Fully convertible” currency is one that can be bought and sold for standard currency. In terms of compatibility with the real world, the virtual currency is similar to any other convertible currency. The Linden Dollar (L$) is an example of a virtual currency that can be used to buy both virtual and real products and services. It is a virtual currency used in Second Life, a virtual environment where users can create their own avatars: users can exchange goods and services in Second Life's own economy and, to do so, they will require Linden Dollars, which may be acquired with any of the system's permitted currencies at the current exchange rate. Users can exchange their unused Linden Dollars for US dollars.

[13] I will not delve into this topic but for more information see Sicignano, Gaspare Jucan, “Bitcoin e riciclaggio”, April 2019, Giappichelli, p. 111.
[14] See European Central Bank, *Virtual currency schemes*, October 2012, p.16, www.ecb.europa.eu.
[15] See http://developers.facebook.com/blog/post/2012/06/19/introducing-subscriptions-and-local-currency-pricing/

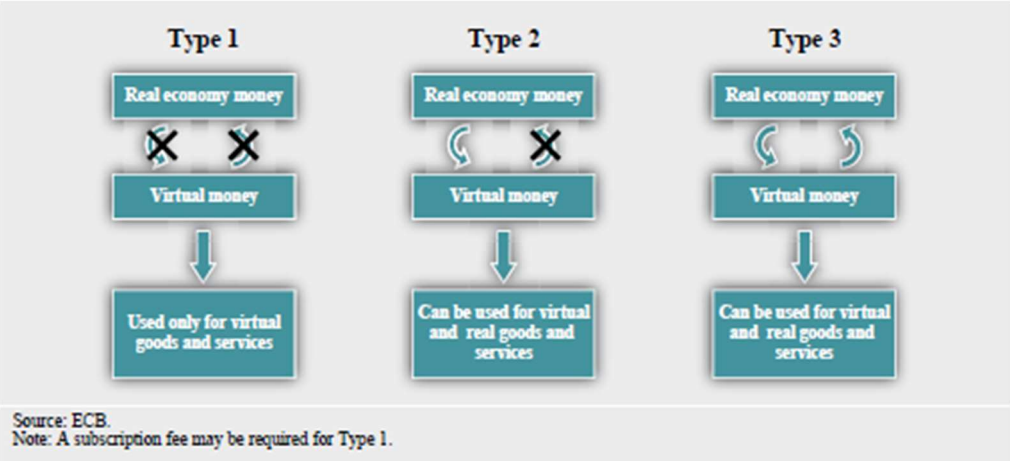


Figure 1: The three types of currencies. Investopedia. Source: ECB.

Specifically, the difference between cryptocurrency and traditional currency lies in the inability of the virtual currency to fulfil the three functions of an official currency: unit of account, means of payment and store of value. The cryptocurrency's unit of account function is intuitively less when their values are subject to wide fluctuations, even within the same day. Finally, as regards the store of value function, it must be borne in mind that the number of cryptocurrency units that can be produced is limited, therefore the more transactions settled in cryptocurrencies, the greater their value. As a result, virtual currencies are unlikely to be used as a store of value.

After underlining the aspects that distinguish cryptocurrency from official currency, care must be taken not to confuse it with electronic money. The requirements of the latter are meticulously reported in Article 1, par. 3, letter b, of Directive 2000/46 / EC, *"concerning the start-up, operation, and prudential supervision of the activity of electronic money institutions"*. It is defined as "*a monetary value represented by a receivable from the issuer that is:*

1. *Stored on an electronic device*;
2. *Issued upon receipt of funds whose value is not less than the monetary value issued*;
3. *Accepted as means of payment by companies other than the issuer".*

In conclusion, it seems to be more correct to classify virtual currencies as an "asset", as referred to in art. 810 of the Italian Civil Code, meaning anything, be it material or immaterial, suitable for satisfying a human need or utility, or even as the possibility of undergoing expropriation and limitation and that, as such, represents the object of rights.

1.2 The reasons for success

The currently uncontrolled expansion of cryptotrading has been, and still is, a wake-up call for international authorities, governments, central banks and supervisory authorities. Specifically, the EBA (European Banking Authority) has investigated and described the phenomenon, highlighting the benefits, both economic and individual, which incentivize individuals to enter the almost completely shady world of cryptocurrencies[16].

Among the economic benefits, surely stands out the triad "economy - speed - safety". The expected transaction costs are undoubtedly lower by virtue of the absence of competitors, on average equal to 1‰ of the amount. Another aspect not to be ignored is certainly the speed of transactions. We have to underline the essentiality of this feature: some of the main altcoins were born precisely with the aim of further accelerating the times envisaged by the Bitcoin platform (Litecoin, Bitcoin Cash, see chapter 1.4). We are also dealing with non-stop platforms, a service available 24/7 – unlike the traditional intermediaries we are used to. In addition, the security and irreversibility of transactions, which are all transcribed in a common register, guarantee a transparency that we do not always enjoy in traditional markets. Indeed, it is not uncommon for traders to have to pay consumer-initiated charges on the basis of false declarations of non-delivery of the product[17]. This potential, however, can actually represent a double-edged sword: the irreversibility of transactions does not protect users from any errors or fraud from which, literally, there is no turning back.

However, on a personal level, the opportunity to be able to transfer money while maintaining anonymity keeping your personal data should not be underestimated. In this sense, paying with virtual currency is like paying in cash, eliminating the risk of handing your data into the hands of those who could abuse it[18]. Finally, let's not forget that blockchain technology was born with the aim of starting a decentralized system, which did not suffer from the control of central authorities with the ability to influence the money supply. One of the benefits of the crypto market is being able to manage transactions with the relative possibility of transferring money beyond EU borders, sometimes taking advantage of less developed and less reliable foreign regulations. In fact, if on one hand this opportunity guarantees financial integration – I would dare to say – worldwide, on the other hand it allows criminal organizations, by collaborating with other organizations abroad, to take advantage of

[16] For further information, please refer to European Banking Authority, *Opinion on virtual currencies*, July 2014, pp. 16 – 21, www.eba.europa.eu.

[17] Cf. J. Brito, *Beyond silk road: potential risks, threats, and promises of virtual currencies, Testimony before the Senate Committee on homeland security and governmental affair*, 18 November 2013, p. 10.

[18] Cf. R. Wu, *Why we accept Bitcoin*, Forbes, February 2014, www.forbes.com.

less stringent and less severe legislation in force in other countries. Unfortunately, in my opinion it is necessary to admit that the very combination of "anonymity – absence of control" arouses the interest of criminal organizations towards cryptocurrencies and, therefore, represents one of the reasons why this market has received – and is enjoying – an important success. The data confirm this: in February 2018 the director of Europol Rob Wainwright estimated an amount of cash of illicit origin equal to 113 billion euros, of which 4% has been converted into cryptocurrencies, for a total of approximately 4.5 billion euros[19]. The European Parliament itself confirms that a percentage of all crypto transactions between 25 and 50% could be destined for illicit purposes[20].

To summarize, the reasons that led to the development and rapid growth of this dark market are: cheapness, speed and security of transactions and the possibility for criminal organizations to operate in an unregulated context. In addition, the three European supervisory authorities EBA (European Banking Authority), ESMA (European Financial Instruments and Markets Authority) and EIOPA (European Insurance and Occupational Pensions Authority) identify another reason in the lack of awareness of the risks involved in using cryptocurrencies. The EBA in particular, tracks 70 risks that all those who use the platforms of the crypto market run, from users to service providers, to the entire financial system. Among the risks that the European authority recognizes as "high risks", the possibility of significant and unexpected changes in the exchange rate stands out. The absence of a central authority that stabilizes the rates makes the price formation operation not very transparent, as well as easily manipulated by a small group of subjects who own an important amount of cryptocurrency. To this aspect are added the intrinsic characteristics of the system that make the price highly volatile. Furthermore, the halving date itself, or the birth of an application that allows users to play on the platform by exchanging virtual currency, can quickly generate a steep rise in price. But on the other hand, the announcement of a possible regulatory intervention – as in fact happened in 2018 – could push the price down. The EBA includes this risk among the medium risks: the possibility of a regulatory intervention would force users either to comply with the rules that will certainly make the system – previously decentralized – less advantageous or to sell, regardless of the market conditions that not favourable. The same applies to the eventuality that the authorities decide to tax transactions involving virtual currency (medium risk): also, in this case the user will be forced to comply with the rules or to sell, possibly recording a significant loss. On the other hand, among the high risks, together with a

[19] So K. Corcoran, *Europol: i criminali ricorrono alle criptovalute per riciclare 5,5 miliardi di dollari in contanti di provenienza illecita in Europa,* February 2018, www.businessinsider.com.

[20] For further information, please refer to M. Dabrowski (CASE) e L. Janikowski (CASE), *Virtual currencies and central banks monetary policy: challenges ahead,* July 2018, www.europarl.europa.eu.

total absence of deposit guarantee, supervision and legal and contractual protection – the most underrated risk in my opinion – is the lack of legal tender of virtual currencies. This is because those who buy them could risk not being able to sell them, exchange them for virtual currencies or make purchases, being the acceptance of payment in cryptocurrency on a voluntary basis; fully convertibles may also not be accepted. In fact, there is a risk of not having an exit option and incurring losses in the meantime[21].

Finally, it is true that a decentralized system guarantees several advantages to users, but it is also true that an entire system based on an algorithm and computer codes does not necessarily correspond to the ideal alternative. In fact, there are two high risks that derive directly from this aspect. First of all, the possibility that malfunctions and crashes in the operating system could send the system into a tailspin, effectively preventing the user from selling and / or buying when he wants, that is when market conditions suggest it. Lastly, episodes of hacking and cyber-attacks of various kinds are not uncommon. In early 2014, the cyber-attack that hit the Japanese cryptocurrency exchange platform Mt. Gox resulted in the loss of approximately 750,000 bitcoins owned by thousands of users[22]. The ways in which a hacker can attack a computer system are so vast and complex that it is highly unlikely that the user who approaches the crypto market for the first time knows them all and knows how to defend himself. This is exactly what the three European supervisory authorities mean when they say that individuals choose to invest in cryptocurrencies also and above all because they are not fully aware of the risks involved.

In recent years, interest in new technology has grown continuously and several new cryptocurrencies have been launched on the market to ride the wave of the current trend. Some new cryptocurrencies have originated from a real detachment from the Bitcoin Blockchain, called “hard fork”. Since changes to the native code of any single cryptocurrency are not allowed, minting a new currency is the only way to make consistent updates. The new coin created is exactly the same as the parent Blockchain up to the fork point, but the new transactions will no longer be backwards compatible. Not all cryptocurrencies were born through hard forks, but they can equally be considered in many respects as those already well established on exchange platforms. Finally, have also been developed

[21] Cf. European Insurance and Occupational Pensions Authority, *Avviso, L’ESMA, l’ABE e l’EIOPA informano i consumatori sui rischi delle valute virtuali,* 2014, www.eiopa.europa.eu.

[22] Cf. Banca d’Italia, *Avvertenze sull’utilizzo delle cosiddette “valute virtuali”*, 30 January 2015, Rome, www.bancaditalia.it.

cryptocurrencies that present new distinctive features such as to make them an innovative product on the market[23].

1.3 The cryptocurrencies.

In this thesis, I selected seven different cryptocurrencies based on their popularity, availability of historic price data and market cap. These cryptocurrencies are: Bitcoin, Ethereum, Bitcoin Cash, Litecoin, Ripple, Dash Coin and Monero.

Bitcoin (BTC)

Bitcoin is a collection of ideas and technology that serve as the foundation for a digital currency ecosystem.[24] This ecosystem includes a currency, the units of which are called bitcoin, which is used to store and transmit value between participants in the Bitcoin network. Bitcoin users can communicate with each other through the Bitcoin protocol – mainly via the Internet – although there are other networks they can use as well. In addition to money and software, Bitcoin also refers to the P2P (peer-to peer) network that is formed as a result of the operation of the Bitcoin software. BTC uses a database distributed over a P2P network to carry out commercial transactions, along with digital signatures and "proof-of-work" (POW) to ensure security. This ensures that BTC can only be spent by the person who owns it. The P2P typology and the lack of a centralized administrative system make it difficult for any authority to manipulate the amount of bitcoin in circulation and, consequently, prevent inflation.[25] Users may move bitcoins over the network to buy and sell things, as well as send money to persons or organizations, just as they would with traditional currencies. Bitcoins may be purchased or sold at a set rate and swapped for another currency. Bitcoin is an ideal form of money for the Internet since it is quick, secure, and borderless in certain aspects.[26] Bitcoin operates without the need of a central authority or banks, instead relying on peer-to-peer technology; transaction administration and Bitcoin issuance are handled collectively by the network. Bitcoin is an open source project with a public design: no one owns or controls Bitcoin, and anybody may participate in it.[27]

[23] Hard Fork: Hard Forks are software updates that are not compatible with previous versions. When a Hard Fork occurs, therefore, the blockchain at the closure of a given block will split into two parts, giving rise to a new chain subject to rules different from the original one, and the latter will not be affected in any way. The two chains will coexist simultaneously, but all participants will have to upgrade to the new software to continue participating and validate new transactions.
[24] Chris Clark, Bitcoin Internals: A Technical Guide to Bitcoin, 31/06/2013.
[25] Andreas M. Antonopoulos, Mastering Bitcoin, O' Reilly, 2014.
[26] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 14/12/ 2010.
[27] From Internet: https://bitcoin.org

With Bitcoin:

- The cost of transactions can be reduced to a few cents (in contrast to the rate costs of typical credit cards);
- Electronic payments can be confirmed in less than an hour with no transfer or commission costs;
- There is a limited risk of monetary inflation because the bitcoin production frequency is limited by an algorithm and cannot produce more than 21 million bitcoins;
- Payments are irreversible, so as to reduce the risk of fraud;
- Payments can be made without identification;
- The responsibility rests with consumers, who can completely lose their bitcoins if they lose their encryption keys.[28]

The singular characteristic of the Bitcoin system of being open source on one hand requires that no entity controls and manages the platform; on the other hand, that anyone participating in the network can make changes to the system in order to improve the underlying mechanism. This possibility has inevitably favoured the opportunity to generate new cryptocurrencies, with the aim of guaranteeing a more or less innovative improvement to the Bitcoin platform.

Based on this aspect, we can define "**Altcoin**" – short for "alternative coin" – all virtual currencies, alternatives to bitcoin, created to replace or improve at least one component of the system[29]. In the following paragraphs we will analyse more carefully the main Altcoins, specifically Bitcoin Cash, Litecoin, ripple, and Ethereum. Just to give an example, one of the most "uncomfortable" aspects of the Bitcoin system is represented by the times required for a transfer which are relatively long, and many of the Altcoins were born to overcome this problem.

Ethereum (ETH)

In second place in order of trading volume, Ethereum stands out immediately after bitcoin, whose innovations compared to the original platform deserve particular attention. In fact, Ethereum is a blockchain platform which, however, allows users to take advantage of different decentralized applications not necessarily anchored to the mere exchange of cryptocurrencies. It is a network that allows all participants to benefit from an unalterable and shared archive of transactions regulated

[28] Chris Clark, Bitcoin Internals: A Technical Guide to Bitcoin, 31/06/2013.

[29] On this point, reference is made to Anonymous, *Che cosa sono le Altcoin e perché sono importanti*, maggio 2018, www.bitcoin.it.

through smart contracts[30]: from crowdfunding to the protection of intellectual property, or to the registration of a domain. In fact, they are smart contracts because they react to inputs by responding with consequential outputs.

To be able to use this tool on the platform, you must pay in ether currency (ETH). The ratio of this blockchain platform born in 2013 from the mind of Vitalik Buterin, lies in the desire to offer users unprecedented possibilities everywhere and forever. In fact, it was not designed to be blocked or censored and there is no maximum limit of ether that can be issued, unlike other cryptocurrencies[31]. However, the smart contract complains of technical-legal application limits: the irreversibility of the blockchain does not allow to modify an error, and it is a useful tool only the contractual conditions which are easy to translate into a computer language (e.g., if there is a deadline, make the payment).

Litecoin (LTC)

Litecoin was born on 7th October 2011 through the work of Charles Lee, a former Google employee. This virtual currency alternative to bitcoin was developed for two reasons: to support a greater number of transactions in less time and to increase the maximum number of units that can be issued[32], equal to 84 million – compared to the maximum limit of 21 million set for Bitcoin. Consequently, Litecoin is worth much less: as of 1st July 2021, 1 BTC is equivalent to € 28,068.78, while 1 LTC corresponds to "only" € 114.70[33]. This was actually the intent of the creator Charles Lee, who defined the two cryptocurrencies as gold and silver: two "assets" with the same function but with different value, because the one is "rarer" than the other.

Bitcoin Cash (BCH)

Bitcoin Cash is one of the market's newest cryptocurrencies. It was born in 2017 to overcome an important operational problem of Bitcoin, which is scalability, that is the ability to support as many transactions as possible per second, based on the size of the block. While Litecoin reduces processing time, with Bitcoin Cash what changes is the size of the block itself, eight times larger than a single block of Litecoin and Bitcoin[34] (on 1st July 2021 32MB). Although the maximum number of units that can be issued is the same as bitcoin – equal to 21 million – bitcoin cash is worth much less, even

[30] From internet https://www.ilsole24ore.com/art/smart-contract-cosa-sono-e-come-funzionano-clausole-blockchain-ACsDo2P.

[31] For further information, please refer to Bellini M., *Che cos'è e quali sono gli ambiti applicativi di Ethereum*, august 2019, www.blockchain4innovation.it.

[32] Data provided by the official litecoin.org website.

[33] Data available on www.coinmarketcap.com as of 1 July 2021.

[34] For further information, please refer to the official website www.bitcoincash.org.

though it ranks third in order of price, immediately after maker: 1 BCH is equivalent to € 415.94[35]. Despite its more than modest value, bitcoin cash is less traded than Litecoin, probably due to a less contained volatility: In the last 52 weeks, the maximum value reached by bitcoin cash is equal to € 1,635.15, against a minimum value of € 205.66 (data as of 1st July 2021). It is clear that, between the two, Litecoin is suitable for more risk averse investors.

Ripple (XRP)

Ripple's cryptocurrency is the latest to boast a trading volume of over one billion. Its innovative aspect deserves particular attention as the Ripple network was developed to provide a useful system for the direct transfer of assets in real time, offering a faster and cheaper alternative to current money transfer systems such as SWIFT[36], especially useful for banks and financial intermediaries. This is because, unlike the Bitcoin platform, on the Ripple network it is possible to exchange and transfer money in different currencies, without paying any fees. What users exchange are IOU credits (I Owe you), whose unit of measurement is the XRP cryptocurrency. The so-called gateways of the Ripple network will then convert the virtual currency into the currency in which the user must send or receive money[37]. Therefore, the ratio of this platform does not have a speculative purpose – as for Bitcoin – but is aimed at facilitating exchanges between multiple agents by reducing time and costs. Also, for this reason there is no maximum number of XRP units that can be issued: at the birth of the platform, 100 billion units were issued available to users who wish to use it as a transfer tool.

Dash (Digital) Coin (DASH)

Another very important alternative currency is the Dash, born in 2015 and initially called Xcoin. This is proposed as an alternative with the intention of improving the main characteristics of Bitcoins wherever possible and adding some new features. Dash coins are also considered as real digital cash as there is the possibility of accessing instant payments and this allows the transfer of digital money from anywhere in the world to different accounts in a matter of seconds. They also use a safer and faster open-source peer to peer network. Regarding the security of the Dash, preventive measures such as MasterNodes have been introduced. These are proof of service systems that protect and enhance the digital currency. The Dash still works on two levels: the aforementioned MasterNodes

[35] Data provided by coinmarketcap.com as of 1 July 2021.
[36] Acronym of Society for Worldwide Interbank Financial Telecommunications: system that allows money transfers through an international network.
[37] For further information, see Gagliarducci C., *Ripple: cos'è, come funziona e quali differenze con il Bitcoin?,* may 2017, www.money.it.

and traditional mining. This type of network is not subject to any kind of scrutiny by any government agency, so currency exchanges are completely unchecked, but that does not mean they are illegal or unsafe coins.[38] In any case, the owner of the account is unique, and no one can have access to his credentials. The growth that has seen this instrument in 2017 starting from a base of 11 dollars is sensational reaching, at the end of the same year, to be worth more than 350, recording a growth of about 3000%, with a market capitalization of more than 1.5. billions of dollars.[39]

Monero (XMR)

It is a cryptocurrency derived from Bitcoin – but completely decentralized – whose primary interest is the privacy of customers, also allowing anonymous transactions with "disposable" addresses. It is very scalable since it does not set fixed limits on the size of the blocks. Certainly, worth mentioning is Monero (XRM), created in 2014 with the name of BitMonero, deriving from Bit (as in Bitcoin) and Monero (translatable as "currency"), whose main feature is based on privacy. Other key issues are scalability and decentralization.[40] It is an opensource digital currency that uses the same proof-of-work system as Bitcoin, in order to avoid double spending and to secure the network. In January 2017, the Ring Confidential Transaction algorithm was then adopted, which strengthened privacy in favour of users. Compared to Bitcoin, Monero is programmed for fewer coins issued, that is 18.4 million, which will run out within eight years. Everything works through the ring signature algorithm, which has introduced an extra layer of security, precisely because it does not allow a third person to see the amount of a transaction. Monero is also characterized by its high scalability: it does not include a maximum size for the block size like Bitcoin which has a limit of 1 Mb. However, the system provides at the same time a penalty mechanism to avoid an excessive increase in blocks. Finally, as its use increases over time, the transaction fees will decrease, while on the other hand the commissions for total transactions will increase.[41]

[38] See Bellini M., "Cos'è *Dash (DASH),* valore di mercato e ambiti applicativi", www.*Blockchain4innovation.it*, 21 August 2018.
[39] See Provenzani F., "*Dash*: cos'è e come comprare la criptovaluta rivale del Bitcoin", www.*money.it*, 11 December 2017.
[40] See Torchiani G., "*Altcoin*: cosa sono e come funzionano le criptovalute sorelle del Bitcoin", www.*Digital4Trade.it*, 2 February 2018.
[41] See "*Monero* (XMR): cos'è e come fare *trading*", www.*tradingonline.com*, 2018.

1.4 Trading

Trading is at the centre and principle of the economy of a population in human history and its evolution is one of the most significant factors in the course of humanity. It is a process that has developed over time, in constant and continuous evolution. It is a way to bring people together for mutual benefits, consisting in the act of buying, selling, or exchanging items. Humans would not have evolved over the centuries if they had been confined to geographic boundaries. Trade inspired technological advancement, which led to faster and better communication between countries, creating a unified world for trade.

Here we will deal with the history, development and continuous expansion to this day of financial trading, on which the study of my thesis is based.

Short Story

From the late 1950s some stock exchanges began to comply, including the New York Stock Exchange or NYSE, with the aim of making equity trading operations more efficient: towards the end of this decade the US market boasted a trading volume of shares equal to 350,000 units a year from which about 100 million dollars a year were wasted because of errors due to the order systems[42].

In 1971, the National Association of Securities Dealers – an association of over the counter (OTC) traders founded in 1939 – created the first electronic stock market: the NASDAQ market, the first online platform for trading in OTC securities available exclusively for NASD associates. The National Association of Security Dealers for Automatic Quotation was the acronym for this site. In 1976, the New York Stock Exchange (NYSE) introduced its Designated Order Turnaround (DOT) system, which allowed brokers to place electronically orders for 100 shares directly to field specialists. These were not true electronic executions as the specialist still matched orders but bypassed the basic brokers. In Europe in 1977 was introduced the SWIFT, an electronic payment system aimed at the interbank market developed in Brussels but which already from the first months of its launch could count on a network of 293 banks from 15 countries worldwide. Then, in 1984, the NYSE adopted a more sophisticated SuperDOT system that allowed up to 100,000 shares to be placed directly on the order plan. Other floor brokers were excluded.[43]

[42] See: "Financial Trading and Investing", John L. Teall, 2018, Elsevier Science editor.

[43] See: David Greising, Laurie Morse: "Brokers, Bagmen, and Moles, Fraud and Corruption in the Chicago Futures Markets", 1991, Wiley, p. 160.

In 1987 e-commerce[44] took another leap forward as NASDAQ expanded the Small Order Execution System (SOES), which allowed brokers with small transactions to place their orders electronically rather than over the phone. This was done because many brokers just stopped answering their phones after the 1987 crash. The 19th October 1987[45] happened to be the day of the famous "black Monday", best known to be the most serious collapse of the markets in the entire world of economic history: the stock exchanges around the world – starting with that of Hong Kong – saw a collapse in a session that in some cases was nearly at 60%; the American Dow Jones index lost 508 points (equal to about 23%), the Milan Stock Exchange index – at the time the Italian Stock Exchange had not yet been established – lost 11% on Monday and 12% on Tuesday. According to a study led by the firm Amen & Associates, the largest US fund managers, compared to the data of January 1987, they lost approximately $ 500 billion or 40% of their assets under management. The rise of electronic commerce took place between 1990 and 1995. It is above all with the development of the World Wide Web that this phenomenon reached important levels and volumes; wanting to provide some numerical indication, in 1995 20% of the US population owned at least one financial instrument, in 1999 Forrester Research – a well-known American company for market research and surveys – estimated the existence of about 8.4 million trading accounts online open and operational in the US.

The current era

Trading is now regarded as an aggressive mode of capital market participation aimed at outperforming conventional buy-and-hold investments. Rather than trying to take advantage of long-term bullish trends in the markets, traders are leveraging short-term price movements to profit from both rising and falling markets. A trader can be a person or an entity who buys and sells instruments in the financial market: stocks, bonds, commodities, derivatives, mutual funds and so on. At the end of 2012, the size of the global stock market (total market capitalization) was approximately $ 55 trillion. By country, United States had the largest market (about 34%), followed by Japan (about 6%) and United Kingdom (about 6%). About 70% of US stocks in 2013 were automated trading. Algorithmic trading accounts for one-third of all Indian cash stock volume and nearly half of all derivatives activity.

[44] E-commerce: Transaction and exchange of goods and services carried out through the use of telecommunications and information technology, falling within the more general field of e-business

[45] The reasons behind the creation of the SOES: https://www.investopedia.com/terms/s/soes.asp.

Trading in today's financial markets generally takes place in what is known as a continuous double offer[46] with an order backlog open during a trade. It is an elegant way of saying that there are buyers and sellers who are put in touch to trade with each other. The stock market is responsible for their correspondence.

When developing trading algorithms, we need to choose the target to be optimized. The obvious answer would be profit, but it's not enough. You should also compare your trading strategy with benchmarks and compare its risks and volatility with other investments. The most basic metrics used by traders are Net profit and Loss (PnL), Sharpe ratio and Maximum reduction.[47]

Importance of historical data

"The stock market is human nature and the psychology of the crowd on display on a daily basis, plus the age-old law of supply and demand at work. As these two factors remain the same over time, it is remarkable but true that the graphic patterns are exactly the same today as they were 50 years ago, or 100 years ago. "- William O'Neil, How to Make Money in Stocks, 2019, Fourth edition.

A trader's decision-making process is aided by historical price charts. Historical data can reveal how a market has behaved to a range of various elements, ranging from regular economic cycles to unexpected international events. The data is incredibly valuable and there is no doubt about its importance to technical analysis[48], which allows someone to study and predict price changes on the financial markets through the history of price graphs and market statistics. The basic idea is that if a financial trader can identify market patterns in the past, then it is possible to make a fairly accurate forecast of price developments in the future. The historical returns are often analysed for trends or patterns that may align with current financial and economic conditions, hence the value of studying historical return trends.

Algorithmic trading[49] has been around for decades and, for the most part, has enjoyed a fair amount of success in its various forms. It began to be developed in the 1980s following the dematerialization of stock market orders and, nowadays, with the increasing computing power of computers, this type

[46] A double offer (sometimes referred to as a double auction) is a method of buying and selling products with many vendors and purchasers. The market institution receives offers from possible buyers and requests from potential sellers. The market institution sets a price p that clears the market: all sellers who asked for less than p sell, and all buyers who offer more than p buy. A common example of a double auction is the stock exchange.

[47] I will present the metrics in the third chapter where I will detail the Sharpe and the Calmar ratio.

[48] https://www.investopedia.com/terms/t/technicalanalysis.asp

[49] Algorithmic trading is also known as high frequency trading and, in financial jargon, as Algo Trading. It is an investment strategy based on an economic model and on computer software capable of taking a position on a market. for further information see: https://www.investopedia.com/terms/a/algorithmictrading.asp.

of trading is experiencing a real expansion. Note that it takes just 13 milliseconds to make a round trip via cable between the NASDAQ and Chicago stock exchanges and that over 70% of US stock exchanges use this form of trading. Traditionally, algorithmic trading has involved people painstakingly designing, optimizing and testing trading rules. While these rules have the benefit of being systematic and capable of operating at considerably higher speeds and frequencies than human traders, they are prone to selection biases and are unable to react to changing market circumstances.

For this reason, it is necessary to have a historical series of prices – and, when available, volumes – of the individual securities (indices, currencies, etc.) which are obtained by recording daily – and for the longest possible period – the prices of that particular security, reporting the passage of time on the x-axis and the price on the y-axis.

"*The past can also give useful indications for the future*". By analysing the time series graphs, it is possible to identify "patterns", or figures that tend to be resolved with greater probability in a precise direction, thus helping the analyst to formulate statistically founded predictions.

The goal is not to always to "guess" but to predict correctly 7 times out of 10, and in any case to provide operational indications to manage even the most difficult and dangerous market situations with a rational and disciplined method.

2. MACHINE LEARNING AND REINFORCEMENT LEARNING

2.1 Machine Learning: a brief introduction

Over the past decade, Machine Learning (ML) has proved to be a turning point for advanced analytics techniques. In an increasingly competitive corporate environment, ML is helping organizations to accelerate digital transformation and transition into an age of automation. To comprehend ML, first we must comprehend the fundamental principles of artificial intelligence (AI). We can describe AI as a software with cognitive abilities comparable to those of a person. One of the fundamental principles of artificial intelligence is to have computers that think like people that can be able to solve problems in the same way that human beings do. ML is dramatically different from AI because it can develop over time. Machine Learning algorithms are able to analyse enormous quantities of data and extract valuable information using a variety of programming approaches. With this approach, they may learn from the data they are given and improve on prior versions. One of the most essential components of Machine Learning algorithms is Big Data. By human standards, Big Data is time-consuming and difficult to analyse, yet good data is the greatest feed for training a Machine Learning system. The more clean, useful and machine-readable data a large dataset has, the more successful the Machine Learning algorithm's training will be. There are two types of data in ML: labelled data and unlabelled data. Labelled data contains both the input and output parameters in a machine-readable format, but labelling the data takes a lot of human effort in the first place. Only one or none of the parameters are machine-readable in unlabelled data. This eliminates the need for human labour but requires the development of more complicated solutions. As said before, Machine Learning algorithms may learn from its mistakes and improve. Algorithms for ML are currently learned in three ways: supervised learning, unsupervised learning and reinforcement learning.

Supervised Learning

Supervised Learning is the simplest form of ML and it is trained using labelled data. In this form of ML, a tiny training dataset is supplied to the Machine Learning algorithm. This training dataset is a subset of the larger dataset that serves to provide the algorithm a basic understanding of the problem and is fairly comparable to the final dataset. Then, the program establishes a cause-and-effect link between the variables in the dataset by finding correlations between the parameters provided. The algorithm understands how the data works and how the input and output are related at the conclusion of the training. Eventually, the final dataset is then used to train this solution, from which it learns in the same way as it did with the training dataset. This implies that Supervised Machine Learning

algorithms will improve even after they have been implemented, identifying new patterns and correlations as it learns fresh data. This form of ML is particularly effective for categorization tasks, such as detecting spam or predicting the next word in a phrase.

