UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MASTER

Design and Development of an Algorithmic Trading System for Cryptocurrencies using Reinforcement Learning and Deep Learning

JAIME JOSÉ PALOS PEREIRA

2020

TRABAJO DE FIN DE MASTER

Título:	Diseño y Desarrollo de un Sistema de Negociación Bursátil Algoritmica de Criptomonedas utilizando Aprendizaje por Refuerzo y Aprendizaje Profundo
Título (inglés):	Design and Development of an Algorithmic Trading System for Cryptocurrencies using Reinforcement Learning and Deep Learning
Autor:	JAIME JOSÉ PALOS PEREIRA
Tutor:	CARLOS A. IGLESIAS
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE MASTER

Design and Development of an Algorithmic Trading System for Cryptocurrencies using Reinforcement Learning and Deep Learning

2020

Resumen

El apredizaje automático está creciendo muy rápidamente y haciéndose hueco en casi todos los campos. Y la bolsa de valores no es una excepción.

Cada día, el llamado "Trading Algoritmico", modalidad de operación en mercados financieros caracterizada por el uso de algoritmos, gana más y más popularidad. Esta tecnología precisa