Unsupervised Learning

Unsupervised Learning has the benefit of being able to operate with unlabelled data: it means that no human intervention is required to make the dataset machine-readable, allowing the program to handle considerably larger data sets. While in Supervised Learning the labels help the algorithm to figure out the specific nature of any link between two data points, in Unsupervised Learning there is a lack of labels against which to work, leading in the formation of hidden structures. No human input is required for the software to perceive correlations between data points in an abstract manner. Unsupervised Learning algorithms' versatility is due to the development of these hidden structures, and it can adapt to the data by constantly altering hidden structures instead of a stated and fixed problem statement. Compared to Supervised Learning methods, this allows for more post-deployment development.

Reinforcement Learning

Reinforcement Learning (RL) is based on how people learn from data in their daily lives. It incorporates a self-improving trial-and-error algorithm that learns from diverse settings. RL uses the psychology concept of conditioning to place the algorithm in a work setting with an interpreter and a reward system. The output result of each iteration of the algorithm is presented to the interpreter, who assesses whether the outcome is favourable or not. If the program finds the proper answer, the interpreter reinforces it by rewarding the algorithm. Instead, if the results are unfavourable, the algorithm is forced to repeat the process until it finds a better result. The incentive system is directly proportionate to the effectiveness of the outcome in the great majority of cases. The solution is not an absolute value in typical RL use-cases, such as finding the shortest route between two points on a map. Instead, it is assigned an efficacy score, which is expressed as a percentage number. The higher this percentage number is, the more the algorithm gets rewarded. As a result, the software was programmed to provide the greatest available solution in exchange for the best potential reward. This type of ML is useful for complex tasks like playing games, where the rules of each game are constantly changing and need to be learned by trial and error.

Each one of these types of ML algorithms have a different purpose and a different set of trade-offs between the time required to compute an operation and the accuracy of the results. For example, more instances are used to train a filter to make it more accurate, but this also requires more

computation power. In that sense Supervised Learning algorithms are better suited for large datasets, whereas Reinforcement Learning is good for complex tasks that cannot be handled by pre-existing methods.

In this chapter we will analyse models to describe sequential decision processes and related resolution methods based on RL. In Sequential Decision Processes a sequence of actions is performed to achieve certain objectives and the same actions influence the rewards and the states, both immediate and future. There are processes used in the Markov Decision Processes which are the fulcrum for the formalization of RL problems. In the first instance, we will specify what is a sequential decision process and the main decision models; then, we will analyse the fundamental problem of RL by defining the functions and the main algorithms that have been developed to solve it.

2.2 Sequential Decision Processes

A Sequential Decision Process is a process that evolves in discrete time frames defined on two sets:

- a set S representing the set of states in which can be found the process;
- a set A that describes the space of the actions that can be performed within the process.

Below there is an example of how a sequential decision process develops.

Environment:	You are in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 7 units. You are now in state 15. You have 2 possible actions.
Agent:	I'll take action 1.
Environment:	You received a reinforcement of -4 units. You are now in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 5 units. You are now in state 44. You have 5 possible actions.
⋮	⋮

Figure 2: an intuitive way to see the evolution of a sequential decision process. [Neural Networks chp. 7 SlidePlayer]

Furthermore, on these two sets is defined a function R called the *reward function*, which belongs to a decision maker process. The process itself consists of a succession of decisions, which are taken at each evaluation instant t = 1, 2, ..., T; the time interval between one decision instant and another is called *epoch*. At each epoch is made a decision and, as a response to this decision, are returned both a value representing the instantaneous gain of the decision maker and the state in which the system has passed. This process is very intuitively shown in Figure 2.

The sets S and A can be discrete or continuous. We will say that the decision process is to:

- finite horizon: if T ≠ ∞. We will denote this type of situation with **FH**.
- infinite horizon: if T = ∞. We will indicate this type of situation with **IH**.

Now we will focus on specifying models, functions and methods, in order to find excellent or approximate strategies – which we will formalize better later – which will allow us to reach, starting from one state, another state called *objective state*.

2.2.1 Markov Decision Process

Regarding the state, is meant that it contains all the information at the agent's disposal and that it can only be influenced by his actions. A state is preferable when it summarizes all past actions and highlights the most relevant ones, but often the environment does not provide the agent with all this information. The signal deriving from the state allows the agent to choose the most appropriate action for the given instant. This signal possesses the Markovian property[50] and an RL approach that satisfies this property is called a Markov Decision Process (MDP).

It is not important to know exactly how the environment is made, but rather to make general hypotheses about the properties that the environment possesses. In RL it is usually assumed that the environment can be described by a Markov Decision Process (MDP).

A Markov Decision Process[51] is defined by 5-pla $(S, A, p_t(\cdot\,|s,a), \gamma, r_t(s,a))$ where:

- S is the state space; at each time t the state of the environment is an element s Є S;
- A is the space of actions; at each time t the agent chooses an action a Є A to perform;
- $p_t(\cdot\,|s,a)$ is the transition probability function; a state transition model describing changes in the state of the environment after the agent performs an action a in the current state s;
- γ ∈ (0, 1][52] is the discount factor; a discount factor that controls the importance of future rewards;[53]
- $r_t(s,a)$ is the reward function at time t. A reward model describing the reward value the agent receives from the environment after performing an action; it depends on the current state s and the action a performed.

[50] Richard S. Sutton, Andrew G. Barto: Reinforcement Learning: An Introduction. London: MIT Press, 2018, p.48,49.

[51] Martin L. Puterman. Markov Decision Processes: Discrete Stochastic Dynamic Programming. New York, NY: John WileySons, 1994, 649.

[52] It makes sense to consider gamma = 1 if the process is FH.

[53] From Sutton, Barto, p.55: *"The discount rate γ determines the present value of future rewards: a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately."*

The MDP model is associated with a decision maker, who can influence the behaviour of the stochastic system that evolves over time. Its goal is to choose a sequence of actions that lead the system to perform optimally compared to predetermined performance criteria. This sequence of actions is called policy – later in the chapter we will formalize this concept better. In the environment the system is stochastically evolving, passing from state s_t to state s_{t+1}. However, the a priori decision maker cannot predict the future state, but for the hypotheses we have made it is able to detect the state S of the system. Trying to solve an MDP model means finding an optimal policy.[54]

The MDP is called Markovian if the probability of passing from state s to state s' depends only on the current state and not on the history of the process; less formally, the fact of arriving in a new state s depends only on the current state and not on how we are arrived in this state.

As mentioned above, when the agent performs an action it affects the environment: at instant t, the agent perceives the environment as $s_t \in$ S. Based on s_t decides to act with $a_t \in$ A and the environment responds by giving the agent an immediate reward $r_{t+1} = r_t\ (s_t\ ;\ a_t)$ and producing the next state s_{t+1} = s'$|S_t = s$.

To complete the introduction to MDP models, we need to talk in more detail about both the reward function and the transition probability function. According to the MDP model, after choosing the action a ∈ A(s_t) in state s at decision time t:

1. the decision maker receives a reward r_t (s, a) and
2. the state of the system in the next decision epoch is determined by the probability distribution $p_{t+1}(\cdot\,|s, a)$.

If the function r_{t+1} (s, a) is positive, it can be seen as a gain; vice versa if it is negative, it can be seen as a cost. From point of view of the model, it is appropriate to specify how the reward is elaborated during a period. The value of the function r_t (s, a) – or the expected value – must be known before choosing an action, and this value must not depend on future actions, but on the information available now. When the reward depends on the state s_t, we denote by r_{t+1} (s, a, j) the value of the reward received at time t+1, which indicates that the state of the system at the decision epoch t+1 is s_{t+1}, with the chosen action a ∈ A (s) and the system occupies state j at the decision time t + 1.

[54] The existence and uniqueness of the solution is guaranteed, for example, if the state space is finite. More generally, the solution does not necessarily exist and, if it exists, it does not necessarily mean that it is unique.

At each decision instant, based on the rewards accumulated in the various states between the current decision instant t and the future decision instant t + 1, the decision maker calculates the expected value conditional on the transition probability to estimate which will be your average future earnings:

$$r_t(s,a) = \sum_{j \in S} r_t(s,a,j) p_t(j|s,a)$$

where the non-negative function $p_t(j|s,a)$ denotes the probability that the system is in the state j ∈ S at instant t, when the decider chooses the action a ∈ A(s) at instant t.

We will further formalize how to solve the problem of finding an optimal policy from an analytic point of view, both for the FH case and for the IH case when we use this model in the context of RL. For the moment, we limit ourselves on providing an example of MDP, to make some more intuitive than theoretical considerations on MDP models.

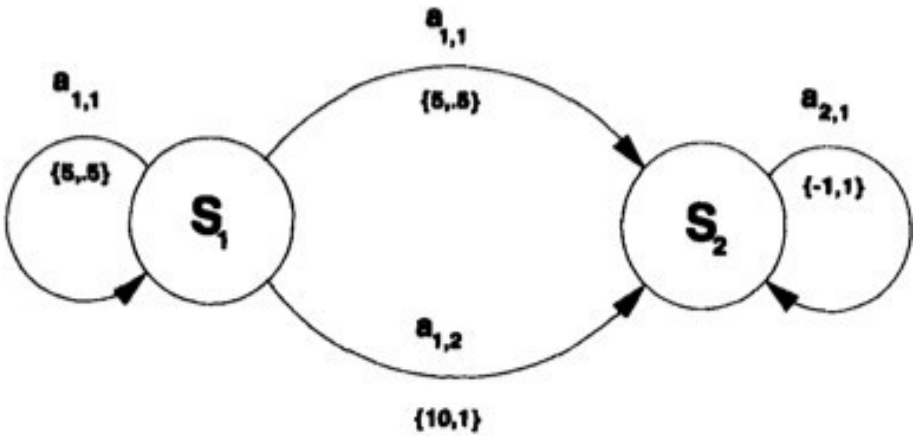


Figure 3: Example of an MDP model. Source: Sutton, Barto, Reinforcement Learning: an introduction

As we can see in Figure 3, there are two states s_1 and s_2 . In state s_1 there are two possible actions $a_{1,1}$ and $a_{1,2}$, while in state s_2 there is only one possible action $a_{2,1}$. In the curly brackets, the first term represents the immediate reward that the decision maker receives, while the second term represents the probability that the system will transit into the new state. So, for example, if we choose action $a_{1,1}$, the process will pass from state s_1 to state s_2 with probability 0.5, providing to the decision maker a reward of 5 units, or it will remain in state s_1 with a probability of 0.5, providing to the decision maker a reward of 5 units. Otherwise, if we choose action $a_{1,2}$, then the process will transit to state s_2 with probability 1, giving the decision maker a reward of 10 units. However, if the environment is in state s_2, the process will always remain in s_2 and it will always receive a negative reward equal to -1. For example, assuming that the decision maker makes greedy choices, at the first decision instant

will be performed the action $a_{1,2}$ to receive immediately a reward of 10 units. A choice that, immediately, offers a good return but, in the long run, does not lead to good results. Let us figure out why: since the decision maker after action $a_{1,2}$ passes into state s_2 and, due to the way the model is made, is "forced" to remain in this state; at each subsequent decision instant, the reward that the decision maker has accumulated will decrease by 1 unit. So, after 10 moments of decision, the decision maker will nullify the "effort" he made to pass into state s_2. This very simple example already suggests that it is not a simple problem to choose a sequence of actions, because an apparently convenient action in the moment of decision may/could not prove/be useful in the future.

For completeness, we can finally mention continuous MDPs, in which the decision instant is no longer at predetermined discrete instants but, the next choice instant t_1 can be found in any point of the time axis starting from a predetermined instant t_0. The time spent in a state by the process can be described by an exponential random variable.

In the next paragraph we will describe the basic concepts of reinforcement learning, in which we will define methods and algorithms to solve the models we saw in the previous paragraphs in exact and approximate form.

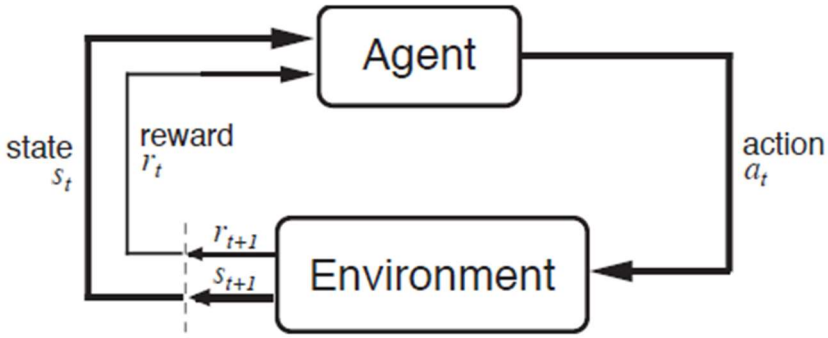


Figure 4: Interaction agent-environment. Source: Sutton, Barto, Reinforcement Learning: an introduction

2.3 Basic concepts

In this section we will face the basic concepts to deal with the study of RL-based algorithms, in order to solve sequential decision problems.

2.3.1 The Agent-Environment model

The task of RL consists in learning through interaction. The entity that makes the decisions is the Agent while everything that is outside of it, and with which it can interact, is called Environment. The Environment my itself be real or simulated: in the first case the Agent learns by interacting directly

with the target Environment; in the second case the Agent learns by interacting with a model that simulates the operation of the target Environment. The evolution of the Agent-Environment system occurs at discrete time intervals t = 0, 1, 2, ...; in every amount of time the Agent receives status information about the Environment, which we indicate with $s^t \in S$, and on the basis of this information decides to carry out the action $a_t \in A(s_t)$. In the next step, the Agent receives a reward $r_{t+1} \in R$ and the new state s_{t+1}, as can be inferred from figure 4.

A policy $\pi: S \rightarrow A$ is a function that associates states to actions. A policy can be of two types:

- Deterministic: returns the action that must be performed in state s. It takes the form $\pi(s)$;
- Stochastic: returns the probability of performing the action a, knowing that it is in the state s; it is indicated by $\pi(s, a)$.

The methods of RL specify how the Agent changes its policy starting from the experience it acquires in the Environment. We have not yet clearly specified what the learning objectives are in RL-type scenarios. To do this, we deepen the analysis of the **reward function**. Basically, the **reward function** is a profit / cost that you want to maximize / minimize in the long term by the Agent. In fact, the reward is calculated independently from the agent, and can be thought of as a reward or a punishment that the Agent receives when, once the actions have been selected, he induces a change in the environment that responds to him by presenting new situations. More specifically, the Agent interacts with the Environment at discrete times, providing an Action and receiving a Reward and a new State on which to process the new Action, thus generating a cycle of States – Actions – Rewards – State – Actions – Reward and so on.

The problems that we will face up to the end of the chapter are essentially two:

- The transition probabilities are unknown to us.
- Learning is onerous from a computational point of view.

However, we can note that there are cases in which, even knowing the system perfectly, the RL process is onerous.

Returning to what the learning objective is, it can be expressed in terms of reward. All the algorithms act to maximize the expected earnings, that is all the rewards that the Agent expects to receive in the long term.

If our process is FH, then the expected gain starting from the instant t ≤ T will be:

$$R_t = r_{t+1} + r_{t+2} + ... + r_T$$

where T indicates the period.

As for the learning processes, they are divided into two categories:

- **Episodic**: learning is elaborated by episodes, or the Agent-Environment interactions are by nature divided into periods. Is defined a final state T in which the sequence must be concluded, and the agent chooses the first action to optimize T steps, the second to optimize the subsequent T-1 steps, the third to optimize the T-2 steps and so on;
- **Continuous**: there is no final state for learning; the interactions are not divided into episodes but are continuous and limitless.

In the case of infinite horizon learning processes, since T = ∞, unless the succession of rewards is limited – which is quite rare – the expected gain could be infinite and, therefore, the process could not converge. Remembering the discount factor γ that we introduced in the various models, being 0 < γ ≤ 1, we can define the expected gain as:

$$R_{t_\gamma} = \sum_{k=0}^{+\infty} \gamma^k r_{t+k+1}$$

For what has been previously said, we observe that the conditions γ = 1 and T = ∞ should not occur simultaneously.

Now we can express the main reason why γ is inserted. In fact, we can think that the discount factor represents how much the agent gives importance to what it will earn in the future: if γ = 0, then the agent is substantially "myopic", in the sense that it is concerned with maximizing its gain immediately; instead, in the case of γ → 1, then the Agent "will look far away", that is, it will deal with long-term profit as well as current profit. The discount rate determines the actual value of future rewards: a reward received k moments from now is only γ^{k-1} times as valuable as if it had been received immediately.

In the next section we will deal with introducing functions defined on the space of states and actions, in order to outline methods for solving the problem.

2.3.2 Estimation of the optimality of the policies

After introducing the problem a little, we can talk about the functions that are used to find the optimal policy.

RL methods are based on the estimation of two functions. They are alternatives to each other, so it can be used only one at a time. The two functions are:

- V(s) called value function or also V-function, which indicates, on average, how much benefit it brings to the agent being in state s – state evaluation function that estimate the average return that can be expected;
- Q (s, a) called action-value function or also Q-function, which indicates, on average, how much benefit it brings to the agent to perform action a in state s – evaluation function of an action performed in a state.

Basically, V(s) and Q (s, a) are functions that depend on policies. In fact, once a policy π has been set, the value V^{π}(s) indicates the expected gain, starting from state s and following the policy π. The formula is:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\{\sum_{k=0}^{+\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

where $E_{\pi}\{\}$ denotes the expected value, given that the Agent follows the π policy.

A fundamental property of value functions is that they satisfy particular recursive relations. The following condition exists between the value of s and the value of its probable successors for any policy and any state s:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\left\{R_t \middle| s_t = s\right\} \\ &= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\} \\ &= E_{\pi}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s\right\} \\ &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_{t+1} = s'\right\}\right] \\ &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right], \end{aligned}$$

where the actions a ∈ A(s) and the states s' ∈ S. Called *Bellman equation* for V^{π}, it describes the link between a state's value and its successor states' values; the expected return of the next state, plus the expected reward between the two states, must equal the value of a state s.

The value function V^{π} is the only solution of the related Bellman equation. For this reason, equation is the cornerstone of several algorithms that calculate and learn V^{π}.

The function V^{π}(s) is called the *state-value function* for the policy π. Similarly, having set a policy π, we can define the value of Q^{π} (s, a), that is the value associated with carrying out action a, in state s, following the policy π as:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{+\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

Remembering that the resolution of an RL problem entails determining that criterion π that allows you to perform best, now you have all the parts needed to introduce the optimization of value functions. Our goal is to find a policy π^* which, assuming the state s as the starting state, give us a maximum expected gain. As a matter of principle, the optimal policy is not necessarily unique – this is true for MDP with finite state space.

These functions have the property of defining a partial ordering between policies, that is:

$$\pi \geq \pi' \iff V^{\pi}(s) \geq V^{\pi'}(s)$$

The optimal policy π^* will be such that ∀s ∈ S:

$$V^{\pi^*}(s) \geq V^{\pi}(s) \quad \forall \pi \in \Pi$$

So, the optimal V function will be:

$$V^{\pi^*}(s) = max_{\pi \in \Pi} V^{\pi}(s) \quad \forall s \in S$$

Similarly, the optimal Q function will be:

$$Q^{\pi^*}(s,a) = max_{\pi \in \Pi} Q^{\pi}(s,a) \quad \forall s \in S \ \ \forall a \in A(s)$$

Furthermore, the functions V^{π^*} (s) and Q^{π^*} (s, a) are related to each other. In fact:

$$Q^{\pi^*}(s,a) = E_\pi\{r_{t+1} + \gamma V^{\pi^*}(s_{t+1}) | s_t = s, a_t = a\}$$
$$V^{\pi^*}(s) = max_{a \in A(s)} Q^{\pi^*}(s,a)$$

These equations are **Bellman's optimality equations**.

Therefore, the problem of RL translates into learning from experience the expressions of V^{π^*} (s) and Q^{π^*} (s, a). Obviously, this is true if we know the function describing the transition probability, that is when we know the **dynamic** of the system. Once we know the dynamic, then it is relatively easy to derive the optimal policy.

Indeed:

$$\pi^* = argmax_{a \in A(s)} \sum_{s' \in S} P^a_{s,s'} V^{\pi^*}(s)$$

The value of a state with an optimal policy must be equal to the expected value of the rewards, obtained with the best possible action in that state; because of each action, there will be one or more actions that will satisfy Bellman's optimality condition. One result to highlight is the following: with respect to optimal V-function, any policy that is greedy is an optimal policy; *greedy* since it selects the action that leads to the state with the greatest value.

In other words:

$$\pi^* \; greedy \; with \; respect \; to \; V^{\pi^*}(\mathrm{s}) \; \rightarrow \; \pi^* \; optimal \; in \; the \; long \; term.$$

However, in many concrete cases the optimal functions are difficult to find: one has to be content with approximations. The better the approximation, the better the resulting policy will be. In the next paragraph we will make some comments on how to behave in case the dynamics of the system is not fully known.

2.3.3 Optimal Value Function

Finding a policy that accumulates a lot of rewards in the long run, is basically what resolving an instance of RL entails. If the expected return on each state is greater than or equal to that of π, a policy is said to be superior or comparable to that of π. In other words, π > π' if and only if V^{π}(s) ≥ ≥ $V^{\pi'}$(s) for every s ∈ S. There is always a policy that is better or equivalent than all the others that is called *optimal policy*. Since there may be more than one, we denote all optimal policies with π^*.

These share the same value function, called *optimal value function*, V^*, equal to:

$$V^*(s) = \max_{\pi} V^{\pi}(s),$$

for every s ∈ S. Obviously, since V^* is the value function of a policy, it must satisfy the Bellman equation. Being optimal, the relative Bellman equation can be written without referring to a specific policy. For V^*, this equation is called the optimal Bellman equation:

$$V^*(s) = \max_{a} \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right]$$

For a finite MDP, the Bellman equation has a single solution. Indeed, this equation is a system of N equations in N unknowns, one for each state, where N = |S| is the number of possible states of the environment. If we know the dynamic of the system ($P_{ss'}{}^a$ and $R_{ss'}{}^a$), then we can solve the system of equations for V^*, using any method for solving systems of nonlinear equations. Once V^* has been calculated, finding the optimal policy is relatively simple. There will be one or more actions for each state s that achieve the maximum in Bellman's optimality equations. An optimum policy is one that gives a non-zero probability to only these actions. Unfortunately, in most cases an Agent can determine the optimal policy only with extremely high computational costs. The critical aspects of the problem are: the amount of computation required at each single step and the amount of memory required to obtain approximations of the value functions, policies, and model of the environment itself. For this reason, especially for models of an environment with a very high number of states, reinforcement learning algorithms aim to obtain an optimal policy approximation.
However, the dynamic nature of this learning paradigm makes it possible to obtain approximate policies such as to achieve great accuracy with a big number of states, at the cost of obtaining less accuracy for rarely visited states.

2.3.4 Exploration vs Exploitation

It often happens that, even with a good representation of the state and a sufficient number of states for learning, the Agent reaches sub-optimal solutions, as it has not searched in the space of states with sufficient depth. In the absence of a model, one of the main problems we have to face is to derive the function of the transition probabilities, expressed in numerical form.

In reality, when the agent does not know the model, it must balance the trade-off between:

- Exploration: consists in the exploration of the space of states which it has not experimented before so as to evaluate the results;
- Exploitation: consists of the enhancement of the action chosen by the agent in a certain state; in other words, the agent passing through that state will choose that action with a high probability.

From the given definitions, it is clear that the two operations contrast each other. In fact, increasing the number of iterations, one prefers to explore starting from state s, rather than taking a path already done, implying a smaller number of iterations in which the agent will choose the same action a in state s. Everything that we will say later is aimed at finding the right compromise between the two operations, to provide adequately accurate solutions, without having impractical execution times. Now, let us suppose that we have obtained some estimates for our functions. in particular we consider Qt (s, a). The question we want to answer is whether it is always better to follow a greedy-type[55] procedure, that is to choose for each s ∈ S:

$$a^* = argmax_a Q^*(s, a)$$

The answer to this question is negative, in the sense that the Agent must also be allowed to explore other areas of the space of states.
The strategies for choosing the actions that are most used in practice are two:

- *ϵ-greedy*: it is considered that in the Environment the Agent does not always act with a greedy policy, so there will be a small probability ϵ that the agent chooses another action, an action chosen following a uniform distribution;
- *Soft-Max*: the problem with a ϵ-greedy choice strategy is that the actions that are chosen with probability ϵ are all equally probable. In other words, you have the same probability of choosing the best action – after the greedy one – and the worst of all. Therefore, in many cases it is preferred to use a distribution that also considers the values assumed by the shares. This distribution must be able to give more importance to actions that have a high value, and less importance to actions that have a low value.

[55] Already defined in paragraph 2.3.2.

The most used distribution is that of Gibbs:

$$p(s, a, \tau) = \frac{e^{Q_t(s,a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(s,b)/\tau}}$$

where n is the number of actions and τ is a non-negative parameter called temperature. If $\tau \rightarrow 0$, the temperature is defined as "low"; in this case the distribution will tend to select actions with higher values rather than actions with lower values. Vice versa, if $\tau \rightarrow +\infty$, the temperature is defined as "high" and the distribution will choose the actions independently of their value, following almost the uniform distribution.

In the next section we will address the fundamental RL algorithms, which are the basis of the advanced approaches, in which there are the algorithms that were used in the agent project; we will analyse this better in chapter three.

2.4 Fundamental algorithms

RL algorithms can belong to one of these two classes:

- **Model free**: if the algorithm can make the agent learn the VF (value functions) without needing to have any kind of environment model, because they primarily rely on learning;
- **Model based**: if the algorithm makes the agent learn the model and, because of this knowledge, the agent modifies the VF values, relying, as their primary component, on planning.

The algorithms – which we will describe in this section – refer to three methods, each of which has its own strengths and weaknesses.

The methods we refer to are:

- **Dynamic programming**: it is well defined from a mathematical point of view, but it needs an accurate model of the environment, which we have said is not always possible;
- **Monte Carlo**: they do not need a model of the environment and are conceptually simple. These methods estimate value functions through experience, by means of samples deriving from actual or simulated interaction with an environment;
- **TD (Temporal Difference)**: they do not need a model of the environment and are incremental; the only drawback is that they are computationally expensive and difficult to analyse.

In the next section we will deal with describing the fundamental algorithms that are the basis of the advanced approaches that we will analyse in the last section.

2.4.1 Dynamic Programming

With the expression Dynamic Programming (from now on DP) we refer to a set of techniques and of algorithms which can be used to calculate optimal policies, after knowing the environment model – formulated for example as MDP.

The basic idea is to use the value function to organize the search for good policies. To do this, the Bellman equations, associated with each state, are cyclically solved in order to obtain more and more accurate approximations of the value functions. Basically, these algorithms have two weaknesses: first, they need to know completely the dynamics of the system in terms of transition probability between states, which is not always possible; secondly, they have a high computational cost. However, they form the essential theoretical foundation upon which all the RL algorithms have been built and, consequently, represent a fundamental subject of study. In fact, all the methods we will analyse are attempts to obtain the similar result as DP, only with fewer calculations and assuming the lack of a model.

In this paragraph we assume the knowledge of the model of the environment. In this case it is an MDP with a finite state and action space. In fact, under these hypotheses, it is shown that the DP methods converge to the optimum. In the absence of these hypotheses, the convergence of such methods is possible only in rare cases. The fundamental idea of dynamic programming – as well as RL in general – is the use of the values of states and actions to seek good policies.

Evaluation of a policy: now we consider how to compute the function $V^{\pi}(s)$ given a policy π and a starting state s. This problem is also known as the *prediction problem* or as the *policy evaluation problem*. In case the dynamics of the system is completely known, it is a matter of solving a system of linear equations of |S| equations in |S| unknowns – |S| is the cardinality of the state space.

In general, the solution is a simple – albeit tedious – calculation. For our purposes, are more appropriate iterative solution methods:

$$V_{k+1}(s) = \sum_{a \in A(s)} \pi(s,a) \sum_{s' \in S} P^{a}_{s,s'} [R^{a}_{s,s'} + \gamma V_k(s')]$$

This is done ∀s ∈ S.

V_k is a fixed point for this rule because, in this situation, the Bellman equation guarantees equality. For k → ∞, V_k converges to the optimum under the same conditions that guarantee the existence of the value function. V_{k+1} is the successive approximation from V_k, of each V_{k+i} approximations. Each state s is subjected to the same policy evaluation procedure: the previous value of s is replaced with a new value derived from the old values of the successor states of s, and the projected immediate rewards are appraised.

Therefore, the evaluation algorithm of a policy consists in applying an iterative method to the solution of the Bellman optimality equation. Since convergence is guaranteed to us only for k → ∞, anyway we must be satisfied with having good approximations by imposing a stopping condition $|V_{k+1}(s) - V_k(s)| < \eta$ with η sufficiently small.

In the next section, we will see if would it better or worse to change the current policy that we are following.

Improvement of a policy: a reason to calculate the value function for a policy is to help find better policies. Let us suppose we obtained the value function V_π for an arbitrary deterministic policy, and for a state s we want to find out if we should change the policy to choose deterministically an action $a \neq \pi(s)$. It should be noted that the policy we are considering is deterministic. We can make similar considerations in case the policy is *stochastic*.

One way to answer this question is to calculate the value of performing action a in state s:

$$Q^\pi(s,a) = \sum_{s' \in S} P^a_{s,s'}[R^a_{s,s'} + \gamma V(s')]$$

In fact, if $Q^\pi(s, a)$ is greater than $V^\pi(s)$, it will be convenient to execute action a each time we encounter the state s. The converse happens in the opposite case. This result is a particular case of a more general one known as the policy improvement theorem[56].

If π and π'are a generic pair of policies, such that ∀s ∈ S, and if

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s)$$

then the policy π' will be better or at most equivalent to π.

[56] *This result is the basis for demonstrating the convergence of all the methods that we will study, in many cases I will not provide proofs.*

In formulas:

$$Q^{\pi}(s,\pi'(s)) \geq V^{\pi}(s) \;\; \forall s \in S \Longrightarrow V^{\pi'}(s) \geq V^{\pi}(s) \forall s \in S$$

From this theorem a consideration derives following which, if we choose in each state the action that seems to us the best, is the one that maximizes Q^{π}(s, a): we will obtain the best policy derivable from the knowledge of Q^{π}(s, a).

The new policy π 'will be calculated as follows:

$$\pi'(s) = argmax_a Q^{\pi}(s,a) = argmax_a \sum_{s' \in S} P^a_{s,s'}[R^a_{s,s'} + \gamma V^{\pi}(s)]$$

What we did was to find a policy that is greedy with respect to the policy that we have calculated before. This procedure is called policy improvement.

The case where π ≡ π' ⇒ V^{π} = $V^{\pi'}$ ⇒ V satisfies Bellman's optimality equation.

As mentioned above, similar criteria can be used in the case of stochastic politics.

Now that we have improved policy π and, using V^{π}, we have obtained $V^{\pi'}$, we can calculate the value function of the improved policy π', with which we will then reapply the improvement procedure. This process is described below.

Policy Iteration: at this point we can observe that if we combine both methods described above, we can obtain an algorithm in which:

- V converges towards its optimal value V^*.
- π ⟶ π ∗.

In fact, suppose we start from a fixed policy π. Therefore, we can first evaluate the policy and then improve it. What will result from this process is to obtain a better policy than the initial one. If we repeat this process until is reached the optimal value, we obtain the so-called policy iteration algorithm.

The steps to take to get the increasing improvement sequence are:

- establish a starting policy π_0;
- to calculate:

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} V^{\pi_2} \xrightarrow{I} \dots \xrightarrow{I} \pi_* \xrightarrow{E} V^{\pi_*}$$

where E denotes the policy evaluation and I denotes the policy improvement.
Another algorithm belonging to dynamic programming is called Value Iteration.

Value Iteration: a disadvantage of the policy iteration algorithm is that in each step we must evaluate a policy. This involves an iterative procedure whose convergence time we do not know in advance, as it will rely on a variety of factors, including how was chosen the starting policy.
One way to overcome this drawback is to cut off the evaluation of the policy at a certain step. This operation does not hinder the guarantee of convergence of the algorithm to the optimal value. A particular case in which the evaluation of the policy is blocked at a backup[57] per state – also called sweep[58] – allows the use of the value iteration algorithm. For completeness we can write the equation that describes what the algorithm does:

$$V_{k+1}(s) = max_a E\{r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s, a_t = a\} =$$
$$= max_a \sum_{s' \in S} P^a_{s,s'}[R^a_{s,s'} + \gamma V_k(s_{t+1})]$$

This is done ∀s ∈ S.
The algorithm ends when the value of V_{k+1} differs from V_k by a sufficiently small amount in a sweep. Therefore, value iteration combines a policy evaluation sweep and a policy improvement sweep. Clearly, we can also think of increasing the number of sweeps: this ensures a faster convergence but a greater computational burden.
In the next paragraph we will see how to use together the two iterative processes previously exposed.

[57] *By backup I mean the process of evaluating V or Q in a state prior to the one given.*
[58] *To service debt with all cash flow now available. Liquidity issues may arise as a result of this.*

Generalized Policy Iteration: the algorithms that we described in these last paragraphs are essentially based on two processes that occur in parallel, which are policy evaluation and policy improvement. The repeated execution of these two processes converges the general process towards the optimal solution, as shown in Figure 2.4.

The idea of making these two processes interact goes under the name of *Generalized Policy Iteration*. Thanks to this algorithm, it is possible to ensure that the two processes act on the same objective and not on two different objectives V and π, that is alternating policy evaluation and policy improvement. Moving directly towards one goal causes a departure from the other. So, the two processes are both in competition and in cooperation. Choosing a policy that is greedy with respect to the function, generally makes the function V inconsistent with respect to the π policy. On the contrary, the π policy is typically non-greedy with respect to the V^{π} function. Despite these problems, the long-run process converges to the optimal value of V and π as summarized in Figure 6. The last thing we want to highlight is a special property of the DP methods. All update estimates in a state are based on estimates of updates made in its successor's states. This property is known as **bootstrapping**.[59]

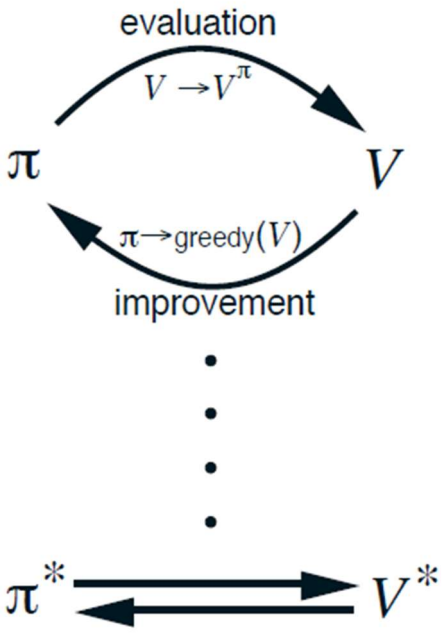


Figure 5: Generalized Policy Iteration process. Source: Sutton, Barto, Reinforcement Learning: an introduction

[59] *V.* https://www.investopedia.com/terms/b/bootstrap.asp.

2.4.2 Monte Carlo methods

In this section we consider the learning methods that we outlined above, for estimating value functions and discovering optimal policies. Unlike the previous paragraph, here is not assumed a complete knowledge of the environment. The types of learning that can be done are based on two types of data, respectively from:

- Simulated experience;
- Experience on the real system.

Generally, it is not always possible to have a priori the distribution of the samples generated by the environment, but it is possible to generate samples that are in line with this distribution to use them, to make learning. The Monte Carlo (MC) method estimates the value functions and optimal policies through experience – i.e., by means of samples. This method interacts directly with the environment, obtaining a sample of states, actions and rewards with which it numerically calculates the value functions, since is not assumed a known model. These samples are generated by means of simulations of the relationship between the agent and the environment. One thing that MC methods have in common with DP methods is how they interact with functions to evaluate the value of states and actions to obtain optimality.

Monte Carlo methods try to achieve optimal policy by averaging sample returns. Therefore, an environment model capable of generating these example transitions is sufficient. Unlike with dynamic programming, it is not necessary to know the likelihood of all possible transitions. In many circumstances, it is simple to generate samples that fulfil the desired probability distributions, but expressing the whole of the probability distributions is impractical.

A feature of MC methods – as explained in detail below – is that the process is based on the average of the rewards of the samples.

MC Policy Evaluation: now we want to use Monte Carlo methods to learn the V-function given a certain policy that we set a priori.

Let us suppose that we want to estimate $V_\pi(s)$ given a set of episodes obtained following the policy π. Whenever we pass through state s during an episode, we will say that we have visited state s.

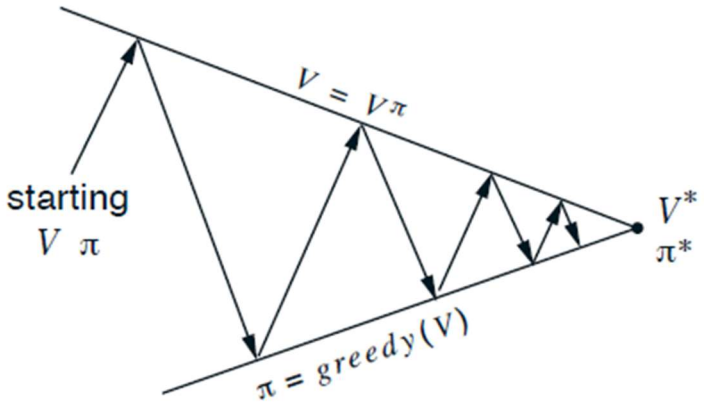


Figure 6: co-convergence, value function and politics. Source: Sutton, Barto, Reinforcement Learning: an introduction

At this point, we divide the MC algorithms into two types according to the visit method, because in the MC method only the transactions from one starting state to the next that are visited during the episode are represented:

- **Every-visit MC**: the estimate of V^{π}(s) is made as the average of the rewards from all transitions until the end of the episode;
- **First-visit MC**: only the rewards from a single state-to-state transition are averaged.

As the number of samples increase, both methods converge to V^{π}(s). An important feature of MC methods is that the estimates for each state are independent. In other words, the estimate we make of V or Q in one state is not based on estimates in other states, as was the case in DP methods. For this reason, we say that MC methods do not have the property of bootstrapping.
Another benefit is that the cost of computing the value of a single state is unaffected by the number of states.[60]

MC Estimate for Action Values: if we do not have a model of the environment, it is more useful to estimate action values rather than state values. One of the main objectives of MC methods is to calculate Q∗. In the case of action valuation, the considerations on the types of MC methods are equivalent. The only thing to add is that there are also actions to consider. The goal of learning action values is to make it easier to choose among the many actions accessible in each stage. To avoid problems and compare alternatives, we need to estimate the value of all actions in a state, not just the one we favour.

[60] For example, it is useful if I want to calculate the value only for a subset of states rather than for the whole set.

We need to keep exploring for policy evaluation to work. One method to achieve this is to define that each episode begins with a state-action pair, with each pair having a non-zero chance of being chosen as the beginning pair. Within the limit of an unlimited number of episodes, this assures that all state-action pairs will be visited an infinite number of times. To ensure that all state-action pairs are encountered, the most frequent alternative solution is to examine only stochastic policies with a non-zero chance of picking all actions in each state. Before proceeding and talking about an MC type control method, let us make a further distinction between the algorithms:

- **On-policy**: the agent learns by changing the policy it is following;
- **Off-policy**: the agent learns by changing another policy, different from the one he is following.

Now it is time to look at how Monte Carlo estimate may be used in control, or to approximate optimum policies.

Monte Carlo type control: also, for MC type methods we can use a GPI (Generalized Policy Iteration) type approach. For the policy evaluation phase, we can follow an evaluation method indicated in the previous paragraph.
Let us make two hypotheses for the moment:

- There is a non-zero probability of choosing each action in the initial state;
- Infinite number of iterations.

The policy improvement is done by making the current policy greedy with respect to the Q^{π}(s, a) calculated in the previous section:

$$\pi(s) = argmax_a Q^{\pi}(s, a)$$

This is done ∀s ∈ S. The policy π_{k+1}(s) calculated in step k is computed from $\pi_k(s)$ in this way. When the policy becomes stable in the sense that π_{k+1}(s) = $\pi_k(s)$ ∀s ∈ S, then it means that we found the optimal policy and the corresponding Q^*.

For each action, a ∈ A(s):

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s, a)) \\ &= \max_a q_{\pi_k}(s, a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &\geq v_{\pi_k}(s). \end{aligned}$$

To conclude, in our case we will not work with an infinite number of iterations, so we will remove the second hypothesis. To do this we can operate in two ways:

1. Place stopping conditions in the policy evaluation of $Q^{\pi k}$ (s, a), which usually requires many steps;
2. After each episode, because of earnings, are carried out a policy evaluation step and a policy improvement step.[61]

In the next section we will deal with **temporal difference** learning methods, which are the basis of the approaches that we will use in the experimental phase.

2.4.3 Temporal Difference methods

Everything we described so far was to introduce **Temporal Difference** methods (TD).
The TD methods:

- Inherit from MC methods the idea of learning directly from the experience accumulated by interacting with the environment, without knowing the dynamics of the system itself;
- Inherit from DP methods the idea of updating the estimation of functions in one state, starting from the estimates made in other states – i.e., the bootstrap.

Let us start by analysing how policies are evaluated, that is the *prediction* problem, the problem of estimating the value function v_π for a given policy π. For the *control problem* – which consists in finding the optimal policy – TD also uses variants of the more general GPI method. The main differences between the TD, DP and Monte Carlo methods lie in the logic of solving the prediction problem.

[61] In the sense that we do not wait for all the steps of the method to be done until convergence is achieved.

The TD prediction problem: we saw that Monte Carlo methods, despite having an update rule that allows learning through experience, must wait for the end of the episode to make the updates. However, in the case of TD methods the approach is completely incremental – i.e., learning is done at each time step. The simplest TD method is known as TD (0) where the update rule is:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

Unlike the MC methods that have R_t as their goal (paragraph 2.4.2), the TD method has r_{t+1} + + $\gamma V_{t+1}(s_{t+1})$ as their purpose. This TD method is called TD (0) or one-step TD, as it is a special case of n-step TD methods, which I will not illustrate in this thesis, and it combines the DP and Monte Carlo estimates:

1. Samples the expected value of Rt as in the Monte Carlo methods;
2. Uses an estimate of V calculated at instant t rather than V^{π} as in the DP methods.

At this point we can analyse the advantages that the TD methods have over the MC and DP methods. The most obvious advantage over DP methods is that they do not require an environment model. The advantage with respect to MC methods lies precisely in the fact that MC methods are episodic and TD methods are "step-by-step". As a matter of principle, this makes learning faster. Furthermore, it is possible to show that TD (0) converges towards the optimal solution.
Let us look at three methods that use TD prediction methods for the control problem: the first two are the ones I will use for analysis in my thesis, while the third is a method that could be applied to this market in the future. The three methods are:

- **SARSA;**
- **Q-Learning;**
- **Actor-Critic.**

In this type of controls, we will use the Generalized Policy Iteration pattern and, for the evaluation part, we will use the TD methods. Once again, we must find a compromise between exploration and exploitation and so the two approaches are divided into two classes: on-policy and off-policy.

SARSA: now let us look at an *on-policy* TD control method. Instead of learning a state-value function, the initial step is to learn an action-value function. In fact, in this case we must estimate Q^{π} (s, a) for

every state and every action that can be done starting from that state. We no longer consider state-state transitions but consider transitions between state-action pair and state-action pair. From the point of view of convergence, considering an S-S or S-A transition is the same thing: it does not alter the final result and ensures convergence. So, applying TD (0) the update rule becomes:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

This update is done after each transition to a non-terminal state s_t. If s_{t+1} is terminal, then Q (s_{t+1}, a_{t+1}) is set equal to zero. This update rule uses all five elements of the quintuple $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$. For this reason it is called SARSA (State-Action-Reward-State-Action). At this point we can make the policy π greedy with respect to Q^{π} and repeat the evaluation of the new policy, using the method we just discussed.

The convergence qualities of the SARSA algorithm are affected by the nature of the policy's reliance on Q. For example, “ε-greedy” or “ε-soft” policies could be used. Since all state-action pairs are examined an unlimited number of times and politics converges in the limit to greedy policies, SARSA converges to an optimal policy with 100% probability and, also, to an action-value function.

Q-learning: this algorithm has an *off-policy* control logic. In its simplest form the update rule is defined as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

In this case, the optimal value-function Q* is directly approximated by the action-value function Q and is independent of the policy followed during the learning process. The only impact that the policy has is to determine which status-action pairs will be visited and then updated. What is required to have a correct convergence is that all state-action pairs continue to be updated. However, demonstrations of Q-learning convergence were provided.

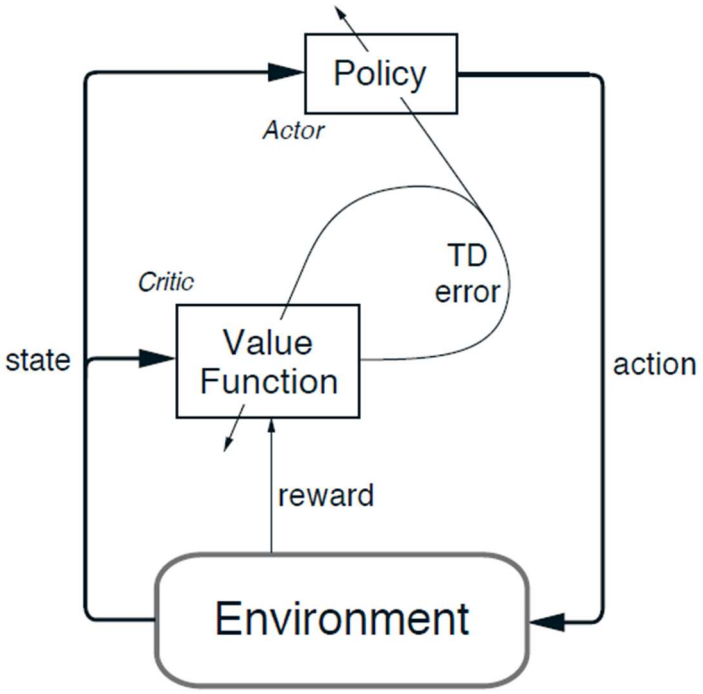


Figure 7: Block diagram for the Actor-Critic algorithm. Source: Sutton, Barto, Reinforcement Learning: an introduction.

Actor-Critic Methods: these methods have the characteristic of separating the memory structure to make the policy independent from the value function. The block of the policy is known as an **actor** because it chooses the actions; the block of the estimated value function is known as **critical**, in the sense that it makes a criticism on the actions performed by the policy that is being followed. From this it is understood that learning is of an *on-policy* type. In fact, the critic learns and criticizes the work of the policies, and in this way the critic takes the form of a *time difference error* (TD error); in Figure 7 is shown the block diagram describing the operation of the algorithm. The signals coming out to the critic are reported to the input both to the actor and to the critic itself. Typically, the critic is a state appraisal function. After each chosen action, the critic evaluates the new state to determine if the situation has improved or worsened than expected.

This evaluation is the TD error:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

where V is the critic's implementation of the current value function. This TD error can be used to evaluate the action just selected, the action a_t done in state s_t. If the TD error is positive, it makes

the critic think that the action a_t must be strengthened, while in the opposite case it must be weakened. Let us assume the Gibbs Softmax method is used to generate actions[62]:

$$\pi_t(s,a) = \frac{e^{p(s,a)}}{\sum_{b=1}^{n} e^{p(s,b)}}$$

The function p (s, a) expresses the actor's preference to select action a in state s. The strength or weakness in choosing an action is modulated using the following relationship:

$$p(s_t, a_t) \leftarrow p(s_t, a_t) + \beta\delta$$

where β is another step selection parameter and δ is the TD error.

This is only one application of the actor-critic approach. For instance, other variations choose the acts in other ways by modifying the amount of credit given to the action taken, a_t. For example, one of the most prevalent such factors is inversely connected to the likelihood of picking a_t, resulting in the following update rule:

$$p(s_t, a_t) \leftarrow p(s_t, a_t) + \beta\delta(1 - \pi_t(s_t, a_t))$$

Actor-critic approaches were used in a lot of the first RL systems that incorporated TD methodologies. Since then, increasing focus has shifted to approaches for learning action-value functions and determining a policy solely based on the estimated values (such as SARSA and Q-learning). This discrepancy might simply be a result of historical circumstance. For instance, intermediary architectures might learn both an action-value function and a self-contained policy.

In the next chapter we will go into detail regarding the SARSA and Q-learning methods by describing the structure of the algorithm applied to cryptocurrencies.

[62] See paragraph 2.3.4.

3. PRACTICAL APPLICATION OF REINFORCEMENT LEARNING TO ONLINE CRYPTOCURRENCY TRADING

In this chapter I will present a practical application of what I explained in chapters 1 and 2. I partially developed the SARSA and Q-Learning algorithms in the MATLAB® environment and I applied these two algorithms to seven cryptocurrencies – presented in the previous chapters – comparing the results coming from different parameter settings. In particular, were used two different reward functions: the Sharpe ratio and the Calmar ratio. The Sharpe ratio is the most widely used and widespread indicator in application to finance of Reinforcement learning; for this reason I will use it as a sort of benchmark to evaluate the performance of the Calmar ratio.

I will present and describe all the elements of the algorithm, starting from how it is developed, up to the use of the parameters. In short, I will present the chosen assets, I will illustrate the structure of the states and actions, the reward function and the squashing function, defining the various phases of the algorithm and starting to show the results, which will be proposed in the fourth chapter.

3.1 Explanation of the chosen parameters

To obtain the best results from the model, I checked various parameter settings and in conclusion I chose to use the following.

Among the parameters set for the application of the model there is k = 500, which indicates the number of iterations that are performed by the algorithm. It is believed that such value is large enough to attribute statistical significance to the study and is a value that allows algorithms to process data without requiring excessive computation time.

As for the parameters α and γ, I decided to use "standard" values, that is values often used in the various trading applications of this algorithm; α = 0.075 is chosen so small that the agent learns nothing from the newly acquired information, exploiting only the previous knowledge, and optimally scanning the solution space. The discount factor γ = 0.9 is chosen in such a way as to make the algorithm foresight, allowing to give importance to future returns.

As for the value of ε, were considered three different settings (10%, 15%, 25%) to test the possible performance improvements and to randomly explore the surrounding environment, following the multiple trade-offs between exploration and exploitation. In this way, the incidence of exploratory action is increased to verify if better performance can be achieved.

N is the number of past returns used to represent the states. In this study it was decided to use two settings, N = 1 and N = 7, i.e., the last N returns and the last selected action. I chose N = 7 because 7 corresponds to the days of the week to operate in the crypto market, or the last week of daily logarithmic returns; N = 1 is considered only the last trading day. In this way we try to understand if the algorithms are able to react more or less quickly to the information they get from the states.
Another parameter of which two settings are tested is L. I chose to consider L = 5 and L = 20 of market returns. The choice of these two parameters lies in the fact that using the very last days of the market would be too few to give significance for the calculation of the indicators, while choosing a too long period would lead to making today's choices on returns made more than a month ago. Since the selected period with L = 20 is much longer than the one with L = 5, I expect divergences between the reward functions. It is possible that with L = 5 the sample is not so large as to capture fluctuations or persistence of returns causing differences between the two indices.

3.2 Assets chosen with time span, short mention

As already mentioned in the previous chapters, the two models use the prices of seven financial assets belonging to the cryptocurrency market: *Bitcoin* (BTC), *Bitcoin-cash* (BCH), *Dash coin* (DASH), *Ethereum* (ETH), *Litecoin* (LTC), *Monero* (XMR) and *Ripple* (XRP). I downloaded the price series from Yahoo Finance considering the period from 17 Sep 2014 until 1 Jul 2021. Most of the assets have 2476 prices, except BCH – which, having been created from a fork in 2017, has a series of 1436 prices – and ETH – which has a series of 2152 prices starting from 7 August 2015. Among all the cryptos I chose the most famous and the most used in transactions.
The following tables show the prices and the returns of all the assets analysed.

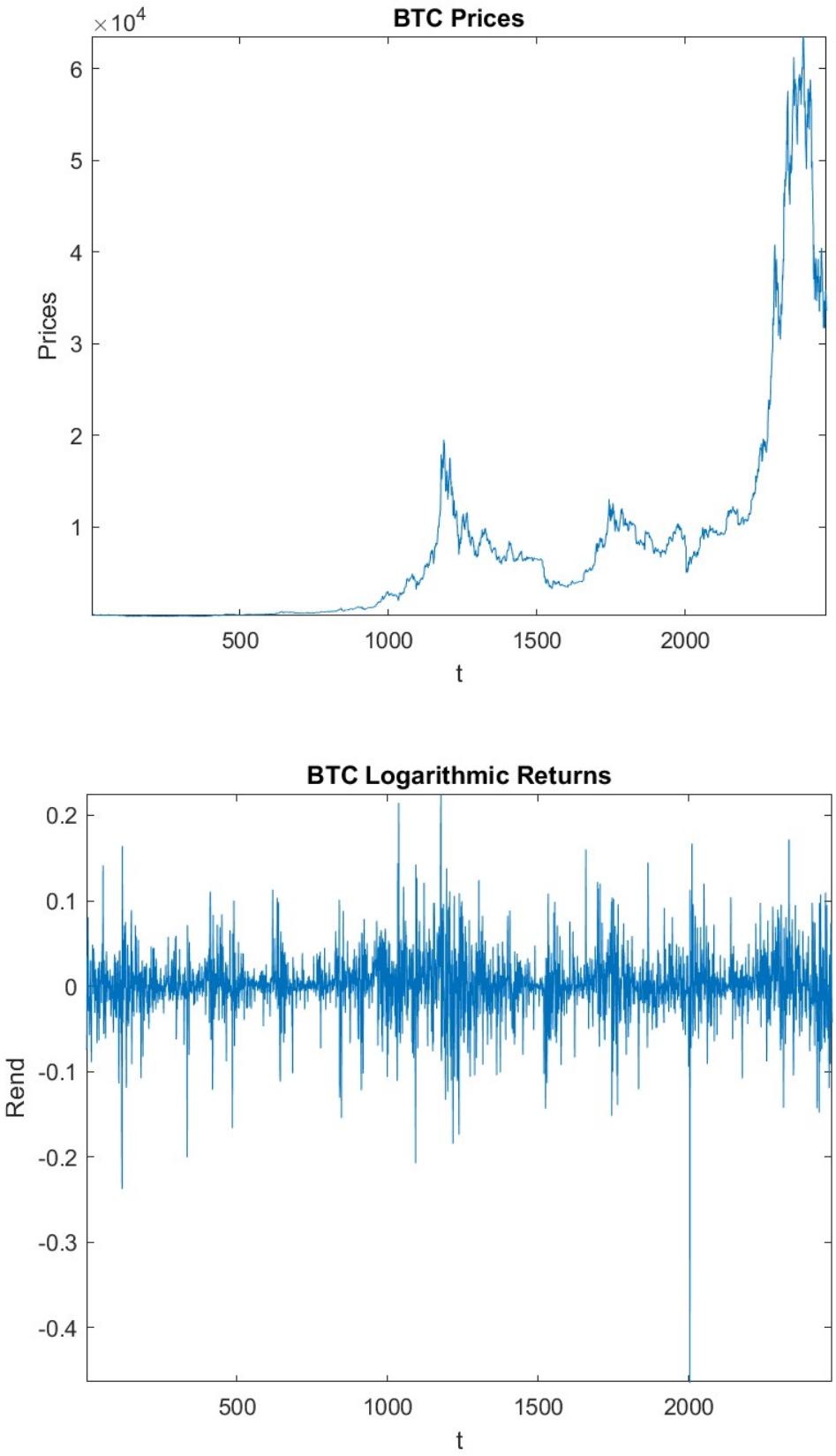


Figure 8: Bitcoin prices and daily returns charts. Source: Matlab.

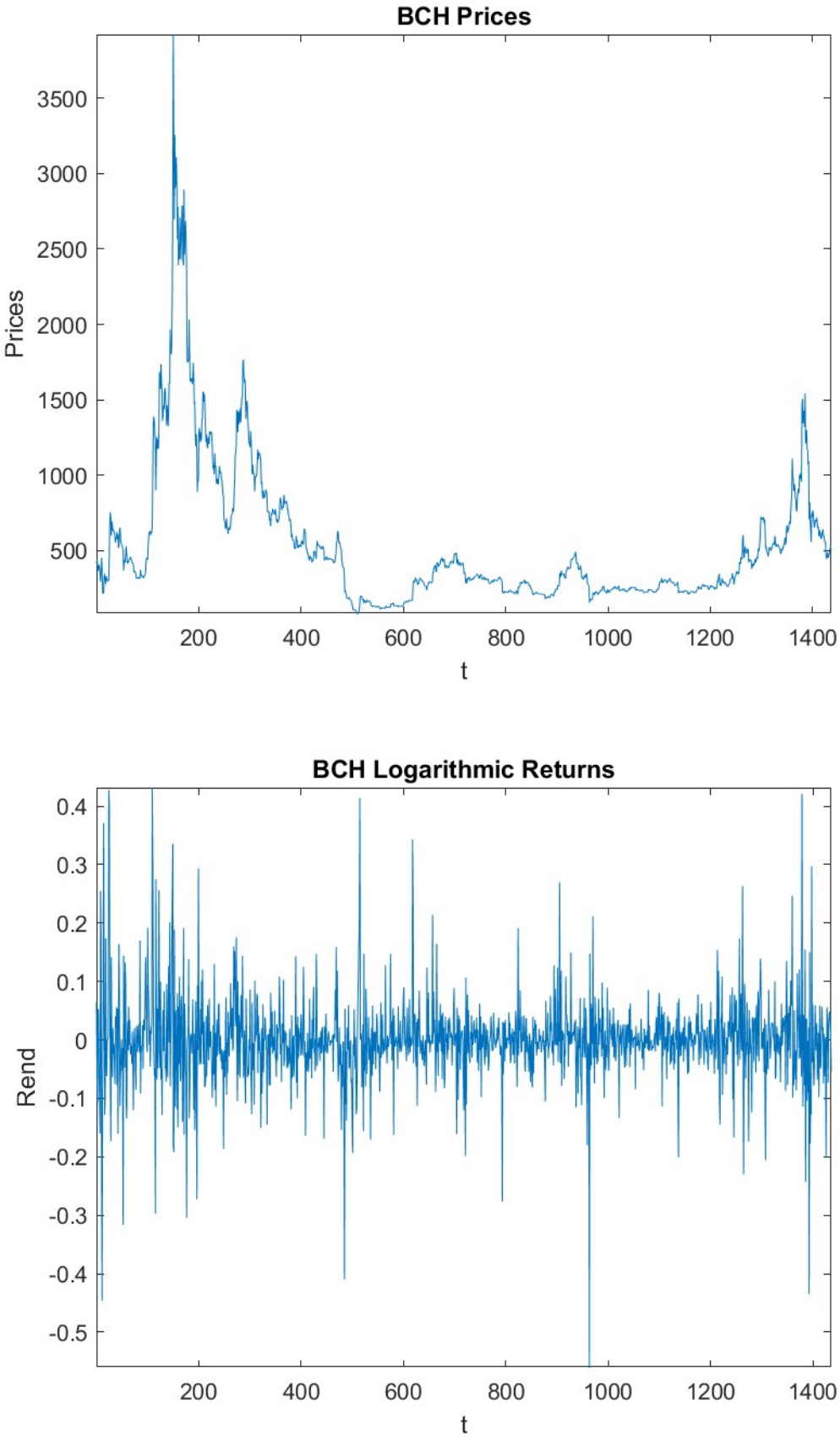


Figure 9: Bitcoin-Cash prices and daily returns charts. Source: Matlab.

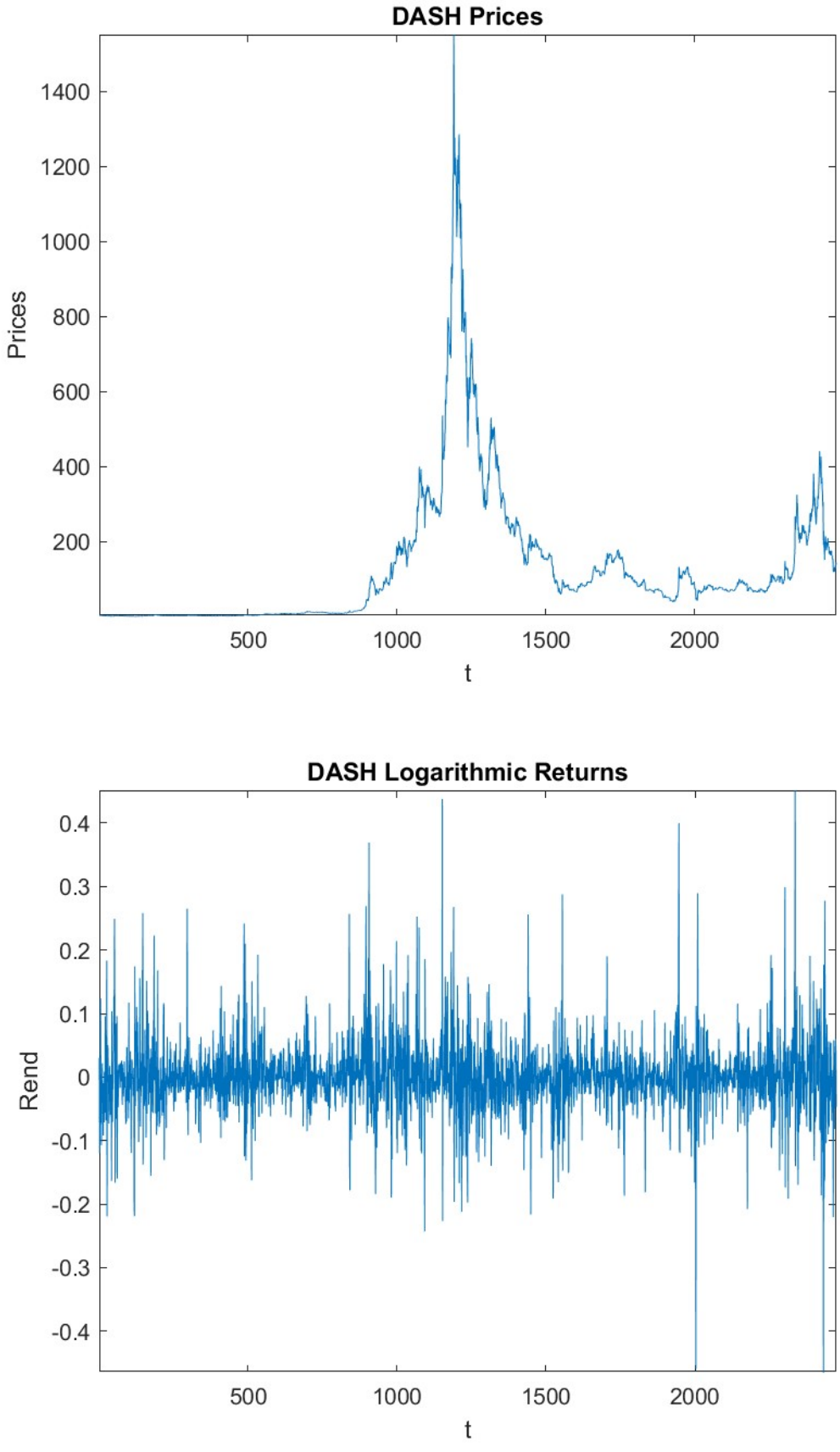


Figure 10: Dash Coin prices and daily returns charts. Source: Matlab.

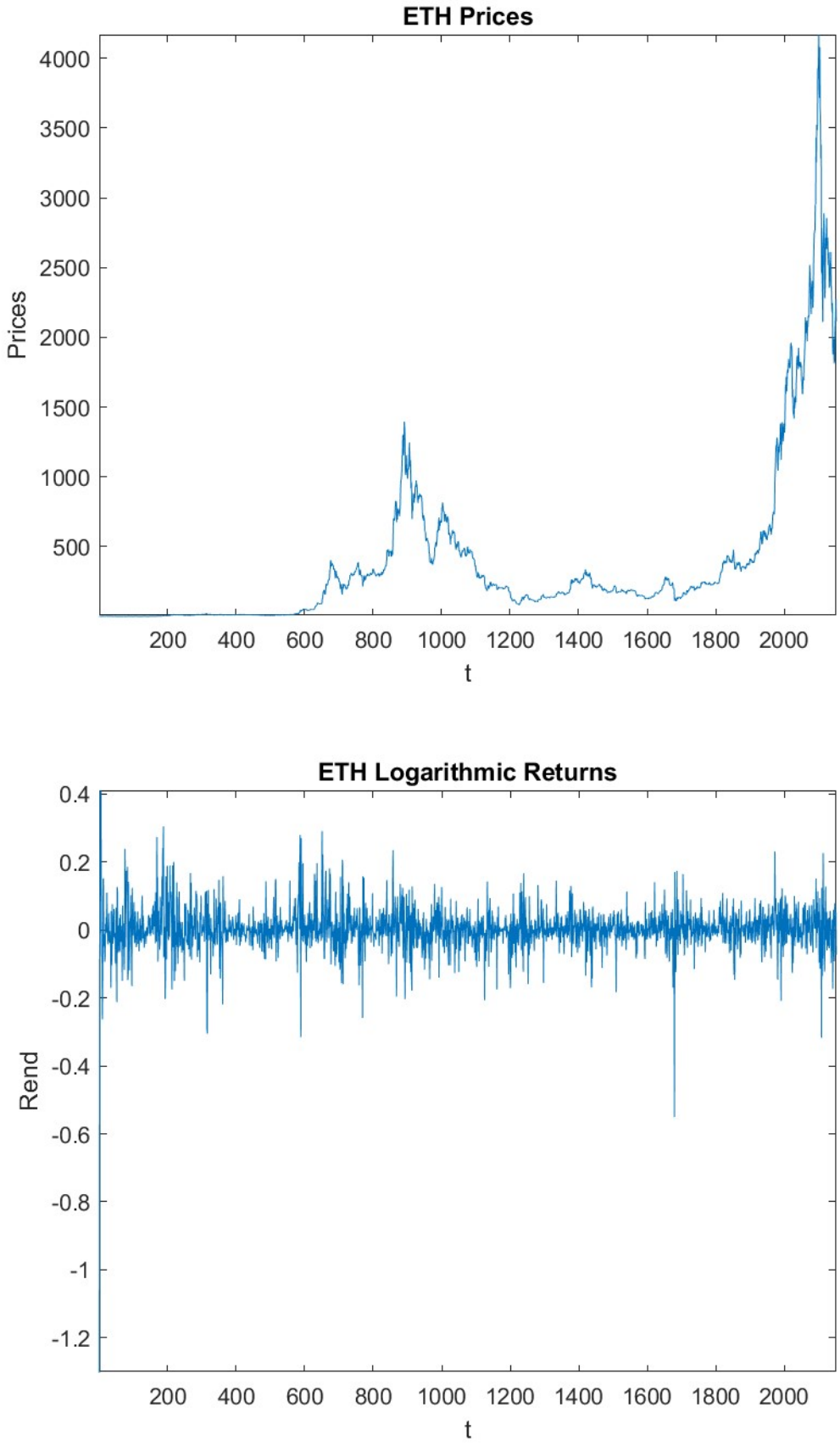


Figure 11: Ethereum prices and daily returns charts. Source: Matlab.

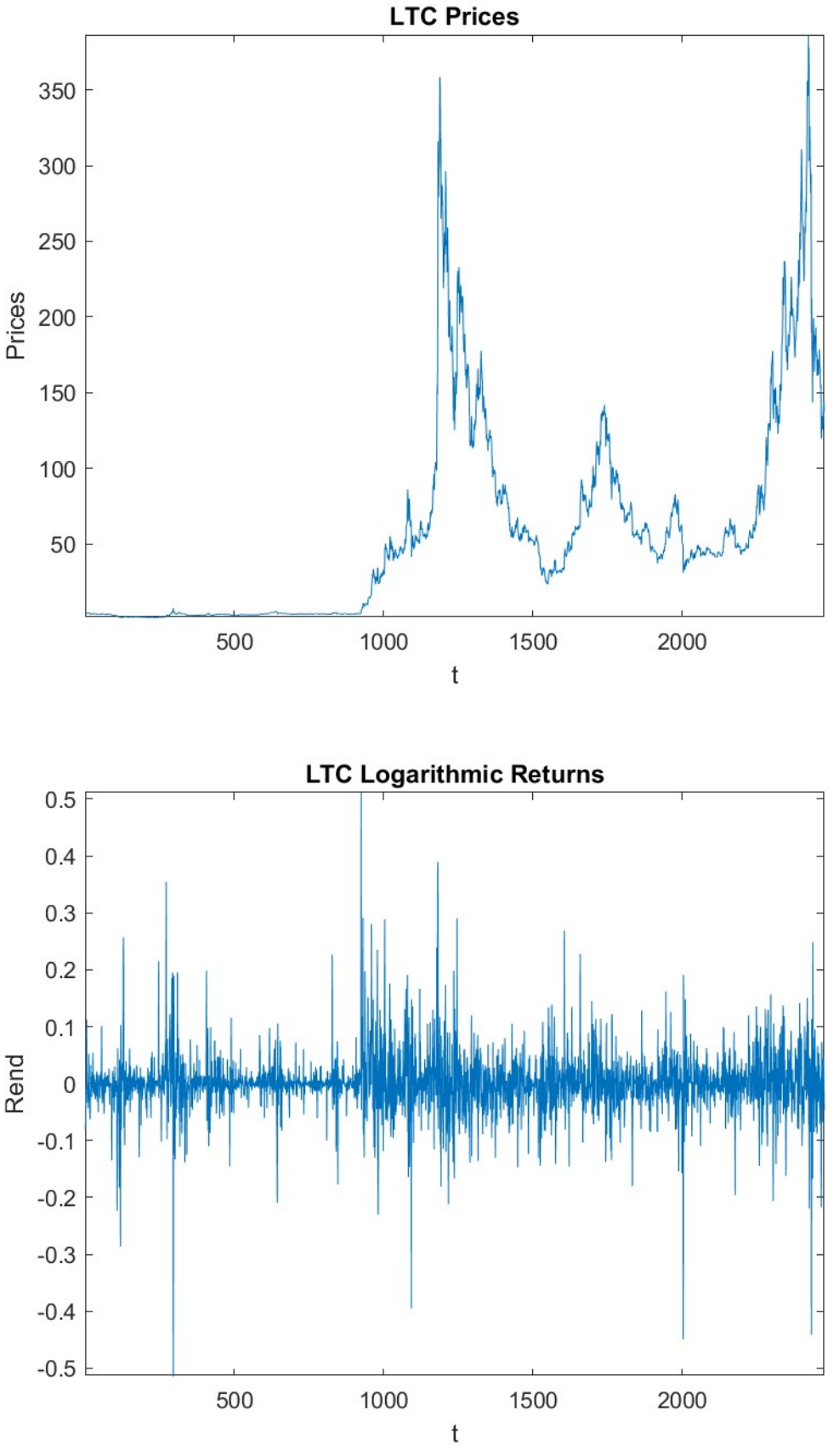


Figure 12: Litecoin prices and daily returns charts. Source: Matlab.

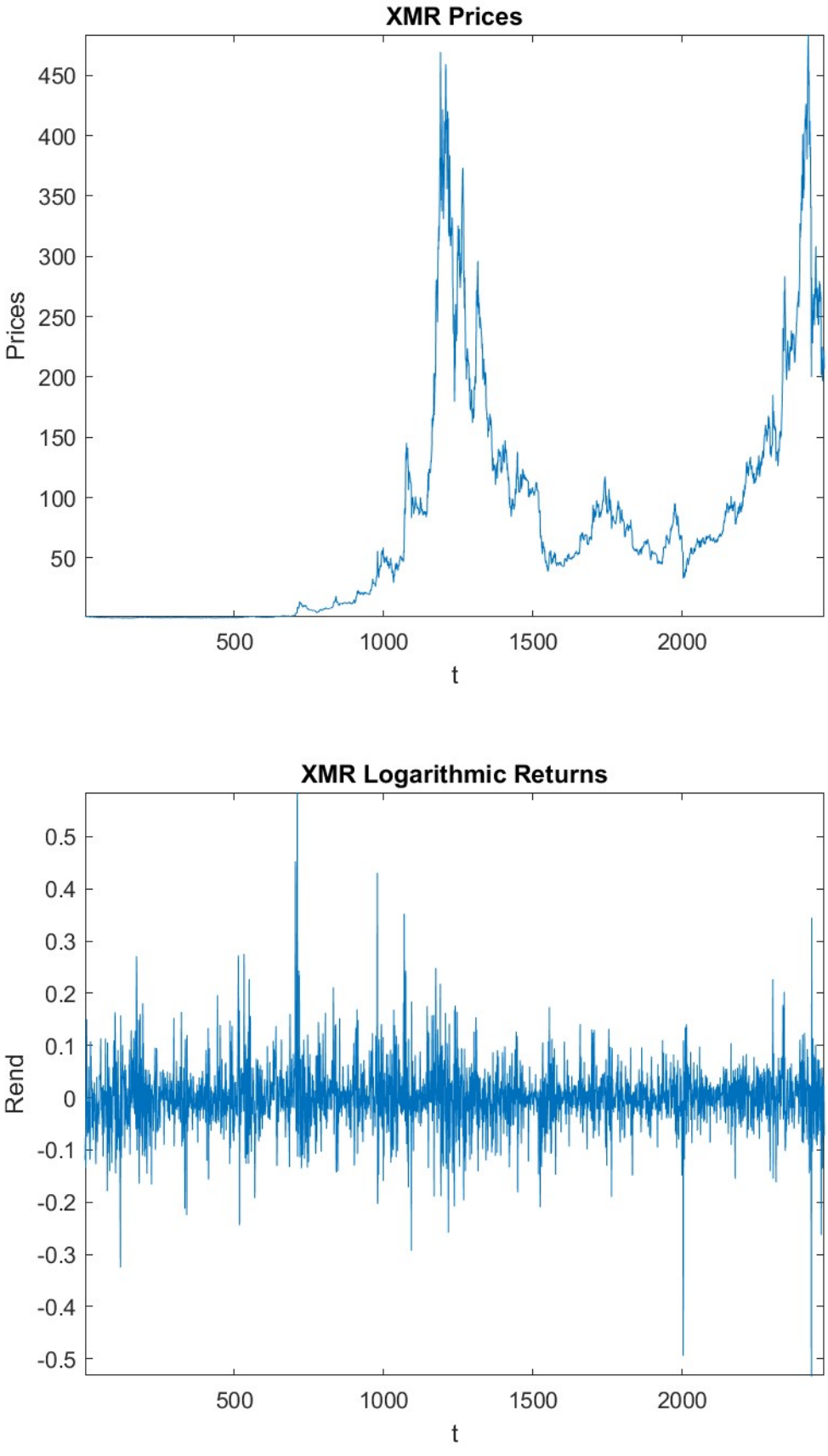


Figure 13: Monero prices and daily returns charts. Source: Matlab.

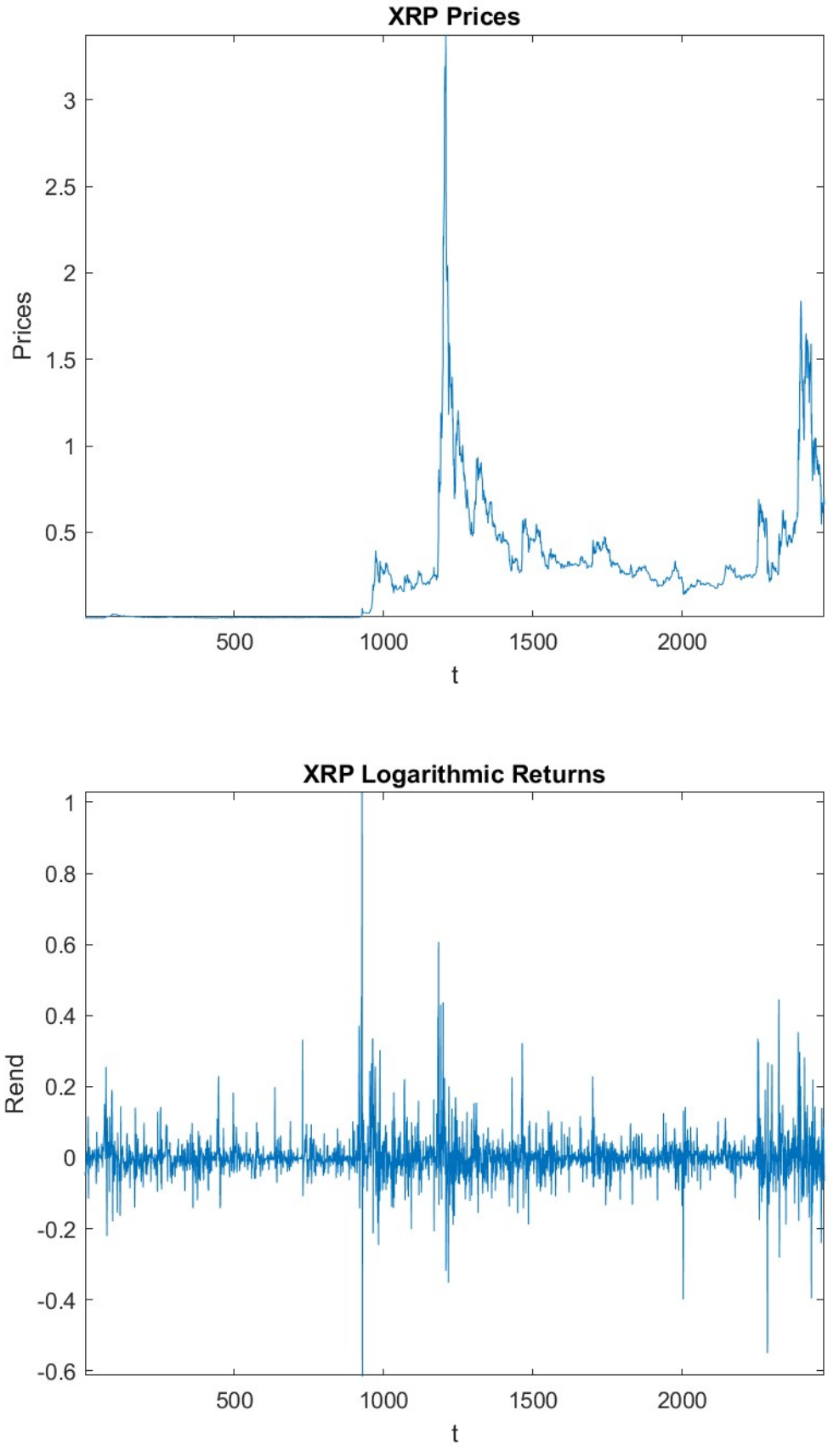


Figure 14: Ripple prices and daily returns charts. Source: Matlab.

	Bitcoin	Bitcoin Cash	Dash Coin	Ethereum	Litecoin	Monero	Ripple
Mean	7932.614772	537.6121868	129.765838	379.5029095	57.209119	77.180647	0.2707862
Variance	143026296.4	234167.3537	35552.0838	354580.9246	4415.6604	8439.0211	0.1240396
St. dev.	11959.3602	483.9084145	188.552602	595.4669803	66.4504357	91.864145	0.3521926
Max	63503.45703	3923.070068	1550.84998	4168.701172	386.450775	483.58362	3.37781
Min	178.102997	77.365776	1.08307	0.434829	1.15701	0.223522	0.00409
Skewness	2.769616342	2.573666811	3.09820418	2.992715993	1.73082243	1.7133822	3.0101406
Kurtosis	10.71642772	11.40119731	15.3388841	13.00918675	6.1509282	5.8963981	17.515136

Table 15: Descriptive statistics of the selected cryptocurrencies (excel processing).

Table 15 reports the main descriptive statistics of the selected cryptocurrencies. From the data it can be seen that the returns in all currencies have a leptokurtic[63] and asymmetric distribution, as is typical of financial historical series.

3.3 Structure of s-states and logarithmic returns

The interest in using the SARSA and Q-Learning structures, applied to a trading system, is to verify their performance capabilities using basic information to describe the environment. In the financial market, it is inconvenient to use a structure that is too complex because it would lengthen the computational times. For this reason, is represented the structure of the states: it consists of the last N-1 logarithmic returns per share, the current logarithmic return and the last action taken a_{t-1} (Corazza and Sangalli 2015). The s_t state is described by following vector:

$$s_t = (e_{t-N+1}, e_{t-N+2}, \dots, e_t, a_{t-1})$$

with the logarithmic return e_t represented by the following ratio:

$$e_t = \log \frac{p_t}{p_{t-1}}$$

[63] A leptokurtic distribution has a more pointed curve than the curve of a normal distribution, the central values are more frequent. In the present case, it means that returns with a value close to the average value occur more frequently (than in the case of a normal distribution).

3.4 Structure of the actions

In financial trading, actions are the possible moves that can be applied to different assets in the corresponding markets. Regarding the action system used in my study it is presented as follows:

$$a_t = \begin{cases} -1, & sell\ or\ short\ position\ in\ the\ market \\ 0, & stay\ out\ og\ the\ market \\ 1, & buy\ or\ long\ position\ in\ the\ market \end{cases}$$

where "to stay out of the market" means also to close any open position on the market (if any).
Once step t has been completed, it will be known which is the best action a_t to select; its evaluation will be available only at time t + 1 when I have obtained the yield of the selected security r_{t+1}, unless I decide to stay out of the market.
As described in the previous paragraph, it is extremely important to ensure a fair balance between exploration and exploitation. So, the action a_t will be chosen through the ε-greedy criterion, that is:

$$a_t = \begin{cases} argmaxQ(s_t, a_t, \theta_t) & with probability 1 - \varepsilon \\ a_t, & with\ probability\ \varepsilon \end{cases}$$

For example, if is set to ε = 10%, it means that with a 10% probability the choice of the action will be chosen randomly.

3.5 Transaction costs

In the applications, transaction costs are applied as a percentage if two consecutive trading operations are different ($a_{t-1} \neq a_t$). Because it is used for both opening and closing applications, the cost is cut in half. Clearly no costs are applied when the algorithm does not change the position ($a_{t-1} = a_t$). For both algorithms the costs are indicated with ($\delta = \frac{tc}{2}$) with $\mathrm{tc} = 0.15\%$. These expenses are imposed not just during the computation phase, but also throughout the operational phase, to make the algorithm responsive to potential earnings costs.

In other terms, during the calculation phase, total expenses are taken into account while calculating the equity line[64], with the following formula used after each step:

$$equity\ line_{t+1} = equity\ line_t * (1 + e_{t+1}a_t - \delta|a_t - a_{t-1}|).$$

It represents the value of the initial invested capital at time t + 1. The absolute value $|a_t - a_{t-1}|$ is used to apply the δ percentage only in the case of two divergent actions, that is if two different actions have actually been carried out between the periods.

3.6 Reward functions

The reward function considered here is a risk-adjusted indicator since it considers both the return and the risk associated with it. It provides the degree of investor satisfaction and allows the financial environment to provide feedback on the degree of goodness of the actions taken. In the developed algorithms I used two different reward functions: the Sharpe ratio and the Calmar ratio.

The Sharpe ratio is a best-known risk-adjusted return measure used for the management of market portfolios. In the literature[65] of the RL it is the best known and used index as a reward function and it is obtained from the following ratio:

$$SR_t = \frac{E_L(g_t)}{\sqrt{Var_L(g_t)}} \in \mathbb{R},$$

where SR_t is the Sharpe ratio at time t, $E_L(g_t)$ is the sample mean and $Var_L(g_t)$ is the sample variance, both calculated on the returns of the last L trading days. The Sharpe Ratio is commonly used in the literature[2] since it is straightforward to calculate and reflects the average of the returns throughout the studied period in terms of its standard deviation. The greater the value of this index, the better the portfolio allocation, punishing periods of high return volatility: the higher it is, the lower the ratio's value will be, resulting in a lower reward. The Sharpe ratio's primary flaw is that it equally weights positive and negative deviations from the period's average. This is quite an important problem because cryptocurrencies are characterized by high volatility and – since SR expresses risk with standard deviation – there is a danger of evaluating these deviations in the same way, positive for

[64] The Equity Line expresses the amount of the initial capital invested at the time of valuation t. It is a graph with the curve of profits and losses realized over time. It shows the earnings trend and allows you to understand the consistency and reliability of the trading system in a simple and intuitive way

[65] Reinforcement Learning for Trading Systems and Portfolios, John Moody and Matthew Saffell, Oregon Graduate Institute.

negative and vice versa. However – in terms of returns in the financial world – a positive deviation does not penalize returns: instead, it is desired since it increases the performance of the trading system. For this reason, it should not be considered in the same way as a negative deviation that instead decreases performance. Therefore, another risk-adjusted indicator was chosen, and above all, performance was compared with the SR_t.

I decided to use the Calmar ratio, a risk adjusted measure already widely used. It is a financial index used to measure the total return in relation to the drawdown risk; therefore it allows to compare the potential gain with the possibility of loss of a given investment.

The formula that allows its calculation is the following:

$$CAR_t = \frac{E_L(g_t)}{DDmax_L},$$

where the numerator is the same as the Sharpe ratio while the denominator $DDmax_L$ is the maximum drawdown which represents a measure of the investor's risk. It can be seen as the maximum loss that an investor could incur by buying at the highest level or by selling at that lower. The difference with the Sharpe ratio, therefore, is that the maximum drawdown replaces the standard deviation of the SR_t as a measure of the risk of the considered asset. The aim is to compare the two risk measures and see which one is the most successful at capturing the volatility in a series of returns, using the Sharpe ratio as a benchmark, and comparing it to the Calmar ratio.

3.7 Squashing function

The squashing function plays the role of transformer of the inputs of the s_t states in order to obtain values that make the evaluation process of the value function Q (s_t, a_t) more sensitive. The procedure applied performs like a clustering process, that is the grouping into clusters of values that are very similar to each other (Corazza and Sangalli, 2015).

The choice was to use the logistics function:

$$\boldsymbol{\phi}(\mathrm{x}) = \frac{a}{1 + be^{-cx}} + d,$$

with the following values as parameters: a = 2, b = 1, c = 10^{15} and d = −1.

The choice of using a very large c value makes the squashing function like a hyperbolic tangent function, graphically comparable to a "step". In this way, even the smallest variations of the states make it possible to give a signal, which can be used by the agent to distinguish even small variations

in yield to weight better the investment choices. Basically, the function will allow to cluster values that are very close to each other and subsequently to distance them, following this transformation.

3.8 Phase definitions

In the first step of applying the SARSA and Q-Learning algorithms, are introduced the weight vectors θ on the basis of a uniform probability distribution within the interval [-1.1]. 500 iterations are performed for each different setting of the parameters. For each iteration K, random values are generated will allow you to select between the greedy or the exploratory action. If the generated values are less than ε, the value of Q(s, a, θ) is maximized by carrying out an exploratory action for which one of the three available actions (-1,0,1) is randomly selected and the relative state-action value function is computed through the logistics function. On the contrary, if the random value is greater than ε, the action a_t will be selected according to the greedy criterion, that means the value function will be calculated for each action and the results will be positioned in a vector, then selecting that value that maximizes the state-action function; this procedure is applied both for SARSA and for Q-Learning. The vector with the resulting value function is the following:

$$[maxQL, rand(3)]) = max\left[Q(s_t, a_{t,-1}, \theta_t) \quad Q(s_t, a_{t,0}, \theta_t) \quad Q(s_t, a_{t,+1}, \theta_t)\right]$$
$$[maxSARSA, rand(3)]) = max\left[Q(s_t, a_{t,-1}, \theta_t) \quad Q(s_t, a_{t,0}, \theta_t) \quad Q(s_t, a_{t,+1}, \theta_t)\right]$$

In the case of two actions with the same result the choice is indifferent, since the value function will always be maximized.

Once the action has been selected, the profit of the following period is obtained through the formula $g_{t+1} = e_{t+1}a_t$, that is the product between the logarithm of the prices net of the transaction cost and the action taken at the previous position – based on daily realized return. Now we can obtain the gross equity lines for each iteration k and period t + 1 expressed in a prefixed currency:

$$equity\ line_{t+1,k}^{QL} = equity\ line_{t,k}^{QL} \cdot (1 + g_{t+1}^{QL})$$
$$equity\ line_{t+1,k}^{SARSA} = equity\ line_{t,k}^{SARSA} \cdot (1 + g_{t+1}^{SARSA}).$$

The next step is to calculate the reward function on the last L returns.

Thus, the components necessary for updating the vector θ_t are obtained: the value of the subsequent state s_{t+1}, the TD error $r_{t+1} + \gamma Q_k(s_{t+1}, a_t) - Q_k(s_t, a_t)$ and the gradient $\nabla_{\theta_t} Q$. At this point it is possible to complete the update of the weight vector $\theta_{t+1} = \theta_t + \alpha\delta_t \nabla_{\theta_t} Q$.

The number of iterations, for the reasons already explained above, is high (k = 500); therefore, for each step are calculated a number equal to k actions. In order to make the algorithm operating, requires that it provides only one trading indication for every t-th period. So, I need a rule to aggregate the K actions. It was decided to follow the approach present in Corazza and Sangalli (2015) or by making the average of all the actions that the algorithms have selected:

$$\bar{a}_t = \frac{\sum_{k=1}^{K} a_{t,k}}{K},$$

where $\bar{a}_t$ is the average of the k actions of the t-th period, $a_{t,k}$ is the action at the k-th iteration of the t-th period and K is the complete number of iterations (in my case 500). Then, the result obtained is translated into an operational signal through the following logic function:

- If -1 < a < -1/3 then a_t = -1 i.e. “short position in the market (sell)”;
- If -1/3 < a < 1/3 then a_t = 0 i.e. “out of the market”;
- If 1/3 < a < 1 then a_t = 1 i.e. “long position in the market (buy)”.

Finally, there is the calculation of all the statistics and information for both methods. To evaluate the number of operations induced by the trading system implemented, are calculated by the algorithm the definitive equity lines, the average daily returns[66] (then used for the calculation of the average annual return[67]), the value of the final capital and the number of annual operations performed.

All this information is calculated at the gross and net of transaction costs.

Is also obtained the percentage of times that the equity line is higher than the level of the initial capital C – 100 in my case. This value is useful to evaluate the performance of the algorithm, not only based on the value of the final profit but also on the performance of the system throughout the analysis period. Moreover, to me this percentage is useful in the event of early termination of the investment: for example, if this percentage were equal to 75% it would mean that I have a 75% probability that my

[66] From each equity line, the final capital is extracted, both gross and net of transaction costs. On the basis of the extracted values, the average daily returns (gross and net) are obtained as follows: $r\_daily = (\frac{eq\_line_T}{C})^{\frac{1}{T}} - 1$, where C is the initial invested capital and T is the total amount of trading days considered in the system.

[67] The daily returns thus obtained are used to calculate the average annual return (gross and net): $r\_annual = (1 + r\_daily)^{365} - 1$.

investment – if it were to stop early – will have a final capital equal to – or greater – than the initial capital.

In the next chapter, I will present and discuss the results obtained with the various settings that I have illustrated in this chapter.

4. APPLICATION OF THE SARSA AND Q-LEARNING METHODS TO THE CRYPTO-CURRENCY MARKET: Observations and results.

In this chapter, I will present and discuss the results obtained from the application of Reinforcement Learning using the different configurations presented in the previous chapter. I will show below the results associated to each cryptocurrency so that to analyse the application of the algorithm in detail. I will present summary tables divided by algorithm, number of the vector of returns and by type of reward function applied. The aim is to highlight the differences between the performance of Q-Learning and SARSA, as well as to evaluate the considered alternative to the Sharpe Ratio as reward function, the Calmar Ratio. The final and primary target is to identify the best method and combination, by varying the setting of the parameters N, L, ε, in order to maximize the profit for a possible trading strategy.

In the next and final chapter, I will present my considerations or possible developments together with the conclusions.

4.1 BITCOIN

As for Bitcoin, Table 16 shows the results obtained with QL for N = 1. It can be noted that, using the Sharpe Ratio (from now SR) as a reward function, the final capital is always greater than 100 with an average annual net return of 58.22%, a very high value that would allow excellent gains in case of investment. Indeed, the maximum value of annual net gain is obtained with the configuration L = 5 and ε = 0.1 equal to 73.92% with a final net capital of $ 4263.43. As for the Calmar ratio (from now CAR), the average annual net gain of the configurations is higher, equal to 61.26%. This value is in line with the Sharpe ratio settings and, even, with the parameters L = 5 ε = 0.25 an annual net return of 78.86% was obtained with a net final capital of $ 5401.78, the relative equity line is 97.74% therefore the algorithm is almost always in profit. In general, with all configurations this value is high, considering that the lowest value is with the Sharpe Ratio L = 5 and ε = 0.15 equal to 80.44%, this means that in 19.56% of the investment time the equity line is at a loss. Although this configuration has the lowest equity line value of all, it still offers an annual net return of 53.48%, therefore very positive. Looking instead at the number of annual operations, it can be seen that with increasing ε the algorithm performs more operations, but also performs better. For example, with ε = 0.1 the operations on average were 11.13 while with ε = 0.25 they are 20.72, almost double.

N=1	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	4499.08	4263.43	75.30	73.92	83.15	83.07	10.62
		15	1975.22	1826.75	55.26	53.48	80.69	80.44	15.34
		25	2294.68	2071.76	58.73	56.36	83.84	83.84	20.06
	20	10	2462.79	2347.45	60.40	59.27	84.61	84.57	9.44
		15	2436.54	2274.26	60.15	58.53	83.68	83.68	13.57
		25	1590.46	1412.70	50.38	47.78	83.64	83.64	23.30
CALMAR	5	10	2104.46	2001.41	56.72	55.57	95.03	94.51	9.88
		15	5503.06	5156.06	80.59	78.86	97.37	97.25	12.83
		25	5922.60	5401.78	82.56	80.10	97.98	97.74	18.14
	20	10	1662.48	1543.65	51.37	49.72	91.56	90.99	14.60
		15	1521.85	1434.20	49.41	48.11	92.77	92.00	11.65
		25	2199.37	1973.04	57.74	55.24	89.94	88.85	21.38

Table 16: Bitcoin for QL with N=1. Excel elaboration.

Table 17 shows the results obtained with the QL algorithm with N = 7. As regards the Sharpe Ratio, the returns are as positive as the net final capital with a maximum of 84.44% for the return and $ 6286.63 for the capital with the configuration L = 5 and ε = 0.25. In line with the results obtained by N = 1, the average annual net return increased by 68.16% while annual transactions decreased (on average 12.81 compared to 15.39) and the percentages over 100 are also high and positive in this case. With the Calmar Ratio the situation worsens slightly compared to N = 1: the average annual net yield drops to 54.97, bringing only one parameter setting with an annual net yield of about 40% but with final net capital always above a thousand, L = 20 and ε = 0.1, equal to 40.92% and $ 1064.42 respectively. In this case the number of yearly operations with CAR N = 7 is similar in number to those with N = 1 (on average 13,93 compared to 14,75) while the percentages over 100 are all above 85%.

N=7	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	5134.23	5012.38	79.00	78.37	95.50	95.50	4.73
		15	2345.33	2168.71	59.43	57.59	75.86	75.82	15.37
		25	6725.08	6286.63	86.29	84.44	95.38	95.38	13.30
	20	10	4024.20	3813.87	72.67	71.31	95.46	95.46	10.64
		15	3078.32	2920.16	65.97	64.68	80.11	79.63	10.35
		25	1954.74	1744.33	55.19	52.60	82.26	81.00	22.47
CALMAR	5	10	1426.98	1350.89	48.09	46.90	90.25	90.21	10.78
		15	1986.43	1816.52	55.50	53.46	91.54	91.22	17.58
		25	4236.99	3801.44	73.92	71.15	87.37	86.00	21.42
	20	10	1064.42	1019.74	41.81	40.92	95.95	95.75	8.42
		15	2052.37	1934.04	56.26	54.89	92.88	92.88	11.67
		25	2872.12	2677.18	64.21	62.51	97.69	97.61	13.74

Table 17: Bitcoin for QL with N=7. Excel elaboration.

Therefore, comparing the results between N = 1 and N = 7, it can be seen that in both cases the results are consistent: both reward functions work adequately and allow the algorithm a constant and almost guaranteed gain. No negative returns were obtained, synonymous with a good and correct reading of the operating signals; the interpretation of these signals is of fundamental importance for obtaining positive returns. To support what has been said, see figure 18. It is made up of six panels, three for the Sharpe Ratio and three for the Calmar Ratio. The first panel shows the price of the considered asset in the period under analysis, the second panel summarizes the actions performed by the algorithm and the third panel shows the resulting equity line, the net one dashed and the gross one continuing. As an example, I have chosen the configuration N = 7, L = 5, ε = 0.1. It can be seen that already up to t = 500 there is a different interpretation of the market trend, with the CAR the action -1 or “short” has almost always been chosen, an action used only twice by the SR. This difference in shares did not affect much the algorithm's performance but, towards t = 1500, were saw the first differences. With the Sharpe the algorithm has decided to stay “out of the market” and to take some “long” action, while with the Calmar the algorithm chose to go often “short” by decreasing the value of the equity line. The fundamental difference was found towards the end of the analysed period: both algorithms went “long” to take advantage of the great growth in the market price. However, only with SR the algorithm managed to close the position in time to conclude with a final net capital of $ 5012.38, while Calmar held the “long” position until the end losing a large part of the capital and closing with a final net capital of $ 1350.89.

In conclusion, the QL performs well with both Sharpe and Calmar, the best results for Sharpe are obtained with N = 7 while with Calmar with N = 1. There is no clear preference for the choice of the greedy action or the exploratory choice, as the yield is higher in algorithms that use L = 5 rather than L = 20. This means that, considering the last 20 trading days and in the case of a trendless price situation, the signals lost their informative power; therefore, a few previous returns (5) are enough to make the algorithm react correctly to price fluctuations.

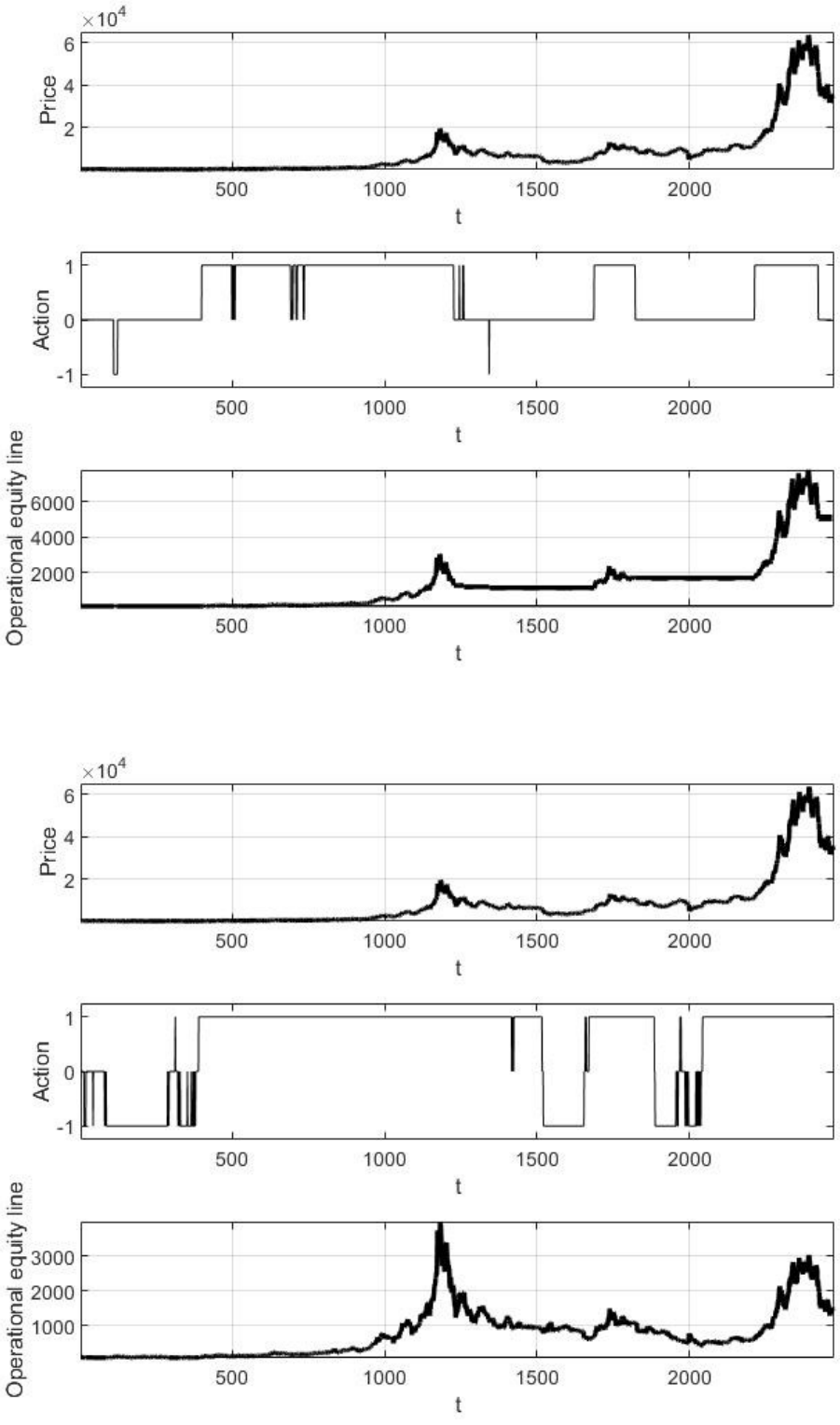


Table 18: QL with N=7, L=5, ε=0.1. Sharpe ratio above, Calmar ratio below. Source MATLAB.

The following tables show the results obtained with the SARSA algorithm. First of all, note that the performance obtained with SARSA worsens for all settings compared to the results obtained with QL.

It goes from an average annual net return of 60.66% with QL to 38.38% with SARSA. The table below shows the results obtained with N = 1. It can be immediately noticed how the performances obtained with respect to the QL case are decidedly of lower magnitude. All values are positive, but the annual yield is practically halved: it goes from 58.22% with QL to 29.53% with SARSA. Furthermore, with the Sharpe N = 1, L = 5 and ε = 0.15 we obtain the lowest return of all the parameter configurations applied to the Bitcoin series: the net annual return is 12.58% and the final net capital is $ 223.40, values well below those obtained with the QL algorithm.

N=1	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	450.66	409.27	24.86	23.10	75.03	74.83	18.88
		15	256.21	223.40	14.88	12.59	72.04	71.35	26.84
		25	678.32	614.41	32.62	30.70	75.03	75.03	19.47
	20	10	635.84	580.62	31.36	29.61	72.69	72.40	17.84
		15	1049.13	982.67	41.43	40.07	70.91	69.45	12.83
		25	1056.82	1034.11	41.58	41.13	75.27	75.11	4.28
CALMAR	5	10	2001.00	1946.40	55.56	54.93	92.65	91.84	5.46
		15	2135.63	2124.63	57.06	56.94	96.48	96.44	1.03
		25	1765.58	1707.02	52.72	51.96	86.55	85.45	6.64
	20	10	1580.02	1532.07	50.24	49.55	89.98	88.97	6.05
		15	1123.56	1107.51	42.87	42.57	82.02	80.77	2.80
		25	1275.42	1234.72	45.56	44.87	84.77	84.28	6.34

Table 19: Bitcoin for SARSA with N=1. Excel elaboration.

With the SARSA algorithm you can see a clearer distinction between the use of the Sharpe or the Calmar ratio as a reward function: the numbers speak for themselves, with N = 1 and SR we have on average a final net capital of $ 640.75, a net return annual average of 29.53% and the over 100 equity line on average equal to 73.03%; on the other hand, as regards CAR the corresponding values are $ 1608.72, 50.14% and 87.96%. Even in the case of N = 7 the Sharpe ratio performed worse than the Calmar ratio. Furthermore, with CAR were obtained better results than SR by carrying out fewer annual operations: on average with SR they are 16.69, while with CAR they are 4.72, almost four times less. In general, there are no differences between the various settings of L, while a trend identified is a slight improvement in performance as ε increases.

N=7	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	257.61	233.95	15.02	13.39	62.01	61.52	18.92
		15	309.49	270.73	18.18	15.86	73.71	72.34	26.31
		25	532.50	478.55	28.05	26.04	76.63	76.63	20.99
	20	10	371.85	342.71	21.43	19.97	65.21	63.99	16.11
		15	667.27	638.30	32.39	31.53	87.28	86.63	8.72
		25	566.95	524.02	29.24	27.74	75.94	73.11	15.52
CALMAR	5	10	1675.73	1632.19	51.65	51.06	85.11	84.42	5.17
		15	2161.02	2085.99	57.45	56.63	97.09	96.11	6.94
		25	1693.44	1627.72	51.88	51.00	92.15	90.37	7.83
	20	10	1524.66	1510.08	49.54	49.33	93.08	92.31	1.92
		15	1469.58	1442.38	48.73	48.32	88.83	87.82	3.69
		25	1808.21	1724.84	53.36	52.29	96.52	94.50	9.31

Table 20: Bitcoin for SARSA with N=7. Excel elaboration.

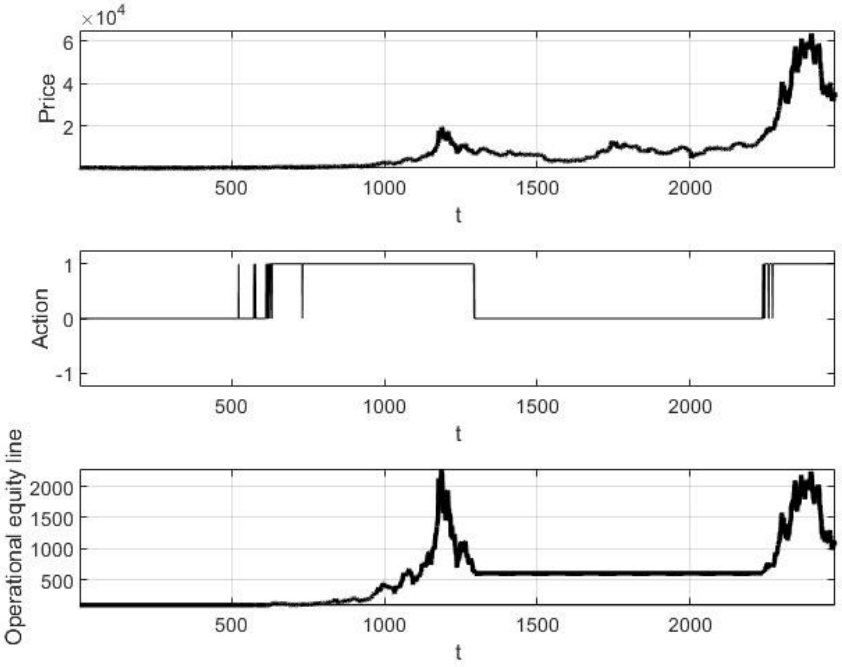


Table 21: SARSA with N=1, L=20, ε=0.25, Sharpe ratio. Source MATLAB.

An interesting comparison to make is that between N = 1 and N = 7 of the SARSA algorithm, reward Sharpe, L = 20 and ε = 0.25. The two algorithms perform practically identical for almost all the time, except for the fact that with N = 7 it performs many more actions in the period between t = 500 and t = 800, clearly losing money between net and gross capital. We can see the substantial difference in the last period: with N = 1 the algorithm understands the imminent fluctuation of the market and does not change its position, with N = 7 it processes too much the input values it obtains and executes those final actions that cause it to lose more than half of the capital. It would seem that the presence

of several elements describing the state vector cannot determine a return as positive as that of the case in which N = 1.

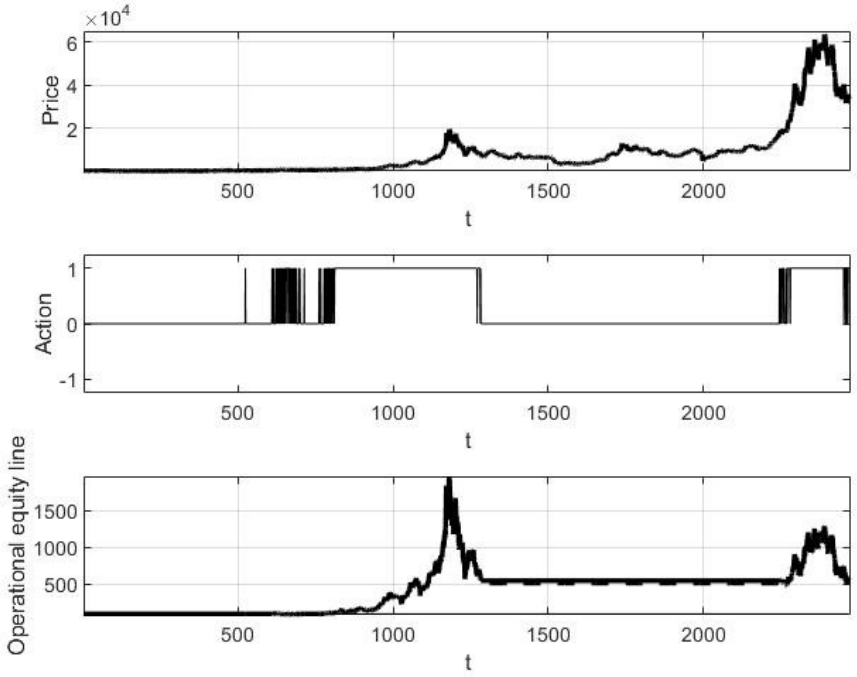


Table 22: SARSA with N=7, L=20, ε=0.25, Sharpe ratio. Source MATLAB.

In conclusion, I can reaffirm that the performances related to Bitcoin with QL are better than SARSA: using the Sharpe ratio as a reward function they are much better, while using the Calmar ratio in some cases they are equivalent. In general, considering all four output tables, the highest performance value is obtained with QL, Sharpe ratio, N = 7, L = 5 and ε = 0.25 equal to an annual net return of 84.44% and a net final capital of $ 6286.63 with an over 100 equity line percentage of 95.38%. on the other hand, the worst performance value is obtained with SARSA, Sharpe ratio, N = 1, L = 5 and ε = 0.15 equal to an annual net return of 12.58% and a final net capital of $ 223.40 with an over 100 percentage of the equity line equal to 71.35%. It should be emphasized that, even if significantly lower than the average of returns, the worst return is still positive and would allow anyone investing with these parameters to have only a 28.65% probability of being at a loss. As for the choice of the reward function, with QL there is no clear preference between SR or CAR, both perform very well and allow for large gains. On the other hand, with SARSA – as mentioned above – there is no such a comparison: with the CALMAR ratio the returns are more than twice as high.

Useful for the comparison between the algorithms are the next four figures: the first two contain all the k = 500 iterations that have been performed by the algorithms to obtain the results and the two

below the average line of these iterations – i.e., the average equity line. I chose to compare the two algorithms with QL and SARSA N = 1, L = 5, ε = 0.1 and Sharpe ratio as a reward function. It can be seen that up to t = 1200 the equity lines are more or less similar between the two algorithms: in fact, in both cases there is a peak in the average equity line. The difference therefore lies in the fact that from that peak the loss of capital was managed in a different way: with QL the loss was quite contained, going from 1600 to 800, while with SARSA the jump in percentage was more marked going from 1300 to 400. Finally, in the last investment period we can see that the equity lines are quite similar in terms of distribution between the two algorithms but with two different scales: QL on average managed to bring the capital from 800 to 2300, while the same "surge" with SARSA brought the capital from 400 to 700, much less than the tripling performed with the other algorithm.

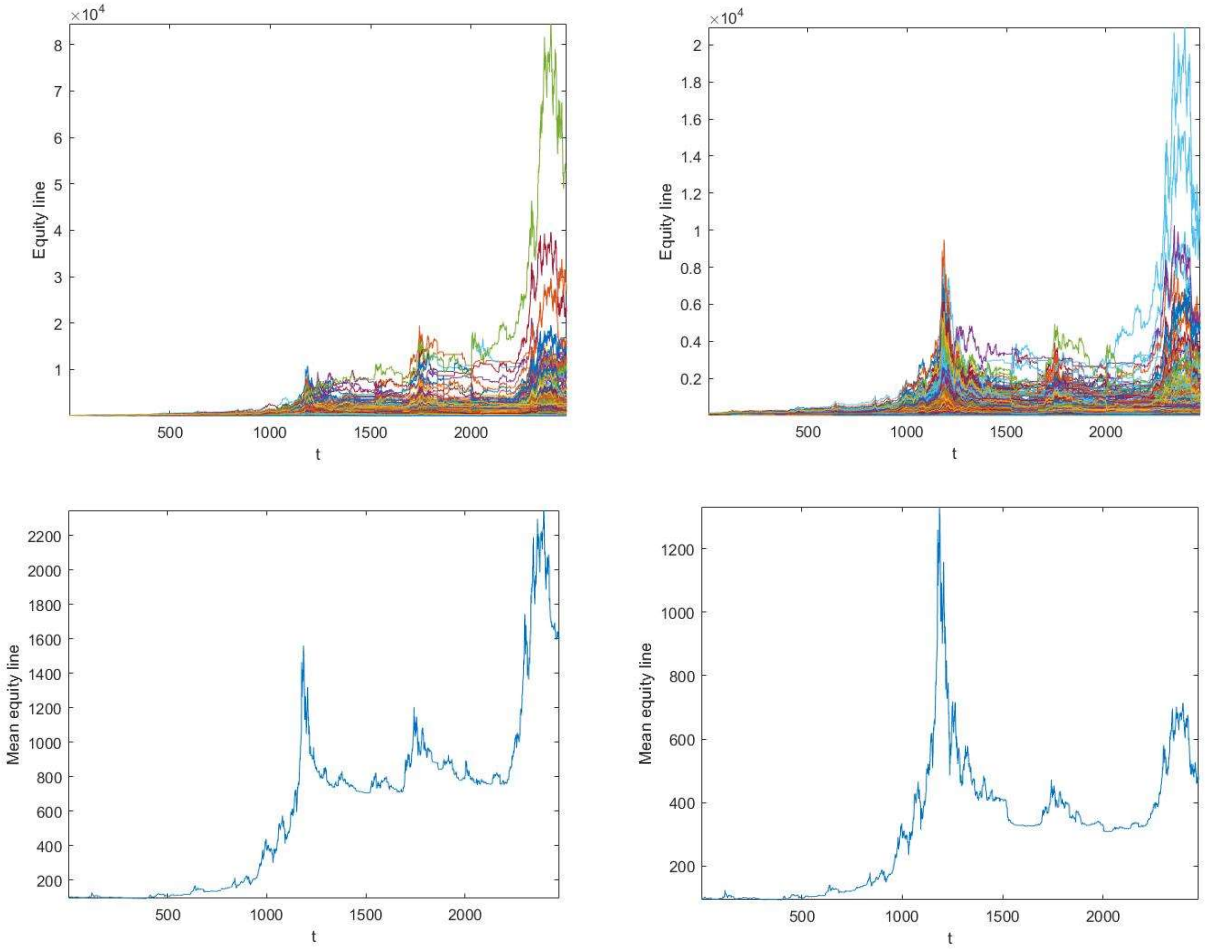


Table 23: Bitcoin, Sharpe ratio with N=1, L=5, ε=0.1. Above: resulting equity lines for QL (left) and SARSA (right). Bottom: average equity line for QL (left), SARSA (right). Source: MATLAB.

Summarizing and trying to draw conclusions, there are significant differences between the performance of QL and SARSA in the application to the BITCOIN. I have noticed that SARSA is less sensitive to large price changes, leading to lower yields than QL. As we saw, the QL algorithm manages

to limit the loss of the capital earned by closing and changing the position earlier than the SARSA algorithm: this is a very useful asset in the cryptocurrency market because the high volatility creates rapid changes in prices which – if incorrectly analysed – lead to serious losses. As a reward function, for QL both perform well and, on average, Sharpe offers higher final net capital and annual net returns, while Calmar offers better over 100 percentages than the other measure. If I had to advise a possible investor which strategy to follow, I would certainly recommend using the QL learning algorithm and, as a reward function, I would choose the Sharpe Ratio due to the slightly better average annual return than the CAR.

4.2 BITCOIN CASH

Continuing with the cryptocurrencies analysed, table 24 shows the results relating to Bitcoin-cash for QL with N = 1. It is immediately evident that there have been no positive returns and with the Sharpe ratio there is an average annual net loss of -22.33% with an average final net return of $ 44.86, or the initial capital has been more than halved. The configuration that tries to "defend" most the investment is with L = 20 and ε = 0.1, with an average annual net return of -10.06% and a final net capital of $ 64.92; the only fair value is the percentage of the equity line over 100 equal to 68.08%. It should be noted that the number of average annual transactions with these parameters is 5.09 while it is equal to 32.56 with the worst configuration with the Sharpe ratio – i.e., L = 20 and ε = 0.25 – which offers an annual net return of -57.57% and the final net capital equal to $ 3.44. It should be emphasized that the investment is 99.72% of the time at a loss.

Looking at the Calmar ratio, the situation worsens drastically. With all configurations, the initial capital is almost completely lost: the average annual net return is -74.48% and the over 100 percentage of the equity line is 0.31% with even two settings where it is equal to 0.00 – i.e., the algorithm is always at a loss. The algorithm also performs many more annual operations than those performed with SR, on average 27.05 with a peak of 53.16 for the configuration L = 20 and ε = 0.25.

N=1	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	48.55	47.52	-16.79	-17.24	57.98	57.70	7.12
		15	59.12	57.09	-12.51	-13.29	57.77	57.35	11.70
		25	65.40	62.78	-10.24	-11.17	47.46	46.48	13.74
	20	10	65.90	64.92	-10.06	-10.41	68.43	68.08	5.09
		15	34.56	33.43	-23.68	-24.32	16.93	16.86	11.19
		25	3.78	3.44	-56.52	-57.57	0.28	0.28	32.56
CALMAR	5	10	1.10	1.05	-68.28	-68.61	0.84	0.84	13.99
		15	0.35	0.33	-76.31	-76.68	0.00	0.00	21.11
		25	0.22	0.20	-78.90	-79.43	0.35	0.35	33.83
	20	10	1.11	1.06	-68.21	-68.54	0.00	0.00	13.99
		15	0.29	0.27	-77.40	-77.84	0.14	0.14	26.20
		25	0.44	0.38	-74.79	-75.78	0.56	0.56	53.16

Table 24: Bitcoin Cash for QL with N=1. Excel elaboration.

Moving to N = 7 the situation does not improve much: all the configurations provide negative returns remaining in line with the results obtained with N = 1. As regards the Sharpe ratio, the average annual net loss is -18.81% with an average final net capital of $ 47.14, slightly higher values than those obtained with N = 1. As for the equity line, only with L = 5 and ε = 0.25 there is a decent value of 79,29% while the remaining values are all below about 30%. The average number of transactions is similar to that of SR – 14,13 compared to 13,56.

N=7	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	52.63	50.53	-15.12	-16.00	60.46	12.39	13.79
		15	33.96	32.80	-24.11	-24.78	31.84	31.56	11.75
		25	93.23	89.70	-1.78	-2.74	79.29	79.29	13.28
	20	10	37.33	36.56	-22.25	-22.67	16.66	16.52	7.15
		15	26.87	25.61	-28.52	-29.39	1.40	1.33	16.35
		25	50.87	47.62	-15.86	-17.26	0.77	0.56	22.48
CALMAR	5	10	2.46	2.32	-61.14	-61.71	1.26	1.26	19.13
		15	1.02	0.97	-68.96	-69.32	0.77	0.49	15.56
		25	3.40	3.10	-57.79	-58.78	12.51	12.16	31.88
	20	10	0.26	0.24	-78.13	-78.51	0.42	0.42	23.21
		15	1.31	1.20	-66.89	-67.62	1.47	1.19	29.84
		25	0.56	0.49	-73.31	-74.20	0.07	0.07	45.66

Table 25: Bitcoin Cash for QL with N=7. Excel elaboration.

Even looking at the results with the Calmar ratio, the situation does not change with respect to N = 1. All the initial capital is practically exhausted having an average annual net loss of -68.36%, a net final capital of $ 1.39 and the over 100 percentage of the equity line on average equal to 2.60%. The average number of transactions is also high and similar to N = 1 (27.55). Whether with a single return in the

state vector, or with a week of returns, the Calmar Ratio was unable to correctly interpret the signals that the algorithm has processed. Even the Sharpe ratio did not achieve positive results but did not carry out the actions with which CAR squandered the entire capital.

To show this, some three-panel graphs that I have already proposed in the previous paragraph are interesting to analyse.

First, I wanted to compare best and worst performance with QL, N = 1, L = 5 and ε = 0.25, on the left Sharpe ratio and on the right Calmar ratio. We can immediately see how in the first period the algorithm differs in the actions performed: SR understands that it cannot make a profit from the period, so it remains out of the market while CAR repeatedly tries to go short, already losing almost all the capital. Subsequently CAR will try to buy or sell in vain because by now the initial capital was already exhausted. However, interestingly as regards SR the algorithm at t = 500 and at t = 1400 approximately had managed to obtain a good gain but, in both cases, it was not able to correctly modify the action necessary to be able to preserve the gains obtained. In the case of t = 1400, the downward decline in the price of Bitcoin-cash was not read correctly by the algorithm, preventing you from going short and capitalizing the investment.

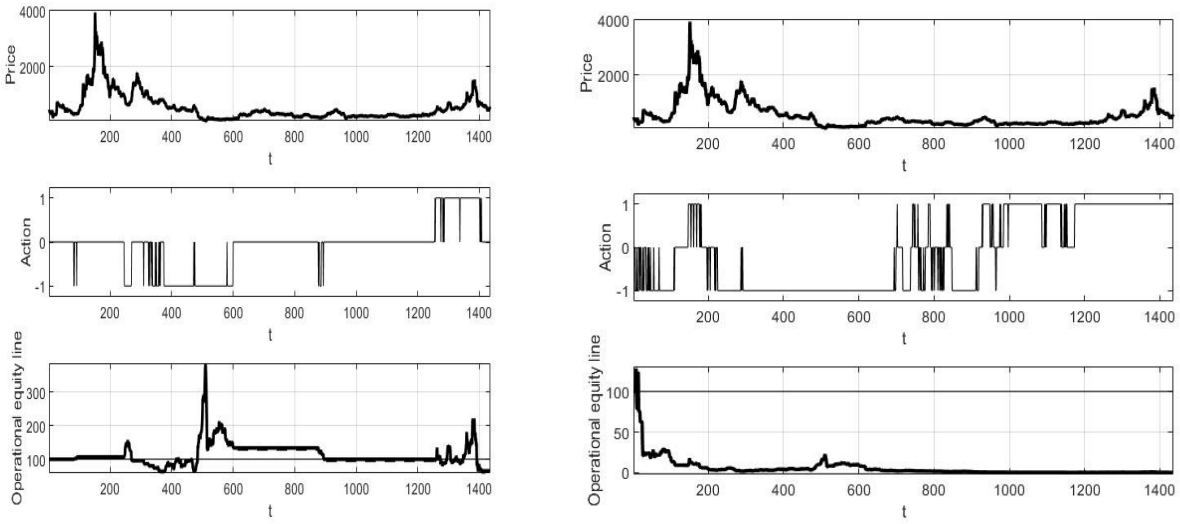


Table 26: Bitcoin Cash, QL with N=1, L=5, ε=0.25. Sharpe ratio on the left. Calmar ratio on the right. Source: MATLAB.

Another interesting comparison to make is that between the Sharpe ratio with L = 20, ε = 0.25 between N = 1 and N = 7. On the left N = 1 and on the right N = 7. The approach of both algorithms is very similar, except that the extra operations that were performed in the period between t = 400 and t = 800 have irretrievably almost zeroed the capital in the case of N = 1; probably having only one day of return in the state vector was not enough informative for the correct analysis of the price series. It

should also be noted that both have an almost zero over 100 percentage – 0.28% and 0.56% – but in the case of N = 7 the algorithm did everything possible to avoid losing all the initial capital, in fact it remained out of the market from T = 700 up to approximately T = 1400.

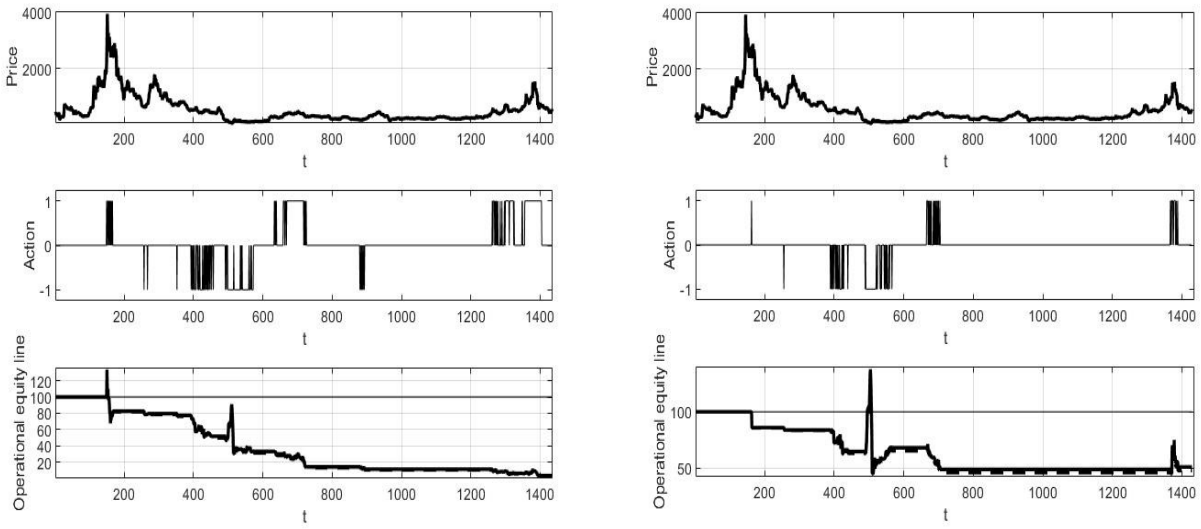


Table 27: Bitcoin Cash, Sharpe ratio, QL with L=20, ε=0.25. N=1 on the left. N=7 on the right. Source: MATLAB.

The results obtained with the SARSA algorithm will be analysed in the next tables. In the first table with N = 1 we can confirm the negative trend obtained with the QL algorithm. In this case, in addition to negative returns – on average -27.75% with Sharpe and -73.67% with Calmar – and in addition to the negative net final capital – on average $ 32.99 with Sharpe and $ 0.62 with Calmar – the over 100 percentage of the equity line on average for the two reward functions is equal to 1.85%, this means that for 98.15% of the time the algorithm is at a loss; it is a very low value which does not allow any profit margin or investment possibility, even in the short term. The value that is much lower than the 20.72% in the case of QL. The highest value obtained with the configurations is offered by the Sharpe ratio N = 1, L = 20 and ε = 0.1 with a net return of -1.93% and a final net capital of $ 92.61. However, the over 100 percentage of the equity line is very low equal to 0.07% and the average annual transactions are 6.11. This means that the algorithm has practically never gone into profit but has only limited the losses by trying to end with the same initial capital. As for the Calmar ratio, the results are quite incomparable, totally negative in line with QL. Even for three configurations, the over 100 percentage of the equity line is equal to 0.00% so the algorithm has always gone at a loss.

N=1	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	18.21	17.30	-35.16	-35.99	4.18	4.11	17.30
		15	20.40	18.45	-33.26	-34.95	9.27	9.27	34.08
		25	24.18	23.68	-30.31	-30.68	7.32	6.90	7.12
	20	10	94.29	92.61	-1.48	-1.93	0.07	0.07	6.10
		15	18.68	18.37	-34.74	-35.01	0.07	0.07	5.60
		25	28.67	27.56	-27.22	-27.95	0.00	0.00	13.23
CALMAR	5	10	0.42	0.41	-75.13	-75.26	0.84	0.84	7.38
		15	0.49	0.47	-74.10	-74.48	0.00	0.00	19.33
		25	0.20	0.19	-79.33	-79.64	0.35	0.35	19.59
	20	10	1.28	1.25	-66.99	-67.18	0.00	0.00	7.63
		15	0.33	0.30	-76.60	-77.10	0.00	0.00	29.00
		25	1.19	1.08	-67.62	-68.38	0.63	0.63	31.54

Table 28: Bitcoin Cash for SARSA with N=1. Excel elaboration.

Turning to the case N = 7, the situation does not change much with respect to N = 1. In one case, the algorithm even decided to stay out of the market rather than make a loss. This occurs with the configuration of parameters L = 20 and ε = 0.1. Clearly the final net capital is equal to $ 100, equal to the initial one and all other values are equal to zero because the only action performed by the algorithm was that (0) or out of the market. However, on average for the Sharpe ratio we get a net annual return of -14.46%, a net final capital of $ 58.976 and an over 100 percentage equal to 9.96%, slightly better values than the case with N = 1. On the other hand, as for Calmar ratio the net annual return is equal to -74.07%, the final net capital is $ 0.76 and the percentage over 100 is equal to 2.89%, values that reflect the trend with N = 1. Also, in this case there are no investment margins because the equity line is always at a loss. Only a parameter setting shows a decent value regarding the equity line. This is the case of L = 20 and ε = 0.25 with the Sharpe ratio, in this case the percentage over 100 is equal to 55.21% which, being above 50%, is a good result. In the graphs below I will show the details of this setting.

N=7	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	77.60	75.63	-6.27	-6.88	0.21	0.21	8.68
		15	44.52	42.92	-18.67	-19.43	0.84	0.84	12.26
		25	21.54	20.40	-32.44	-33.37	0.07	0.07	18.39
	20	10	100.00	100.00	0.00	0.00	0.00	0.00	0.00
		15	71.46	69.24	-8.22	-8.96	3.57	3.43	10.73
		25	46.92	45.67	-17.58	-18.14	55.21	55.21	9.20
CALMAR	5	10	1.18	1.15	-67.80	-67.99	3.42	3.42	7.65
		15	0.42	0.41	-75.23	-75.45	0.35	0.35	11.73
		25	2.52	2.35	-60.91	-61.57	10.27	8.87	22.70
	20	10	0.08	0.07	-84.02	-84.29	0.42	0.42	22.45
		15	0.33	0.30	-76.77	-77.34	0.00	0.00	32.65
		25	0.30	0.28	-77.30	-77.76	0.07	0.07	27.29

Table 29: Bitcoin Cash for SARSA with N=7. Excel elaboration.

In the figures below I compare two settings of the SARSA algorithm with the Sharpe ratio as a reward function, L = 20, ε = 0.25. The one on the left has N = 7 while the one on the right has N = 1. We can see how both graphs poorly process the information obtained from the price trend, almost always remaining out of the market. We can see the first difference at around t = 570 when both algorithms try to enter the market with a short position. N = 7 realizes that there was no possibility of gain and after two attempts at shares it returns out of the market avoiding the loss; N = 1 instead tries repeatedly to sell by losing more than half of the initial capital. A reason for this difference may be in the use of multiple elements of the state vector: the use of a greater numbers of returns may have helped the algorithm. The problem is that towards t = 1370 the situation repeated itself but with reversed parts. N = 7 goes long several times and after a good initial gain it loses everything practically halving the initial capital; for this reason, the percentage over 100 is 55.21%. Instead, N = 1 tries to go long a few times but only loses capital again. Both algorithms picked up a chance of earning given the growth of the market, but both made wrong actions. Finally, as regards the transactions performed, the Calmar ratio on average performed many more transactions than the Sharpe ratio – 23.60 compared to 12.87 – transactions mainly due to errors in the correct interpretation of the status signals and attempts to raise capital from a loss situation now almost complete.

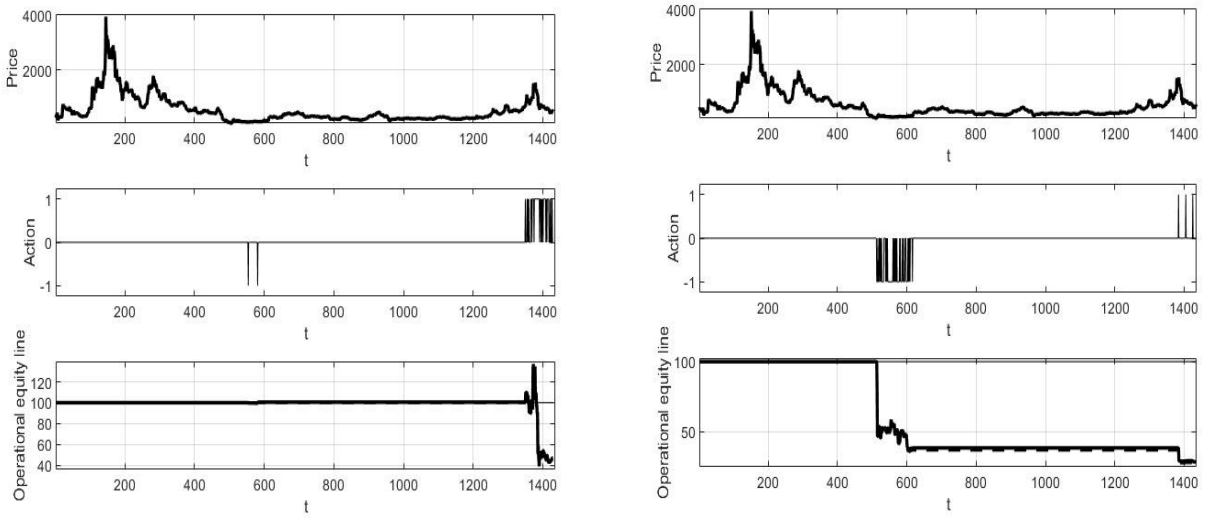


Table 30: Bitcoin Cash, Sharpe ratio, SARSA with L=20, ε=0.25. N=7 on the left. N=1 on the right. Source: MATLAB.

In conclusion, the possibility of investing in Bitcoin-cash both with the SARSA algorithm and with QL is almost nil. There are no substantial differences in performance between the two logarithms. Instead, between the two reward functions there is a big gap: although always at a loss, the Sharpe ratio managed – especially with QL – to obtain fair results as regards the over 100 percentage of the equity line. With N = 7, L = 5 and ε = 0.25, the net annual negative return was only -2.74%, but the equity line for 79.29% was in profit. On the other hand, Calmar ratio managed to lose more than 95% of the initial capital with all configurations, leading – on average among all configurations – to have an over 100 percentage equal to 1.35%. The investment with the Calmar ratio for 98.65% of the time was negative. Finally, referring to the transactions performed, the Calmar ratio on average performed many more transactions than the Sharpe ratio – respectively 23.60 per year compared to 12.87 per year – transactions mainly due to errors in the correct interpretation of the status signals and attempts to raise capital from a loss situation now almost complete. If I were to advise an investor which strategy to follow, I would definitely recommend that they try another method or with another cryptocurrency.

4.3 DASH-COIN

In tables 31 and 32 I will show the results for Dash-Coin obtained with the use of the QL algorithm, first with N = 1, then with N = 7. In general, the performance of QL is not very consistent bringing only a few configurations to have a positive return. As for N = 1, we find only a positive return for Sharpe with the configuration L = 20 and ε = 0.15, which allows a small annual net return of 3.17%

with a final net capital of $ 123.53. This positive outcome differs from the negative results obtained on average with the Sharpe ratio: on average we have a negative net return of -7.26% with a net final capital of $ 66.49. As for the equity line, the over 100 percentage is on average equal to 61.98%, a good value that allows investment margins. However, the average number of operations per year is quite high, on average equal to 22.57, a value which is approximately double compared to those carried out with the Calmar ratio (11.26). Even with the Calmar ratio there was only a configuration of parameters with a positive return: L = 20 and ε = 0.25 with an annual average net return of 2.38% and a final average net capital of $ 117.31. However, in the overall Calmar's performance compared to Sharpe was lower. Aside from the number of transactions, the average final net capital is almost half, equal to $ 36.37, the average annual net return is equal to -20.28% and for 50.75% of the time the equity line was lower than the starting capital. From these first results we can see how the algorithm improves performance with increasing exploration and that with L = 5 it is possible that the sample is too small to capture the fluctuations of returns in the market.

N=1	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	25.77	23.12	-18.13	-19.43	47.25	46.12	21.24
		15	79.19	70.25	-3.38	-5.08	62.61	62.41	23.61
		25	75.37	65.17	-4.09	-6.12	61.64	61.16	28.62
	20	10	46.26	42.76	-10.75	-11.78	62.69	62.09	15.34
		15	134.81	123.53	4.51	3.17	63.82	63.78	17.11
		25	86.08	74.11	-2.19	-4.32	76.43	76.35	29.51
CALMAR	5	10	14.84	14.27	-24.53	-24.97	32.90	31.65	7.82
		15	5.01	4.71	-35.70	-36.30	25.22	24.45	12.25
		25	26.97	24.74	-17.58	-18.62	48.87	47.29	16.97
	20	10	8.79	8.56	-30.14	-30.41	59.78	59.46	5.16
		15	38.60	36.61	-13.10	-13.78	63.38	62.81	10.47
		25	126.52	117.31	3.53	2.38	69.85	69.81	14.90

Table 31: Dash-Coin for QL with N=1. Excel elaboration.

Turning now to the table with the results of N = 7, we find better results than N = 1 for the Sharpe ratio: for the configurations both the annual average net return and the final average net capital is positive: 0.37% and $ 124.74 with an over 100 percentage on average equal to 62.53%. However, the number of average transactions per year remains high but lower than N = 1 (17.50). The maximum return obtained with QL was obtained with the configuration N = 7, L = 5, ε = 0.25 and the Sharpe ratio as a reward function: the annual net return is equal to 14.66%, the final net capital equal to $ 252.23 and the percentage over 100 equal to 69.77%. They are discrete values always obtained with the highest level of exploration and, if with SR the performance has improved, with CAR it has definitely

worsened: no parameter configuration provokes positive returns, the average net return is equal to -24.97%, the final average net capital is $ 22.48 and the average over 100 percentage is equal to 41.68%. The number of average transactions per year also grew, amounting to 13.99. A possible cause of this performance lies in the fact that, given the conservative nature of the index that uses the drawdown, the reduction in operations can have a negative impact on the final profit.

N=7	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	208.00	190.13	11.44	9.97	70.02	69.69	17.75
		15	34.16	31.06	-14.69	-15.88	44.73	44.61	18.63
		25	273.46	252.23	16.04	14.66	69.89	69.77	15.97
	20	10	151.06	142.07	6.29	5.33	73.06	73.01	12.13
		15	64.87	58.81	-6.20	-7.55	73.99	73.50	19.23
		25	82.62	74.13	-2.78	-4.33	45.18	44.61	21.30
CALMAR	5	10	6.20	6.01	-33.69	-34.00	23.89	23.52	6.35
		15	9.03	8.56	-29.91	-30.46	26.03	25.30	10.49
		25	12.39	11.21	-26.55	-27.63	47.65	46.11	19.65
	20	10	12.07	11.72	-26.84	-27.15	57.45	57.17	5.76
		15	11.41	10.26	-27.44	-28.57	33.52	30.77	20.84
		25	96.82	87.13	-0.48	-2.02	67.49	67.21	20.84

Table 32: Dash-Coin for QL with N=7. Excel elaboration.

An interesting comparison that we can make is the one between the two positive returns obtained with N = 1: SR, L = 20 and ε = 0.15 on the left; CAR, L = 20, and ε = 0.25 on the right. We can see how both algorithms tend to remain out of the market, indeed with the Calmar ratio it has almost always gone long or short, while Sharpe has remained out of the market for longer periods. The performance of both is very similar: the actions performed are almost always the same which, as can be seen from the period t = 1000, begun to bring large gains: with CAR they even went over $ 3000 in earnings. However, subsequently an incorrect interpretation of the market led to the loss of almost all of the capital. As seen in the previous paragraphs, Sharpe manages to stop a little earlier while Calmar continues to execute actions hoping for a growth and, therefore, a gain as happened at t = 1250.

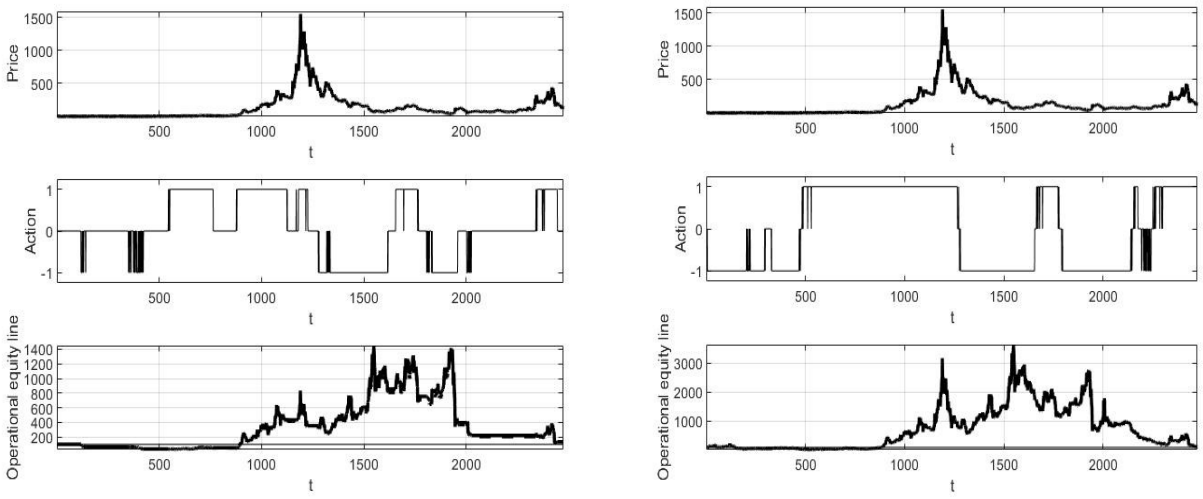


Table 33: Dash-Coin, QL with N=1. Sharpe ratio with L=20 and ε=0.15 on the left. Calmar ratio with L=20 and ε=0.25 on the right. Source: MATLAB.

Now, let us pass to the analysis of the SARSA algorithm. The two tables below show the results obtained with N = 1 and N = 7. The first thing we can notice is the yield gap between the Sharpe ratio and the Calmar ratio: the average annual net return in the case of the SR is -1.03% with a net final capital of $ 134.20, while for the CAR the average annual net return is equal to -42.19% with a net final capital of $ 4.03. Even with the QL algorithm the Sharpe ratio performed better than Calmar, but the difference was not so substantial. As for N = 1, there are some interesting details to point out. First of all, with SR there is another big difference in performance between L = 5 and L = 20: when with the first you get only negative returns with even an over 100 percentage equal to 0.23%, with the second we only have returns positive, an over 100 percentage equal to 65.40% and the configuration with the highest return as regards Dash-Coin – i.e., L = 20 and ε = 0.15 where the net annual return is equal to 20.24% and the final net capital is of $ 348.77. Second interesting thing to note is the average number of annual operations performed with L = 5 (10.23) compared to those with L = 20 (3.34). As already noted with the QL algorithm, it is possible that with L = 5 the sample used is too small to capture fluctuations or persistence of returns in the market, and for this reason it performs more actions than necessary. We can see the same similarity with the Calmar ratio: L = 20 performs better than L = 5 – average net return equal to -35.80% compared to -51.72% – the percentage over 100 is higher – 22.31% compared to 4.00% – and performs fewer transactions on average – 22.08 compared to 28.87.

N=1	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	23.13	22.22	-19.43	-19.90	0.00	0.00	7.67
		15	42.86	40.17	-11.75	-12.59	0.69	0.69	12.69
		25	56.48	53.58	-8.08	-8.80	0.00	0.00	10.33
	20	10	342.15	339.09	19.90	19.74	63.86	63.86	1.77
		15	352.95	348.77	20.45	20.24	63.95	63.95	2.36
		25	160.87	156.08	7.27	6.79	68.43	68.39	5.90
CALMAR	5	10	1.23	0.98	-47.73	-49.48	4.53	4.41	45.44
		15	0.51	0.44	-54.13	-55.07	4.08	4.04	27.59
		25	0.90	0.84	-50.10	-50.60	3.68	3.56	13.57
	20	10	8.85	7.58	-30.07	-31.66	21.95	21.75	30.39
		15	2.48	2.19	-42.05	-43.09	20.13	20.09	23.90
		25	6.83	6.43	-32.70	-33.30	25.55	25.10	11.95

Table 34: Dash-Coin for SARSA with N=1. Excel elaboration.

Turning to the table with N = 7, we notice many similarities with the results obtained from N = 1. Also, in this case L = 20 performs better than L = 5 and with the Sharpe ratio we obtain only the three positive returns as in the case of N = 1. The difference between SR and CAR is abysmal: when with the former I get an annual net return of -2.97% – negative due to the negative performance of L = 5 – and a net final capital of $ 108.42, with the latter the annual net return drops to -40.52% and the final net capital to $ 4.98.

N=7	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	30.78	30.10	-15.99	-16.27	14.18	14.14	4.44
		15	26.39	25.61	-17.88	-18.25	0.16	0.16	5.92
		25	64.30	62.48	-6.32	-6.72	0.24	0.24	5.62
	20	10	269.40	267.44	15.78	15.66	63.98	63.98	1.48
		15	103.68	101.14	0.54	0.17	63.49	63.49	4.73
		25	174.88	163.75	8.62	7.57	61.10	60.90	13.01
CALMAR	5	10	2.51	2.07	-41.99	-43.64	7.98	7.94	38.13
		15	0.65	0.62	-52.49	-52.87	5.75	5.51	10.79
		25	3.07	2.83	-40.25	-40.96	5.55	5.34	15.96
	20	10	18.15	17.07	-22.29	-22.99	24.98	24.70	12.12
		15	1.17	1.03	-48.17	-49.15	17.00	16.23	25.27
		25	6.89	6.31	-32.65	-33.53	24.70	23.97	17.59

Table 35: Dash-Coin for SARSA with N=7. Excel elaboration.

In conclusion, both for QL and SARSA there are no great differences in performance when the parameter N varies. As regards the QL algorithm, there are no substantial differences even on the choice of L or the degree of exploration. The only difference is that the results obtained with the Calmar ratio are lower than those obtained with the Sharpe ratio, exactly the opposite of the SARSA

algorithm which, albeit with lower average values than QL, offers – with some settings – greater investment security. If I were to advise a possible investor which strategy to choose to apply to Dash-Coin, I would tell him to use the SARSA algorithm and the Sharpe ratio with L = 20 as a reward function since it was the only one to always obtain positive returns. I would avoid using the Calmar ratio because, although the inputs are the same as SR, it concentrates several selling actions when the price falls, accumulating a loss which it then maintains for the rest of the investment time.

4.4 ETHEREUM

From table 36, which contains the results for QL for N = 1, we show the greatest yield and gain of the entire study of this thesis. With L = 20, ε = 0.15 with SR the annual net return is equal to 129.24% with a net final capital of $ 13280.01 and an over 100 percentage equal to 91.49%. These values are the highest compared to those obtained with the other configurations, a symptom of correct signal processing on the market trend. In the figures below I will illustrate this result better. However, in general with the QL algorithm there were no negative returns: on average with SR as a reward function, the annual net return is equal to 72.51% with a net final capital of $ 4976.67 and an over 100 percentage equal to 83.35%, percentage for the equity line practically identical for CAR (83.30%) but with a net return and a net final capital that is decidedly lower – respectively 39.04% and $ 779.94. As for the number of average annual operations, with the increase in exploration they increased from, on average, 14.55 for ε = 0.1 to 22.06 for ε = 0.25. To conclude with the first table, I did not find any particular preferences for the exploration value, while as regards L for the Sharpe ratio, the results obtained with L = 20 were significantly higher than those obtained with L = 5. In numbers: average net return and net final capital for L = 20 – 103.25% and $ 8929.51 – average net return and net final capital for L = 5 (41.76% and $ 1023.82).

N=1	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	251.76	233.75	16.96	15.50	58.81	57.65	16.80
		15	2128.95	1993.65	68.03	66.16	91.17	91.17	14.93
		25	946.14	844.06	46.42	43.61	88.66	88.52	25.79
	20	10	12807.51	12208.35	127.83	125.99	91.49	91.49	10.86
		15	14145.15	13280.01	131.70	129.24	91.49	91.49	14.25
		25	1454.85	1300.18	57.51	54.54	79.96	79.78	25.45
CALMAR	5	10	755.01	713.92	40.92	39.59	94.10	93.96	12.73
		15	449.74	427.09	29.06	27.94	82.80	82.52	11.71
		25	1246.71	1184.09	53.44	52.10	90.14	89.96	11.71
	20	10	279.25	258.06	19.04	17.45	63.78	63.09	17.82
		15	902.47	845.15	45.25	43.64	94.05	93.72	14.76
		25	1400.35	1251.31	56.50	53.54	76.66	76.57	25.28

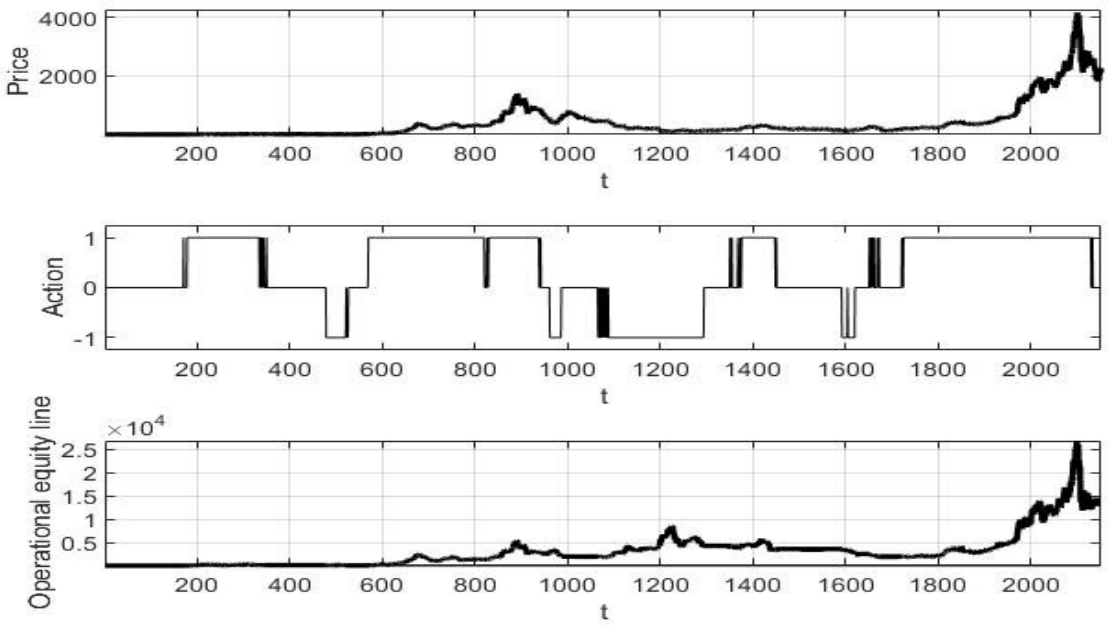
Table 36: Ethereum for QL with N=1. Excel elaboration.

Looking now at the results obtained with N = 7, I did not find such a high performance as in the case of N = 1. With the Calmar ratio, the performance remained rather unchanged compared to N = 1; with the increase of ε the returns increased but, on average, the annual net return remained equal to 38.03% with a net final capital of $ 764.06. The over 100 percentage also remained unchanged at 83.92% while, with the Sharpe ratio, it decreased the average value to 74.89%; the annual net return and net final capital also decreased – 44.10% and $ 1258.15, respectively. As for the number of annual operations, the situation remains the same as for N = 1 with more operations as ε increases and with an average number of annual operations equal to 17.31. Therefore, the best performance is obtained with the same setting for N = 1, L = 20, ε = 0.15 and with the Sharpe ratio: the annual net return is equal to 81.98% with a final net capital of $ 3374.07 and a percentage over 100 not that high but equal to 74.78%.

N=7	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	417.65	394.63	27.54	26.31	71.28	71.28	12.93
		15	384.22	348.60	25.74	23.68	63.03	50.63	22.12
		25	388.15	348.51	25.96	23.67	90.68	90.68	24.50
	20	10	2621.85	2460.88	74.34	72.47	83.73	83.68	14.46
		15	3674.67	3374.08	84.65	81.99	74.97	74.78	19.40
		25	691.64	622.24	38.97	36.49	86.39	78.28	24.16
CALMAR	5	10	397.26	379.01	26.43	25.42	87.75	87.47	10.71
		15	377.81	361.54	25.35	24.42	78.85	77.27	10.03
		25	1239.11	1148.84	53.40	51.44	90.92	90.78	17.17
	20	10	539.72	512.51	33.19	32.02	86.68	84.82	11.73
		15	611.37	570.23	36.04	34.44	75.83	74.94	15.81
		25	1798.00	1612.23	63.43	60.42	88.45	88.26	24.65

Table 37: Ethereum for QL with N=7. Excel elaboration.

In the figures below we can analyse better the performance of the QL algorithm with Sharpe ratio, N = 1, L = 20 and ε = 0.15. First of all, we can immediately see how the algorithm was able to analyse market fluctuations and develop the right actions to increase the initial capital. Apart from an initial period where he tried to perform some actions, but which mostly remained out of the market, from t = 600 approximately he exploited well the price changes so as to bring the capital already to approximately $ 5000. The situation remained largely unchanged, as can also be seen in the figure of the K = 500 equity lines, up to t = 1800, when, taking advantage of the surge in the price of Ethereum, it immediately went long, taking the investment beyond the $ 25000. As noted also with previous cryptocurrencies, the QL algorithm fails to exit in time when there is a collapse of the markets, as in this case. After the peak that was recorded towards the end of the recorded period, the algorithm continued to stay long thus losing nearly $ 10000 but still ending with a final net capital of $ 13280.01. Also, in the figure of the equity lines we can see how they are all well distributed around the average, and in the graph of the average equity line we can see even better the surge in price that occurred at t = 1800. The positive aspect is that, after the peak of performance, the algorithm was able to change position fairly in time, avoiding losing even more capital as it had previously happened with other configurations.



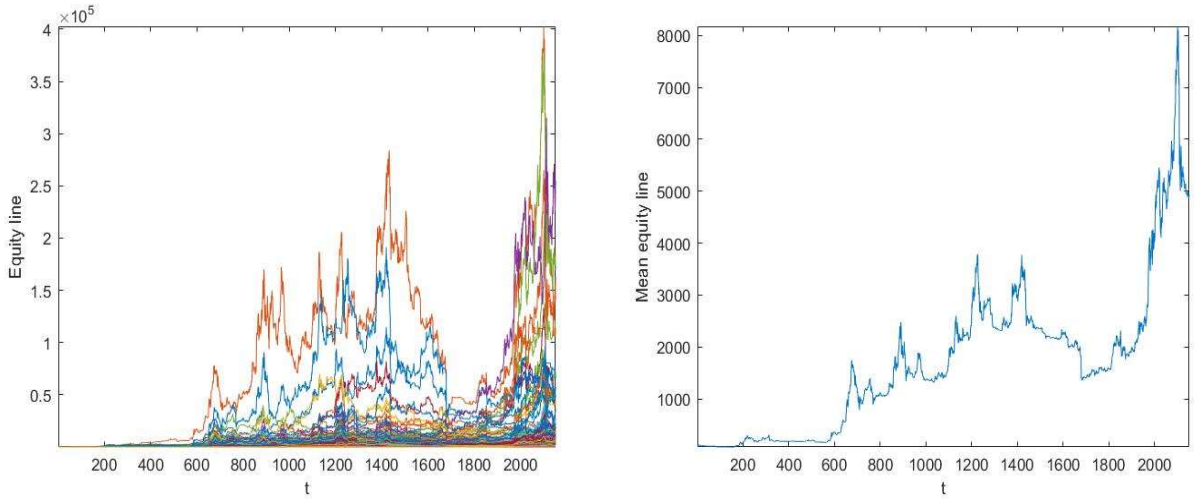


Table 38: Ethereum, Sharpe ratio QL with N=1, L=20 and ε=0.15. Above the three layers panel. Bottom Left the resulting equity lines. Bottom right average equity line. Source: MATLAB.

Let us now analyse the results obtained with the SARSA algorithm. In general, the performances obtained with this algorithm are worse than with QL. Looking at the first table for N = 1 we can see the first negative returns with the Sharpe ratio: for L = 5 and ε = 0.1 the net annual return is equal to -13.44% and the final net capital of $ 42.70, with a percentage over 100 equal to 0.14%; in this case the algorithm was always at a loss. On the other hand, the highest performance is with the Calmar ratio for L = 20 and ε = 0.15 with a net annual average return of 45.93%, a net final capital of $ 1053.55 and the over 100 percentage equal to 99.44%, the algorithm in this case it is always in profit. Using the Calmar ratio as a reward function, the performances are better than the Sharpe ratio: for CAR the average annual net return is equal to 24.53%, the final net capital on average is $ 419.38 and the percentage over 100 is equal to 70.81%; for SR the average annual net return is equal to 10.25%, the final net capital on average is of $ 277.63 and the percentage over 100 is equal to 47.80%. As for the average number of annual operations, CAR executes on average about twice as many transactions as SR – 19.66 compared to 8.14 – but, given the good returns, they are more than justified.

N=1	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	43.09	42.70	-13.31	-13.44	0.14	0.14	2.04
		15	75.49	70.80	-4.66	-5.69	83.17	82.52	14.42
		25	116.95	114.42	2.69	2.31	8.51	8.37	4.92
	20	10	936.42	922.59	46.17	45.80	91.49	91.49	3.39
		15	362.59	351.00	24.43	23.75	89.77	89.68	7.30
		25	178.99	164.28	10.38	8.79	15.11	14.60	19.51
CALMAR	5	10	603.48	591.46	35.67	35.20	86.47	86.01	4.58
		15	309.42	302.27	21.13	20.65	51.14	50.02	5.26
		25	218.14	193.59	14.15	11.86	39.80	39.10	26.98
	20	10	308.22	279.77	21.05	19.07	78.75	76.71	21.89
		15	1053.55	927.49	49.12	45.93	99.44	99.44	28.68
		25	250.97	221.70	16.90	14.47	78.57	73.59	28.00

Table 39: Ethereum for SARSA with N=1. Excel elaboration.

Analysing the results obtained with N = 7, we note a further deterioration in the performance provided by the Sharpe ratio. If with L = 20 we have an average annual net return of 23.11% with L = 5, the average annual return is negative equal to -14.03%, and even in the first case the percentage over 100 has a good value of 74.98%, while in the second case it is almost totally in loss of 2.24%. As regards the Calmar ratio, all the configurations of the parameters have produced positive returns and specifically the average annual net return is equal to 24.63%, the final net capital on average is of $ 441.43 and the percentage over 100 is equal to 67.33%; values similar to those obtained with N = 7. Also, in this case the average number of shares for the Calmar ratio is double that of the Sharpe ratio – 19.45 compared to 8.37 – but, even in this case, they are more than justified given the positive return of all the configurations.

N=7	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	27.13	26.80	-19.91	-20.07	0.09	0.09	2.72
		15	28.15	27.68	-19.40	-19.63	0.47	0.47	3.74
		25	88.34	86.84	-2.09	-2.37	6.53	6.15	3.91
	20	10	449.24	438.66	29.13	28.61	83.03	83.03	5.45
		15	205.29	184.51	13.02	10.99	72.21	72.17	24.33
		25	479.63	461.85	30.58	29.74	69.84	69.74	8.68
CALMAR	5	10	1104.39	1080.39	50.43	49.87	92.87	92.87	4.93
		15	269.53	240.57	18.36	16.09	47.28	45.51	25.67
		25	172.08	151.38	9.67	7.30	38.43	37.45	29.07
	20	10	252.72	219.81	17.07	14.33	66.60	64.79	31.45
		15	396.19	380.82	26.37	25.52	78.39	78.16	9.01
		25	622.79	575.64	36.47	34.66	85.51	85.19	17.85

Table 40: Ethereum for SARSA with N=7. Excel elaboration.

In conclusion, with Ethereum the algorithm performed at its best, leading to a very high profit compared to other cryptocurrencies. As for the QL algorithm, it performed better with the Sharpe ratio, bringing an average annual net return of 58.30%, much better than the 38.54% of the Calmar ratio. Instead, the opposite happened with the SARSA algorithm, with SR the yield was only 7.40% while with Calmar 24.58%. In general, I could observe that with SR the best performances were obtained with L = 20, for all the parameter configurations; instead, with CAR I have not noticed a preference of parameters between L or ε. If I were to advise an investor what to do, I would absolutely recommend using the QL algorithm with the Sharpe ratio as a reward function because, as I highlighted earlier, it allows an average annual gain of more than 100%.

4.5 LITECOIN

Tables 41 and 42 report the results obtained with the QL algorithm for N = 1 and N = 7. For N = 1 there are generally better returns with Sharpe, especially with L = 20. In fact, if we observe the returns with L = 5, two out of three are negative compared to the average annual net returns of L = 20 equal to 17.70%. With the Calmar ratio all configurations provide negative returns, the annual net return is equal to -17.75 with an average net final capital of $ 31.15 and an over 100 percentage equal to 28.86%. All these low values do not allow the possibility of investment with the Calmar ratio. On the other hand, with Sharpe the percentage over 100 for L = 20 and ε = 0.15 is equal to 96.04%, this means that for 96% of the time the algorithm is in the earning phase thus allowing a good investment.

N=1	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	33.74	31.24	-14.81	-15.77	10.30	9.66	15.04
		15	175.25	161.84	8.63	7.36	60.48	59.31	15.63
		25	102.76	90.88	0.40	-1.40	61.33	55.64	24.19
	20	10	349.12	331.60	20.25	19.34	68.77	68.77	10.03
		15	640.80	609.10	31.51	30.53	96.08	96.04	10.03
		25	142.00	123.99	5.31	3.22	47.39	46.59	26.84
CALMAR	5	10	18.41	17.08	-22.09	-22.94	17.09	16.57	14.60
		15	12.90	11.69	-26.06	-27.14	15.52	13.86	19.32
		25	23.87	20.46	-19.05	-20.86	12.24	11.47	30.23
	20	10	27.78	26.37	-17.21	-17.85	41.54	40.97	10.18
		15	77.34	72.90	-3.72	-4.56	62.67	60.32	11.65
		25	44.05	38.41	-11.39	-13.16	35.76	29.98	26.99

Table 41: Litecoin for QL with N=1. Excel elaboration.

The situation with N = 7, as we can see from the table below, is similar to that with N = 1. Indeed, the performance with the Sharpe ratio worsened, passing from a net positive return with N = 1 equal to 7.21% to an average net negative return with N = 7 equal to -7.95%. It is noted that SR, when more past returns are taken into account – in our case L = 20 – worsens the performance of the algorithm as epsilon increases. On the other hand, with the Calmar ratio the performance slightly improved but remained negative: on average, the net annual return is equal to -15.51% and the final net capital is $ 42.75. Therefore, these results suggest that Sharpe works better when the state structure contains information only about the day before trading rather than over several past days, while Calmar works exactly the opposite.

N=7	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	35.49	32.74	-14.20	-15.22	11.50	10.65	15.97
		15	51.87	48.06	-9.25	-10.27	13.08	11.42	15.08
		25	14.27	13.06	-25.01	-25.99	7.57	2.35	17.44
	20	10	316.59	301.02	18.57	17.69	67.84	67.76	10.05
		15	383.20	355.64	21.97	20.63	68.73	68.37	14.78
		25	6.72	5.69	-32.91	-34.54	0.57	0.57	32.82
CALMAR	5	10	26.86	25.55	-17.65	-18.26	21.08	20.28	9.90
		15	6.59	5.76	-33.08	-34.40	11.21	10.44	26.44
		25	25.87	22.35	-18.11	-19.86	11.86	10.00	28.80
	20	10	32.53	30.53	-15.28	-16.08	41.72	40.87	12.56
		15	84.75	78.45	-2.42	-3.52	64.27	61.47	15.21
		25	108.91	93.86	1.27	-0.93	66.29	64.47	29.39

Table 42: Litecoin for QL with N=7. Excel elaboration.

Tables 43 and 44 show the results for Litecoin obtained using the SARSA algorithm. In general, the performance of this algorithm is negative and for N = 1, there is only one setting for which there is a positive result: SR, L = 5 and ε = 0.1 offers an annual average net return of 4.36% and a net final capital of $ 133.60. These parameters allow a good investment opportunity since the algorithm is in profit for 96.40% of the time and the average annual operations are 2.65. As for the Calmar ratio, the algorithm works very badly: it performs on average 19.66 operations per year to obtain an average over 100 percentage of 16.26% and an average annual net return of -41.23% with an average final net capital of only $ 3.58.

N=1	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	135.39	133.60	4.57	4.36	96.40	96.40	2.65
		15	101.56	99.74	0.23	-0.04	96.00	5.78	3.54
		25	55.36	52.98	-8.35	-8.94	6.26	6.26	8.55
	20	10	79.35	77.59	-3.35	-3.67	11.31	10.83	4.42
		15	45.56	44.09	-10.95	-11.38	2.67	2.46	6.49
		25	46.82	44.63	-10.59	-11.22	19.03	18.95	9.29
CALMAR	5	10	1.23	1.18	-47.72	-48.06	12.20	11.72	8.55
		15	2.62	2.50	-41.54	-41.95	13.86	13.74	9.44
		25	1.88	1.62	-44.36	-45.57	10.42	10.10	29.35
	20	10	12.17	10.67	-26.70	-28.10	32.12	31.72	25.81
		15	4.16	3.88	-37.43	-38.07	14.59	14.55	13.72
		25	1.88	1.61	-44.34	-45.62	15.88	15.72	31.12

Table 43: Litecoin for SARSA with N=1. Excel elaboration.

N=7	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	123.57	122.65	3.18	3.06	96.56	96.56	1.48
		15	120.88	118.01	2.84	2.48	96.40	96.23	4.73
		25	95.30	93.73	-0.71	-0.95	1.86	1.82	3.25
	20	10	55.99	51.80	-8.22	-9.27	10.17	9.92	15.37
		15	95.63	90.75	-0.66	-1.43	14.18	13.28	10.35
		25	70.13	64.87	-5.11	-6.20	2.92	2.31	15.37
CALMAR	5	10	0.72	0.62	-51.79	-52.76	10.44	9.83	27.03
		15	0.35	0.31	-56.63	-57.44	7.57	7.41	25.26
		25	3.43	2.91	-39.24	-40.71	12.22	11.82	32.64
	20	10	5.94	5.10	-34.10	-35.56	17.81	15.94	29.99
		15	1.47	1.27	-46.39	-47.51	11.94	11.94	27.92
		25	3.28	2.65	-39.63	-41.51	25.70	25.01	42.10

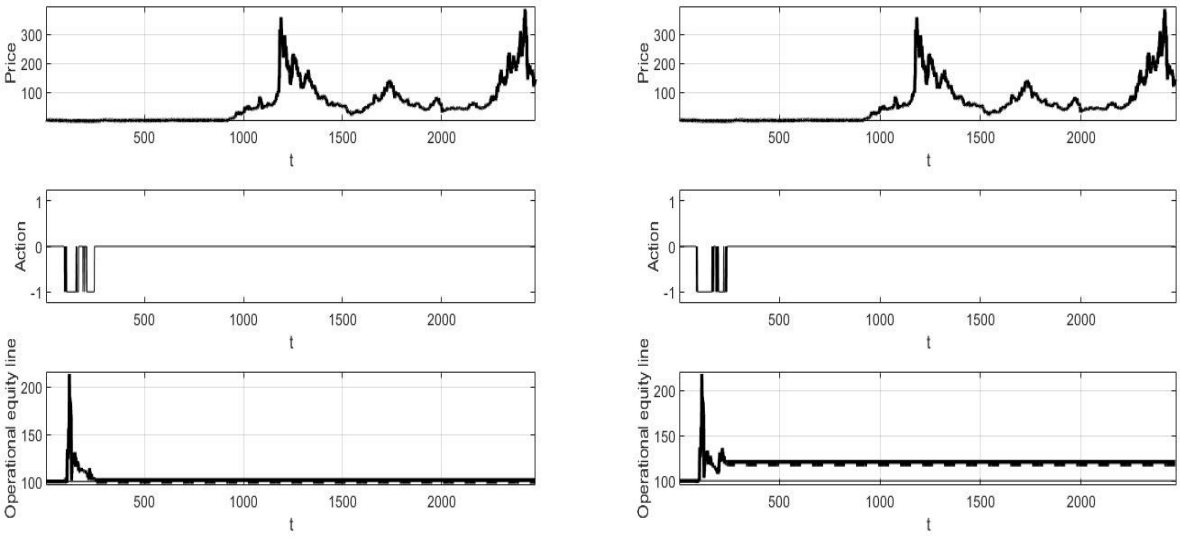
Table 44: Litecoin for SARSA with N=7. Excel elaboration.

Moving on to the results obtained with N = 7, we notice a partial improvement in returns, probably due to the fact that the state vector works with a week of returns and, therefore, manages correctly the market information. In fact, on average, with the Sharpe ratio net annual returns increased with respect to N=1 by 36.69% with a net final capital of $ 90.30. However, only two settings have positive performance, and they are with L = 5 and ε = 0.1 and ε= 0.15. In this case, by increasing the exploration, the algorithm does not offer better performance. Also, because on average the number of annual operations with the Sharpe ratio, L = 5 are decidedly low equal to 3.15 compared – for example – to those carried out with L = 20 equal to 13.70. As I will show in the figures below, using the SARSA algorithm, the fewer operations are carried out, the higher the yield. In fact, all the results

obtained with the Calmar ratio, N = 7 obtained negative net returns, on average equal to -45.91% with a final net capital of $ 2.14 and the number of annual operations is decidedly high equal to 30.82.

The following figures show the operations and equity lines of four parameter settings for the SARSA algorithm. Top row left SR, N = 1, L = 5, ε = 0.15; top row right SR, N = 7, L = 5 and ε = 0.15; bottom row left CAR N = 1, L = 5 and ε = 0.15; bottom row right CAR N = 7, L = 5, ε = 0.15. In the first two graphs above we can see the reason for the few operations performed with SARSA: only in the first period up to t = 250, the algorithm went short to take advantage of a possible fall in prices. As always, after an initial peak in return he lost a large part of the gain only that with N = 7 he managed to stop a little earlier and execute two more actions that brought the capital up to $ 118.01. Instead with N = 1 we can see how the gross over 100 percentage is equal to 96% while the net percentage is equal to 5.78%. As we can see from the figures, this is due to the capital which, at the time of the last action, before going out of the market, reached $ 101.56 gross and $ 99.74 net; therefore, the gross equity line always remained above 100 while the net one always below 100.

However, looking at the figures below, with the Sharpe ratio we can see the reason why the algorithm has decided to remain out of the market for a large part of the market. Unfortunately, SARSA with this series of prices has never been able to correctly process the price fluctuations from t = 1000 onwards: in fact, with the Calmar ratio it has performed many actions in that period bringing the initial capital almost to zero. Maybe a different setting of the parameters could lead to a more correct interpretation of the market signals.



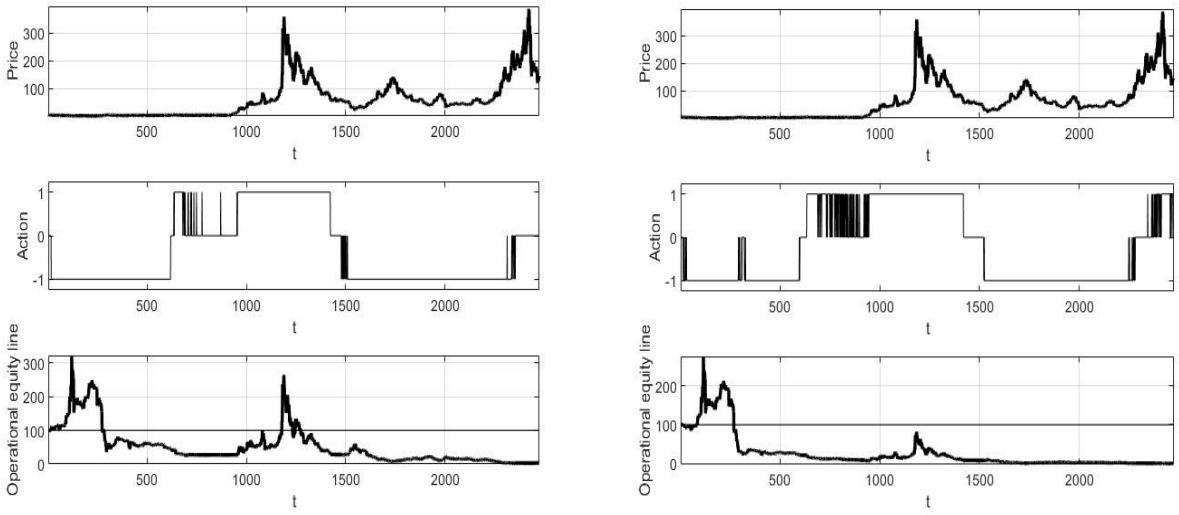


Table 45: Litecoin for SARSA; Top left SR, N = 1, L = 5, ε = 0.15; Top right SR, N = 7, L = 5 and ε = 0.15; Bottom left CAR N = 1, L = 5 and ε = 0.15; bottom right CAR N = 7, L = 5, ε = 0.15. Source: MATLAB.

In conclusion, with Litecoin, the QL algorithm performs better than SARSA with both reward functions and, in particular, with L = 20 it offers good returns and good investment opportunities. The results obtained with Calmar ratio are all negative and it has never been able to properly exploit the price fluctuations from t = 1000. If I were to advise an investor what to do, I would tell him to use the QL algorithm with the Sharpe ratio as a reward function and, in particular, I would recommend using the following settings: N = 1, L = 20, ε = 0.15. As we can see from the graph below, some operations have been carried out corrected by t = 1000 or since the most important market fluctuations occurred.

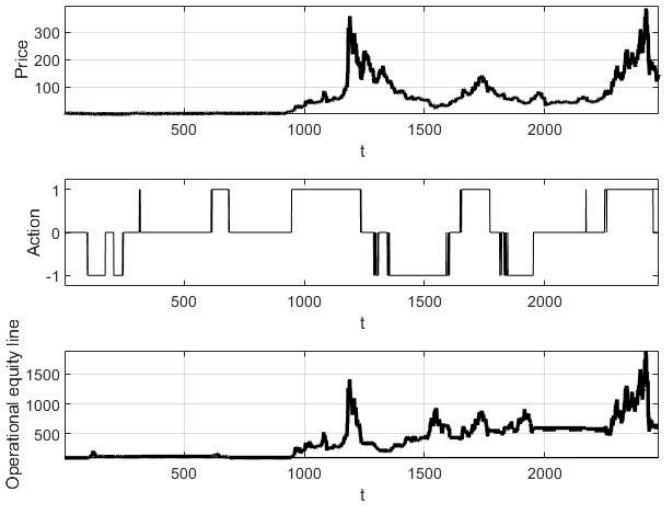


Table 46: Litecoin for QL, Sharpe ratio with N=1, L=20, ε=0.15. Source: MATLAB.

4.6 MONERO

Tables 47 and 48 show the results of applying the QL algorithm to the Monero cryptocurrency. We can see how both for N = 1 and for N = 7 the Sharpe ratio is able to better grasp the operation with low ε values. Otherwise, the Calmar ratio – preferring the exploratory action – behaves better, allowing to achieve higher performances. Furthermore, as seen with most cryptocurrencies by applying the QL algorithm, performance improvements are also appreciated by increasing the value of L, since the reward function is more sensitized to the volatility of the period as it is no longer considered 5 but 20 days of market return. In fact, the best configuration of the parameters is obtained with SR, N = 1, L = 20 and ε = 0.1 with an annual net return of 39.84%, a net final capital of $ 1025.52 and an over 100 percentage equal to 94.22% with, on average, 10.47 operations per year. These high values offer an excellent investment opportunity and are well above the average Sharpe ratio with the QL algorithm: on average, the net return is equal to -3.56% with a net final capital of $ 194.65 and an average number of annual operations equal to 17.21. As for the Calmar ratio, instead, I could not notice a yield preference between the exploratory action or the greedy action, as with SR the performance with L = 20 is much higher than L = 5. Counting all the results, for L = 5 the average net year return is equal to -11.40%, the average final net capital is $ 50.28, the average over 100 percentage is equal to 39.00% with an average of 16.69 annual operations; while for L = 20 the average net year return is 8.74%, the average final net capital is $ 307.07, the average over 100 percentage is 63.94% with an average of 17.73 operations per year.

N=1	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	59.33	54.19	-7.41	-8.64	53.25	51.92	17.84
		15	40.87	36.83	-12.36	-13.70	39.47	38.67	20.50
		25	7.50	6.41	-31.75	-33.32	1.90	1.82	30.82
	20	10	1025.52	971.78	40.96	39.84	94.22	94.22	10.47
		15	326.85	304.18	19.08	17.83	91.68	91.60	14.16
		25	11.42	9.73	-27.38	-29.08	4.28	3.80	31.56
CALMAR	5	10	42.67	41.06	-11.80	-12.30	54.55	54.38	7.52
		15	56.88	53.27	-7.98	-8.87	43.27	41.78	12.83
		25	86.09	79.93	-2.18	-3.25	52.48	49.86	14.60
	20	10	37.43	35.12	-13.49	-14.30	57.37	56.04	12.54
		15	152.39	144.08	6.41	5.53	80.65	79.23	11.06
		25	565.47	513.76	29.11	27.30	86.26	85.98	18.73

Table 47: Monero for QL with N=1. Excel elaboration.

N=7	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	36.70	34.14	-13.77	-14.69	48.81	44.84	14.19
		15	68.97	62.30	-5.34	-6.76	40.38	38.64	20.11
		25	27.87	24.67	-17.21	-18.69	5.27	5.27	23.95
	20	10	528.15	498.78	27.89	26.82	71.81	71.81	11.24
		15	344.52	311.77	20.07	18.31	69.58	68.93	19.81
		25	24.68	21.02	-18.68	-20.59	17.46	16.28	31.64
CALMAR	5	10	59.86	56.60	-7.30	-8.06	42.90	41.93	11.08
		15	102.59	97.60	0.38	-0.36	59.73	58.44	9.90
		25	61.47	56.36	-6.94	-8.12	41.68	40.47	16.99
	20	10	49.15	46.38	-9.96	-10.73	46.78	45.45	11.37
		15	273.29	246.65	16.01	14.27	75.64	75.19	20.24
		25	643.62	581.62	31.66	29.70	79.16	78.75	19.94

Table 48: Monero for QL with N=7. Excel elaboration.

In the two figures below, we can compare the results of the QL algorithm with N = 1, ε = 0.1 and, on the left L = 5 while on the right L = 20. We can immediately notice a significant difference in the number of actions performed by the two algorithms. It seems that for L = 20, using more days of market returns helped the algorithm to better evaluate market fluctuation to execute fewer but more targeted actions, not as in the other case where they are excessive and fail to take advantage of the fluctuations in the market price. In particular, in the period between t = 1000 and t = 1500, the algorithm on the right performs a few but well-targeted actions that lead to a net increase in capital; above all, it is able to take advantage of the drop in price that occurred at around t = 1500, remaining in a short position. Instead, in the graph on the left, it can be seen how many actions were performed with which the initial capital was almost zero. This huge difference is due to the fact that the algorithm considers more days of market returns and, therefore, processes past information better.

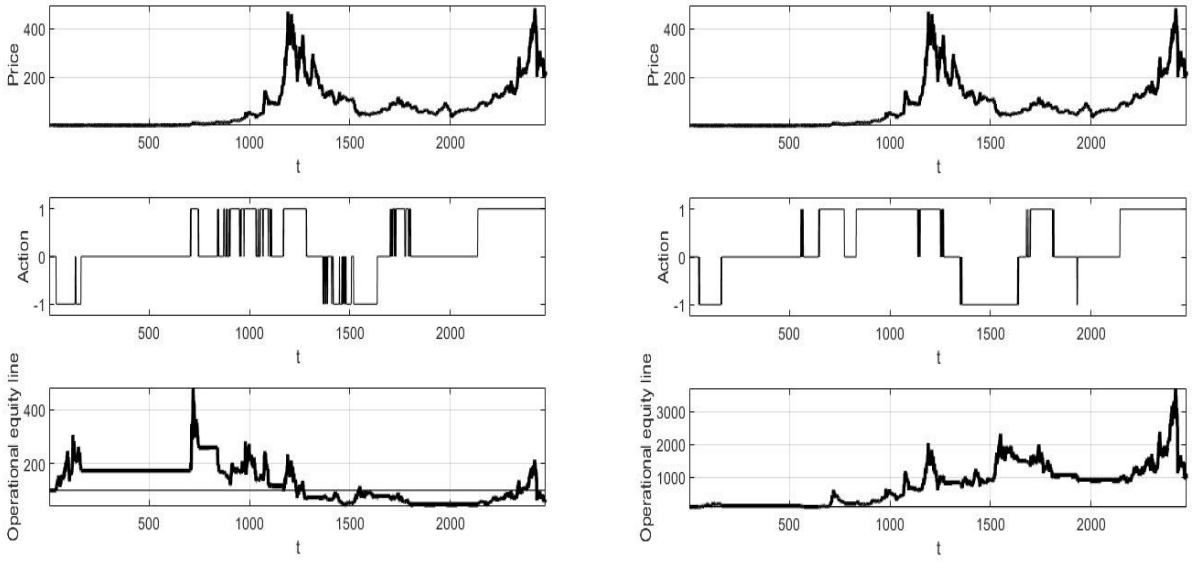


Table 49: Monero for QL, Sharpe ratio with N=1, ε=0.1. On the left L=5; On the right L=20. Source: MATLAB.

Tables 50 and 51 show the results using the SARSA algorithm. We can immediately see how the performances with this algorithm are lower than those obtained with QL. First of all, only five out of twenty-four configurations achieved positive returns and only with the Sharpe ratio as a reward function. Calmar ratio achieved very negative returns with each setting. In case of positive returns, they were obtained practically without performing any actions in the market: with SR, N = 1, L = 5 and ε = 0.1 the annual net return is equal to 6.18% with a final net capital of $ 150.15; the problem is that the average number of operations per year is equal to 0.59. Were carried out two actions to keep the return above the initial capital: in fact, the percentage over 100 is very high equal to 98.74%. Therefore, the trend identified is to increase operations as ε increases, penalizing the exploratory action compared to the greedy action and decreasing profits. The average annual net return for the Calmar ratio is equal to -21.17% with a net final capital of $ 23.69, an average net over 100 percentage of 25.64% with average annual operations of 15.82. As regards the results obtained with N = 7, there is a slight deterioration in performance due to the fact that, by increasing the number of returns within the vector representative of the initial state, there is greater persistence of negative returns in s_t.

N=1	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	150.59	150.15	6.22	6.18	98.55	98.55	0.59
		15	117.42	113.71	2.40	1.91	97.45	97.45	6.34
		25	33.98	31.87	-14.72	-15.52	1.62	1.13	12.54
	20	10	90.27	86.65	-1.50	-2.09	16.48	16.28	7.96
		15	135.26	125.77	4.55	3.44	65.86	65.82	14.31
		25	45.71	41.23	-10.90	-12.25	6.30	5.66	20.20
CALMAR	5	10	35.01	34.77	-14.34	-14.43	36.20	35.80	1.33
		15	33.99	32.92	-14.71	-15.11	36.65	36.40	6.34
		25	13.71	11.90	-25.40	-26.95	16.57	16.28	27.87
	20	10	44.76	43.67	-11.18	-11.50	43.80	43.52	4.87
		15	7.46	6.69	-31.80	-32.89	31.68	30.99	21.38
		25	35.15	32.64	-14.29	-15.22	41.01	40.20	14.60

Table 50: Monero for SARSA with N=1. Excel elaboration.

N=7	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	146.46	146.24	5.80	5.78	98.74	98.74	0.30
		15	110.80	110.48	1.53	1.48	98.34	98.34	0.59
		25	22.17	20.84	-19.96	-20.70	0.49	0.49	12.12
	20	10	71.11	70.56	-4.92	-5.02	0.00	0.00	1.48
		15	15.14	13.97	-24.35	-25.25	3.36	2.96	15.97
		25	66.15	60.85	-5.93	-7.08	59.50	59.17	16.41
CALMAR	5	10	21.05	20.82	-20.56	-20.69	21.65	21.37	2.22
		15	8.25	7.39	-30.82	-31.94	12.75	12.38	21.71
		25	13.94	12.24	-25.25	-26.67	8.90	8.42	25.55
	20	10	46.14	45.15	-10.80	-11.08	30.92	30.43	4.28
		15	10.48	8.90	-28.34	-30.04	11.45	10.56	32.05
		25	31.27	27.19	-15.78	-17.50	22.10	21.29	27.62

Table 51: Monero for SARSA with N=7. Excel elaboration.

In the two figures below, we can see what was said above about the operation of the SARSA algorithm. On the left there is the panel with the price of the cryptocurrency, the actions performed and the equity line of the algorithm with SR, N = 7, L = 5 and ε = 0.1. On the right, the parameters are SR, N = 7, L = 5 and ε = 0.25. As we can see from the figures, the algorithm is unable to process correctly the fluctuation in the price towards t = 1000. In fact, on the left, with the low level of ε it immediately tries to go short, obtains a good return and remains out of the market for the rest of the time. On the other hand, on the right it executes many more short actions at the beginning, reducing the initial capital by half and staying out the market for almost the entire remaining time because, towards the

end of the period, he tries to follow the rise in market prices with a set of long actions but, executed too late, they lose even more capital.

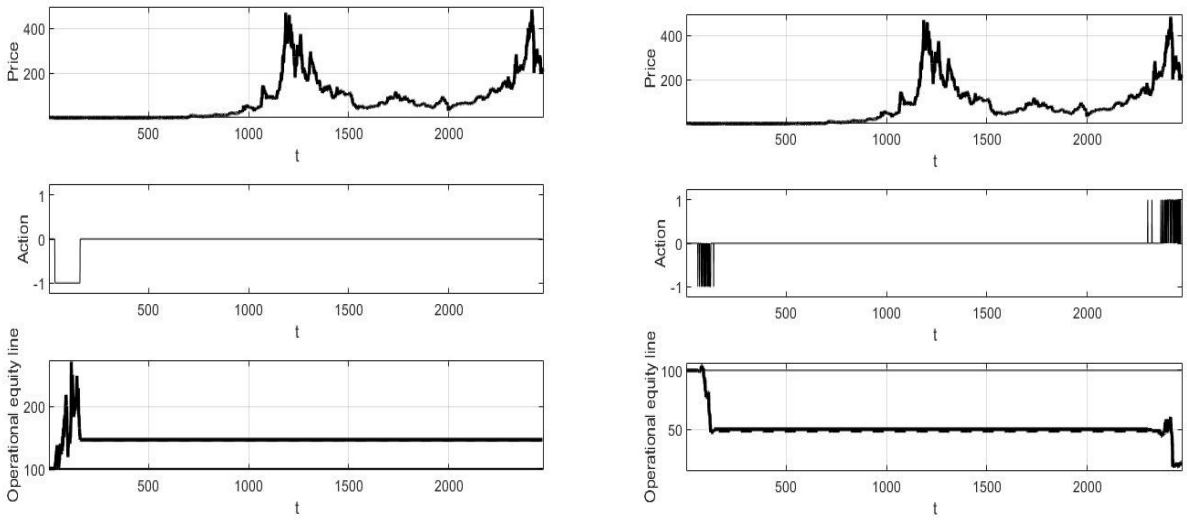


Table 52: Monero with SARSA, N=7, L=5. On the left ε=0.1; On the right ε=0.25. Source: MATLAB

In conclusion, with Monero the algorithm that works best is the QL offering, with L = 20, a good profit and an excellent investment opportunity. I could see how the preference for greedy action rather than exploration allowed for a better performance for the Sharpe ratio and, with relatively fewer shares than the average, obtained a good final capital. On the other hand, with the Calmar ratio the performance increased with the increase of ε. As for the yield number in the state vector SR it works better with a single yield, while the CAR gets higher values when N = 7. For a possible investor, there are good earning opportunities using the QL algorithm and, preferably, the Sharpe ratio as a reward function. The fundamental thing is to consider several days of market yield: in my case I used L = 20 but probably, by varying this value a little, you can get higher performances.

4.7 RIPPLE

Tables 53 and 54 show the results obtained by applying the QL algorithm to the cryptocurrency RIPPLE. As we can see, for both N = 1 and N = 7 there are no positive returns. The performance of the algorithm is decidedly negative, and no particular preferences are seen between the choice of parameters L or ε. As regards the reward functions, the results obtained with the Calmar ratio are slightly better than those obtained with the Sharpe ratio: with CAR the average annual net return is equal to -36.84%, the final net capital is 19.31 $ with an average percentage over 100 equal to 17.79% and an average number of annual transactions equal to 17.28; with SR, the average annual net return is equal to -51.90%, the final net capital is 6.95 $ with an average percentage over 100 equal to 1.03%

and an average number of annual operations equal to 20.81. The greater effectiveness of the Calmar ratio compared to the Sharpe may depend on the fact that, in periods of fluctuations without a precise trend, it is unable to map the volatility more completely by providing fewer effective indications to the algorithm; while CAR, using the maximum drawdown, is not penalized too much by the absence of trends.

N=1	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	4.15	3.86	-55.49	-56.30	1.39	1.39	24.42
		15	14.79	13.85	-38.50	-39.52	1.60	1.46	22.38
		25	2.59	2.35	-60.50	-61.47	0.42	0.42	33.07
	20	10	5.98	5.65	-51.15	-51.86	0.56	0.56	19.84
		15	1.36	1.26	-66.47	-67.11	0.35	0.35	25.44
		25	4.56	4.29	-54.39	-55.10	0.56	0.56	20.35
CALMAR	5	10	9.75	9.38	-44.69	-45.22	1.60	1.60	12.97
		15	18.75	18.01	-34.68	-35.34	0.77	0.70	13.48
		25	9.29	8.77	-45.36	-46.16	0.56	0.49	19.59
	20	10	3.35	3.18	-57.86	-58.41	1.18	1.11	17.55
		15	11.45	10.88	-42.38	-43.12	1.60	1.53	17.04
		25	29.36	27.82	-26.78	-27.77	7.60	7.04	18.06

Table 53: Ripple for QL with N=1. Excel elaboration.

N=7	QL		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	4.98	4.75	-53.52	-54.07	1.19	1.19	15.84
		15	3.64	3.39	-57.11	-57.88	0.49	0.49	24.01
		25	17.29	16.33	-36.13	-37.05	0.56	0.56	19.41
	20	10	3.25	3.09	-58.34	-58.86	1.68	1.68	16.86
		15	9.94	9.54	-44.54	-45.12	1.89	1.75	13.79
		25	15.66	14.99	-37.72	-38.42	14.35	1.96	14.30
CALMAR	5	10	10.57	9.93	-43.63	-44.52	3.77	3.77	21.17
		15	31.18	29.65	-25.72	-26.66	74.00	72.40	17.09
		25	23.59	22.53	-30.81	-31.62	13.21	11.88	15.56
	20	10	13.12	12.57	-40.43	-41.07	24.39	21.52	14.54
		15	39.29	37.76	-21.20	-22.00	47.24	44.51	13.52
		25	44.57	41.21	-18.63	-20.24	51.22	46.89	26.78

Table 54: Ripple for QL with N=7. Excel elaboration.

Tables 55 and 56 show the results obtained with the application of the SARSA algorithm with the same parameters seen for QL. With this algorithm the performances with the Sharpe ratio are higher than those with Calmar, the opposite of what we saw with QL. With SR the average annual net return is equal to -16.25%, the final net capital is $ 52.92 with an average percentage over 100 equal to 4.79% and an average number of annual operations equal to 13.50; with CAR the average annual net return

is equal to -60.31%, the final net capital is $ 3.62 with an average percentage over 100 equal to 7.95% and an average number of annual operations equal to 15.41. Regardless of the settings used, the algorithm fails to exploit environmental feedback to generate an effective trading strategy that leads to satisfactory profits: the price series has a large number of swings, and the algorithm is unable to adapt quickly to the market, thus producing a downgrade of profits in all cases.

N=1	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	45.61	43.27	-18.10	-19.19	0.28	0.28	17.80
		15	45.79	41.83	-18.02	-19.88	4.25	3.76	30.52
		25	33.01	30.80	-24.57	-25.89	0.00	0.00	23.40
	20	10	75.03	71.19	-7.05	-8.28	51.01	50.73	17.80
		15	58.91	57.77	-12.59	-13.03	1.39	1.11	6.61
		25	97.15	96.85	-0.73	-0.81	0.00	0.00	1.02
CALMAR	5	10	1.53	1.49	-65.48	-65.68	0.77	0.77	7.38
		15	1.01	0.99	-68.92	-69.09	0.00	0.00	7.38
		25	2.21	2.11	-62.09	-62.51	2.23	1.95	14.75
	20	10	1.07	1.04	-68.44	-68.73	0.77	0.77	12.21
		15	0.86	0.80	-70.21	-70.68	1.32	1.32	20.86
		25	1.52	1.44	-65.51	-66.01	2.09	2.02	19.08

Table 55: Ripple for SARSA with N=1. Excel elaboration.

N=7	SARSA		Equity Line		Annual Return		% over		n° ops.
Reward	L	ε (%)	G $	N $	G %	N %	G %	N %	#
SHARPE	5	10	41.41	41.19	-20.17	-20.27	0.00	0.00	1.53
		15	48.23	46.02	-17.00	-17.98	0.63	0.35	15.84
		25	57.31	52.11	-13.25	-15.33	0.14	0.14	32.18
	20	10	37.36	37.02	-22.24	-22.41	0.91	0.91	3.07
		15	25.73	25.00	-29.30	-29.82	0.28	0.14	9.71
		25	92.66	91.97	-1.93	-2.12	0.00	0.00	2.55
CALMAR	5	10	1.76	1.72	-64.33	-64.54	0.77	0.56	7.40
		15	9.67	9.23	-44.90	-45.55	19.92	17.19	16.07
		25	6.03	5.68	-51.15	-51.88	11.95	10.76	20.15
	20	10	3.43	3.32	-57.70	-58.03	5.66	5.31	10.46
		15	2.84	2.67	-59.69	-60.32	5.59	5.52	21.17
		25	14.03	12.92	-39.40	-40.67	55.00	49.27	28.06

Table 56: Ripple for SARSA with N=7. Excel elaboration.

In conclusion, both algorithms did not lead to the desired results. The reward functions did not work properly and failed to give the necessary feedback for the development of an adequate strategy. With these settings there is no investment opportunity, and it is strongly discouraged for anyone who wants to invest in RIPPLE to use these reward functions as I see no room for improvement. One possibility could be to use a different reward function because, in addition to the reported results, I tried to work

with different parameter settings but none of these offered satisfactory results. Otherwise, the RL approach could be modified in such a way as to also consider the volumes because Ripple, unlike Bitcoin, works with very low market prices – closed price for XRP 01-Jul $ 0.66 compared to $ 33572.12 of BTC – so much to ruin the performance of the algorithms.

5. FINAL CONSIDERATIONS AND OBSERVATIONS

In this thesis, I proposed some Reinforcement Learning-based approaches for online cryptocurrency trading. In the MATLAB environment, has been developed an algorithm to implement the Q-Learning and SARSA methods with the aim of applying it to the selected currencies, in order to create profitable investment strategies in real time. In this last chapter I will bring the observations and final considerations regarding the analysis carried out, with the related conclusions and the possibilities for future development.

5.1 Observations and Final Results

This first paragraph shows the final results for the application of the Q-Learning and SARSA algorithms by analysing the performance and providing some considerations. As analysed in the previous chapter, the first important aspect is that the cryptocurrencies Bitcoin and Ethereum are the ones with the best performances, while Ripple and Bitcoin Cash are the ones with the worst results. To support this, tables 57 and 58 present the averages of the net annual returns for the various parameters (N, L, ε) of the seven cryptocurrencies divided by algorithm and reward function. For almost all the currencies QL performs better than SARSA; the only two currencies that have higher yields with SARSA are Dash-Coin and Ripple both with the Sharpe ratio. As regards Dash-Coin, were obtained higher results because the SARSA algorithm performed few actions at the beginning of the analysed period, bringing the investment in positive and remaining out of the market for the remaining period. Looking at the reward functions, the Sharpe ratio offers better performance than the Calmar ratio. Actually, it can be noted that – in the event of negative returns for SR – with CAR they fall irreparably: this is the case of Bitcoin Cash, Dash-Coin and Litecoin for the QL, and for SARSA algorithm with the addition of Monero and Ripple. Speaking of Bitcoin and Ethereum, regarding the positive returns the Calmar ratio manages not to lose too much capital with the QL algorithm and, even with the SARSA algorithm, the performance is twice as good.

Average Annual Net Returns with QL (%)							
	Bitcoin	Bitcoin Cash	Dash Coin	Ethereum	Litecoin	Monero	Ripple
Sharpe	63.19	-20.57	-3.45	58.30	-0.37	-3.55	-51.90
Calmar	58.12	-71.42	-22.63	38.54	-16.63	0.90	-36.84

Table 57: Average annual Net Returns with QL divided by reward and cryptocurrency function. Excel elaboration.

Average Annual Net Returns with SARSA (%)							
	Bitcoin	Bitcoin Cash	Dash Coin	Ethereum	Litecoin	Monero	Ripple
Sharpe	25.98	-21.11	-1.03	7.40	-3.60	-5.76	-16.25
Calmar	50.79	-73.87	-42.19	24.58	-43.57	-21.17	-60.31

Table 58: Average annual Net Returns with SARSA divided by reward and cryptocurrency function. Excel elaboration.

Going back to analysing the best and worst performances of the study carried out, we can compare the equity lines in the computation phase between the cryptocurrency Bitcoin and Bitcoin Cash. The difference between the first two images above is substantial: in the first, the equity lines follow a well-defined and upward trend while in the second there is a lot of dispersion oriented downwards, thus bringing the capital towards a constant loss. In the two figures below, we can see the average equity line, ascending in the case of Bitcoin and descending in the case of Bitcoin Cash. The main reason for this systematic loss of capital is highlighted in the operational phase or when is defined the trading action system. The dispersion created with the iterations shows that the operational signals have not been well interpreted due to the market phases in which an evident trend is not defined, and the price fluctuates around similar values. The rapid fluctuation of prices means that the algorithm captures the signals for the change of operation in times that are not fast enough to obtain a positive profit. This does not guarantee that the operating ranges chosen will therefore be effective for these cryptocurrencies. The problem is related to the structure of the algorithm: due to poorly defined price trends and continuous price fluctuations that are not evident, the algorithm is unable to define the best investment strategy because it must continuously alter the behaviour held up to that instant, to learn a new one. So, the performance suffers because the algorithm slows down, and wrong actions are taken. A possible solution is to modify the value of the learning rate α to speed up the learning process. However, for the convergence conditions described in chapter 2, a too high value of α would not allow to obtain the necessary convergence to achieve the optimum of the value function. For this reason, this issue needs further study and needs to be deepened in the future.

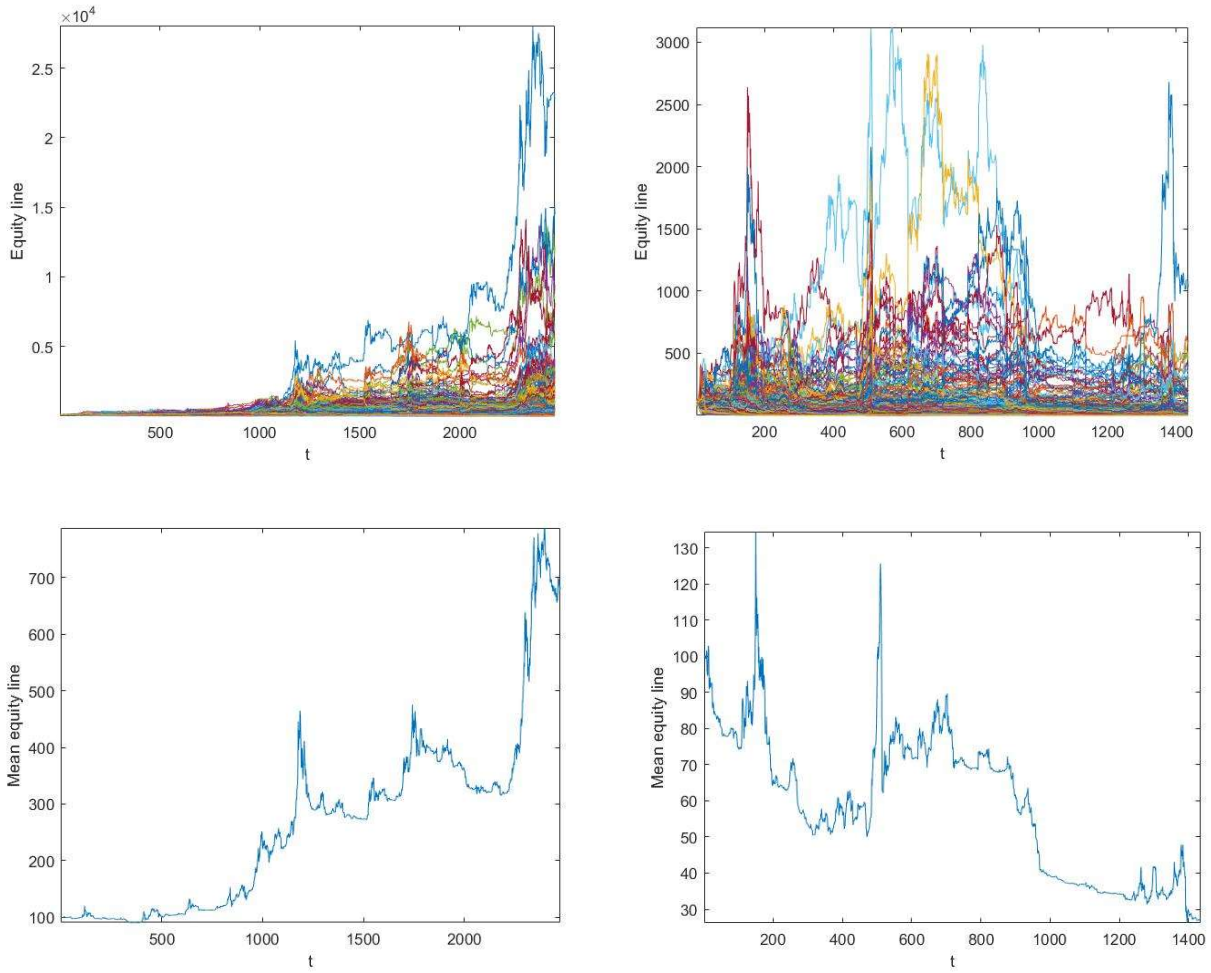


Table 59: Left: Bitcoin. Right: Bitcoin Cash. QL, Sharpe ratio with N=1, L=20, ε=0.25. Above the resulting equity lines and bottom the average equity line. Source: MATLAB.

A fundamental element for the algorithms concerns the operations performed during the time frame. The table below summarizes the average number of operations carried out by algorithm and by cryptocurrency. As we can see from the underlying values, the QL algorithm has always performed on average a greater number of actions than SARSA. As seen in the previous paragraphs, SARSA's investment strategy is rather static, consisting of a few shares and, in most cases, these shares are concentrated at the beginning of the period, staying out of the market for the remaining dates. As for QL, the greater number of shares costs more in terms of transaction costs but leads to a more profitable strategy, since the algorithm better assesses the price trend of a stock by assuming the correct positions in the market. Given that the cryptocurrency market is characterized by high volatility with a continuous fluctuation of prices without a well-defined trend, dynamism and speed of response to events are fundamental for an algorithm; for this reason, QL is more suitable than SARSA to operate in this market.

Average Annual Operations							
	Bitcoin	Bitcoin Cash	Dash Coin	Ethereum	Litecoin	Monero	Ripple
QL	14.22	20.57	16.33	17.07	18.46	17.21	19.04
SARSA	11.25	15.90	14.52	13.91	16.18	12.44	14.46

Table 60: Average annual Operations divided by algorithm and cryptocurrency function. Excel elaboration.

As a confirmation of what has just been said, we can see in the table below the over 100 percentages of the equity line divided by algorithm and cryptocurrency. In all seven currencies, the percentage is higher with the QL algorithm because – as mentioned above – it processes market signals more effectively by proposing a profitable investment strategy. However, it should be noted that in addition to Bitcoin and Ethereum – as already mentioned – are the two cryptocurrencies that have obtained the highest returns, even Dash Coin and Monero on average have acceptable over 100 percentages (above 50%).

Average Percentage Over 100 (%)							
	Bitcoin	Bitcoin Cash	Dash Coin	Ethereum	Litecoin	Monero	Ripple
QL	89.04	16.91	53.86	81.36	36.57	51.47	9.41
SARSA	81.07	3.96	23.43	56.14	22.51	35.51	6.37

Table 61: Average Percentage Over 100 divided by algorithm and cryptocurrency function. Excel elaboration.

Tables 62 and 63 show the over 100 percentages of the previous table, in this case divided by reward function. In confirmation of the above, all percentages are higher with the QL algorithm, except with Monero and Ripple which are slightly larger with SARSA. Looking now for the reward function, it is interesting to note how for Bitcoin and Ethereum – i.e., for the algorithms that have obtained excellent positive returns – the percentage over 100 is higher with the Calmar ratio although the average net return is lower. The reason for this high percentage is due to the operations performed with the SARSA algorithm: on average, the annual actions performed with the Calmar ratio are 5.26 compared to 17.23 performed with the Sharpe ratio, because the SARSA algorithm, as we can see from the figure 5.8 below, with all the parameter configurations, it performed actions at the beginning of the period and then remained long until the end of the analysed time. This made it possible to obtain an over 100 percentage that was twenty percentage points higher than the Sharpe ratio. In general, with the Sharpe ratio were obtained on average percentages over 100 higher than the Calmar ratio, even if acceptable values (> 50%) were achieved only for 6 configurations out of 14.

Average Percentage Over 100 with QL (%)							
	Bitcoin	Bitcoin Cash	Dash Coin	Ethereum	Litecoin	Monero	Ripple
Sharpe	85.17	32.37	62.26	79.12	41.43	43.98	1.03
Calmar	92.92	1.45	45.46	83.61	31.72	58.96	17.79

Table 62: Average Percentage Over 100 with QL divided by reward and cryptocurrency function. Excel elaboration.

Average Percentage Over 100 with with SARSA (%)							
	Bitcoin	Bitcoin Cash	Dash Coin	Ethereum	Litecoin	Monero	Ripple
Sharpe	72.70	6.68	33.32	43.20	30.07	45.38	4.79
Calmar	89.44	1.25	13.55	69.07	14.96	25.64	7.95

Table 63: Average Percentage Over 100 with SARSA divided by reward and cryptocurrency function. Excel elaboration.

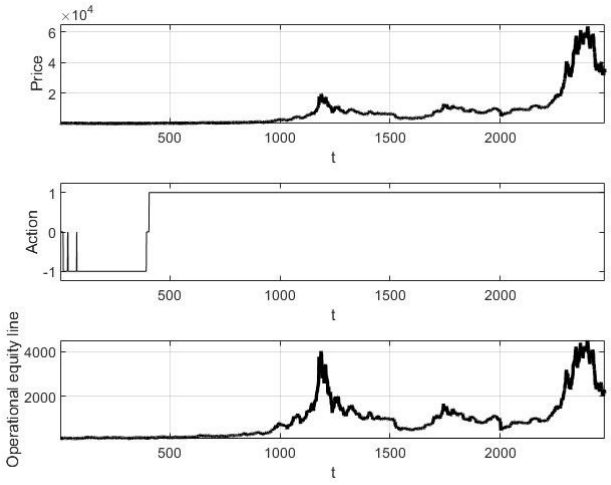


Table 64: Price, actions and equity line with SARSA. Source Matlab.

Analysing the Calmar ratio, it was chosen as an alternative to the Sharpe ratio to evaluate its effectiveness, given its wide use in the financial field. This index is constructed using the maximum drawdown of the equity line in the denominator and the average of the last returns obtained in the numerator. If there is no drawdown in the analysed period, the reward value for t becomes the average of the last returns L achieved. This index was chosen as a valid alternative to the Sharpe ratio because of its structure, to allow for a better comparison. Given the great volatility of the cryptocurrency market, I chose to use this index also to remedy the misinterpretation of the positive and negative deviations of the SR; in fact, with CAR when choosing not to invest or to stay out of the market, the equity line remains flat bringing the drawdown value to zero and this algorithm considers a drawdown value of zero to be better, rather than a positive one. Unfortunately, the Calmar ratio was not an excellent alternative: the poor operation and the inadequacy in grasping the operational signals did not allow to obtain good results in terms of performance and yield.

Moving on to the analysis of the various parameter settings, we can see in table 65 the summary of the net returns of the analysed cryptocurrencies divided by reward function and by parameters N and L. I chose to display only the results for the QL algorithm, since with SARSA low trading has led to poorer returns and “insincere” values for some cryptocurrencies. As regards the parameter N out of

28 combinations analysed, 39.28% of the time the parameter N = 1 offers higher yields than N = 7. It turns out that 17 times out of 28, more elements in the state vector perform better. Even if the computational effort is higher, the general performance of the algorithm benefits from it, allowing to make greater profit even in those situations where the earning opportunities are more complicated to exploit. Contrary to what was stated above, the Ethereum cryptocurrency offers a much better performance with N = 1, probably because with ETH the returns are on average good and there are more profit opportunities, so increasing the number does not generate better performance.

QL		Bitcoin		Bitcoin Cash		Dash Coin		Ethereum		Litecoin		Monero		Ripple	
Rwd	L	N=1	N=7	N=1	N=7	N=1	N=7	N=1	N=7	N=1	N=7	N=1	N=7	N=1	N=7
SR	5	61.3	71.5	-13.9	-14.5	-10.2	2.9	41.8	24.6	-3.3	-17.2	-18.6	-13.4	-52.4	-49.7
	20	55.2	62.9	-30.8	-23.1	-4.3	-2.2	103.3	63.6	17.7	1.3	9.5	8.2	-58.0	-47.5
CAR	5	71.5	57.2	-74.9	-63.3	-26.6	-30.7	39.9	33.8	-23.6	-24.2	-8.1	-5.5	-42.2	-34.3
	20	51.0	52.8	-74.1	-73.4	-13.9	-19.2	38.2	42.3	-11.9	-6.8	6.2	11.1	-43.1	-27.8

Table 65: Average annual Net Returns with QL for N and L. Excel elaboration.

Turning to the evaluation of the parameter L, for 9 combinations out of 28 the yields are higher in the case of L = 7, so the QL algorithm performs better in 67.86% of cases with L = 20. This means that a more correct evaluation of the actions taken in the market is more effective if is used a longer analysis period, thus using the last twenty trading days rather than a week. What has been stated can be seen in the results of the Monero cryptocurrency: with both reward functions, was obtained a positive return when the choice of the action to be performed in the market was made based on the returns of the last twenty trading days; instead, the return was negative when was used the last week of trading. Exactly the opposite of what has just happened with Bitcoin: the information obtained using a shorter analysis period is sufficient to obtain a more efficient investment strategy.

QL	Bitcoin		Bitcoin Cash		Dash Coin		Ethereum		Litecoin		Monero		Ripple	
ε	SR	CAR	SR	CAR	SR	CAR	SR	CAR	SR	CAR	SR	CAR	SR	CAR
10%	70.7	48.3	-16.6	-69.3	-4.0	-29.1	60.1	28.6	1.5	-18.8	10.8	-11.3	-55.3	-47.3
15%	58.6	58.8	-22.9	-72.9	-6.3	-27.3	75.3	32.6	12.1	-17.4	3.9	2.6	-52.4	-31.8
25%	60.3	67.2	-22.2	-72.1	0.0	-11.5	39.6	54.4	-14.7	-13.7	-25.4	11.4	-48.0	-31.4

Table 66: Average annual Net Returns with QL for ε. Excel elaboration.

Finally, it is evaluated if there are improvements for the parameter ε = {10%, 15%, 25%}. Table 66 summarizes the values of the parameter ε divided by cryptocurrency and reward function. It is noted that, for the Sharpe ratio, an increase in the exploratory action does not correspond to an improvement in performance. The degree of exploration in the RL is essential to experiment with new actions and the right compromise must be found between exploration and exploitation. With SR, a possible

development could be to find a possible value between 10% and 15% that can further improve the performance of the algorithm. On the other hand, with the Calmar ratio for all cryptocurrencies, as the degree of exploration increases, the yield of the algorithm also increases. In particular, with ε = 0.25 the best results were obtained on average, a symptom that the random choice of the investment strategy offers the algorithm more prolific earning possibilities than the greedy action. For the calmar ratio, a further evaluation could be made by verifying how much it is possible to increase the value of ε to further improve the good result already obtained.

5.2 Conclusions

The aim of this thesis is to apply a machine learning method known as Reinforcement Learning to the cryptocurrency market, in particular the already known Q-Learning and SARSA algorithms, testing their effectiveness and evaluating the investment possibilities. An advantage of using these automatic trading systems lies in the fact of being able to eliminate the psychological factor that unites all investors. As studied in the Behavioural finance course, a loss – psychologically – weighs more than a gain and too many gains often lead to a fallacious analysis of one's investments.[68] For this reason, the creation of these automatic systems that learn quickly the reinforcements and rewards provided by the surrounding environment, can lead to the resolution of these biases and market inefficiencies. Another fundamental issue was to find a valid alternative to the Sharpe ratio as a reward function and, given the wide and consolidated use of the latter, I considered it as a benchmark for evaluating the Calmar ratio. I chose this reward function because, given the great volatility of this market, I thought that taking advantage of the numerous drawdowns could lead to greater gains.

Basically, I showed how the application of the Q-Learning and SARSA algorithms to the seven cryptocurrencies: Bitcoin, Bitcoin Cash, Dash Coin, Ethereum, Litecoin, Monero and Ripple. The application has brought both gains and losses on the initial investment, results that I have widely exposed and commented. As described above, I applied different settings through two reward functions: the Sharpe ratio and the Calmar ratio. Were tested 48 different configurations with both highly positive and negative results: the use of the Calmar ratio as a reward function on average disappointed initial expectations. Although it is a widely used measure in the financial field, it is not effective in this market, making the strategies immobile and, especially with the SARSA algorithm, almost always out of the market. Due to its lower trading on the market, this algorithm achieved worse returns than QL. So, the best choice is the Q-Learning algorithm with the Sharpe ratio as a reward

[68] Behavioural Finance: Psychology, Decision-Making and Market; Ackert & Deaves; Ch. 5.

function. As for the parameters, 500 iterations offer adequate computational time; it is also recommended to use a week of past returns (N = 7) as information for the state vector, while for information processed for performance calculation it is more effective to consider the last twenty trading days rather than a week. Instead, for the degree of exploration ε I recommend setting it to 15%.

As for future developments and in support of what I mentioned in previous chapters, I think it is appropriate to find new reward functions to be used as an alternative to the Sharpe ratio and which are effective applied to this market. It would be appropriate to test the configurations of the QL algorithm with other cryptocurrencies – i.e., Tether, Cardano, Dogecoin, … – to assess whether there are further earning possibilities in addition to those highlighted with Bitcoin and Ethereum. Other intermediate parameter settings could lead to better performance and, as already highlighted in the analysis of the Ripple cryptocurrency, the introduction of some technical analysis indices, and also the processing of transaction volumes during market operations, could be effective. Finally, it would be convenient to develop a platform / app that applies the algorithm to the selected cryptocurrencies on a daily basis, offering the action to perform in the market more immediately so as to also be able to keep track of the entire investment.

BIBLIOGRAPHY

D. Chaum, Blind signature for untraceable payments, in R. L. Rivest and A. T. Sherman (eds), Advances in cryptology: proceedings of crypto 82, Springer book archive, 1983.

John Moody and Matthew Saffell, *Reinforcement Learning for Trading Systems and Portfolios*, Oregon Graduate Institute.

John L. Teall, *Financial Trading and Investing*, 2018, Elsevier Science editor.

Barto, A.G., Sutton, R.S. (2018). *Reinforcement Learning: An Introduction.* The MIT Press.

Corazza, M. and Fasano, G. and Gusso, R. and Pesenti, R. (2019). *A Comparison among Reinforcement Learning Algorithms in Financial Trading Systems.* University Ca' Foscari of Venice, Dept. of Economics Research. Paper Series No. No. 33/WP/2019.

Corazza, M. and Sangalli, A. (2015). *Q-Learning and SARSA: a comparison between two intelligent stochastic control approaches for financial trading*. Working Papers 2015:15, Department of Economics, University of Venice "Ca' Foscari", revised 2015.

David Greising, Laurie Morse: *Brokers, Bagmen, and Moles, Fraud and Corruption in the Chicago Futures Markets*, 1991, Wiley.

D. Knezevic, *Impact of blockchain technology in changing the financial sector and other industries*, March 2018, Montenegrin Journal of Economics.

Bhramrah Dipender, "*Blockchain internet of transaction*", June 2018, BPB Publications, p. 10.

J. Brito, *Beyond silk road: potential risks, threats, and promises of virtual currencies*, Testimony before the Senate Committee on homeland security and governmental affair, 18 November 2013.

R. Wu, *Why we accept Bitcoin*, Forbes, February 2014, www.forbes.com.

S. Capaccioli, *Criptovalute e bitcoin: un'analisi giuridica*, Giuffrè editore, Milan, 2015, quote at p. 254

Ackert & Deaves, *Behavioral Finance: Psychology, Decision-Making and Market*; Ch. 5

S. Nakamoto, *Bitcoin: a peer-to-peer electronic cash system*, October 2008, www.cryptovest.co.uk.

A. Caponera e C. Gola, *Aspetti economici e regolamentari delle "cripto-attività"*, in the Bank of Italy (edited by), *Questioni di Economia e Finanza (occasional papers)*, March 2019, www.bancaditalia.it.

Sicignano, Gaspare Jucan, "Bitcoin e riciclaggio", April 2019, Giappichelli, p. 111.

European Banking Authority, *Opinion on virtual currencies*, July 2014, pp. 16 – 21, www.eba.europa.eu.

European Central Bank, *Virtual currency schemes*, October 2012, p.16, www.ecb.europa.eu.

K. Corcoran, *Europol: i criminali ricorrono alle criptovalute per riciclare 5,5 miliardi di dollari in contanti di provenienza illecita in Europa,* February 2018, www.businessinsider.com.

M. Dabrowski (CASE) e L. Janikowski (CASE), *Virtual currencies and central banks monetary policy: challenges ahead,* July 2018, www.europarl.europa.eu.

European Insurance and Occupational Pensions Authority, *Avviso, L'ESMA, l'ABE e l'EIOPA informano i consumatori sui rischi delle valute virtuali,* 2014, www.eiopa.europa.eu.

Banca d'Italia, *Avvertenze sull'utilizzo delle cosiddette "valute virtuali"*, 30 January 2015, Rome, www.bancaditalia.it.

Chris Clark, Bitcoin Internals: A Technical Guide to Bitcoin, 31/06/2013.

Andreas M. Antonopoulos, Mastering Bitcoin, O' Reilly, 2014.

Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 14/12/ 2010.

Anonymous, *Che cosa sono le Altcoin e perché sono importanti*, maggio 2018, www.bitcoin.it

https://www.ilsole24ore.com/art/smart-contract-cosa-sono-e-come-funzionano-clausole-blockchain-ACsDo2P.

Bellini M., *Che cos'è e quali sono gli ambiti applicativi di Ethereum*, august 2019, www.blockchain4innovation.it.

Bellini M., "Cos'è *Dash (DASH),* valore di mercato e ambiti applicativi", www.*Blockchain4innovation.it*, 21 August 2018.

Gagliarducci C., *Ripple: cos'è, come funziona e quali differenze con il Bitcoin?,* may 2017, www.money.it.

Provenzani F., "*Dash*: cos'è e come comprare la criptovaluta rivale del Bitcoin", www.*money.it*, 11 December 2017.

Torchiani G., "*Altcoin*: cosa sono e come funzionano le criptovalute sorelle del Bitcoin", www.*Digital4Trade.it*, 2 February 2018.

Monero (XMR): cos'è e come fare *trading*", www.*tradingonline.com*, 2018.

Martin L. Puterman. Markov Decision Processes: Discrete Stochastic Dynamic Programming. New York, NY: John WileySons, 1994, 649.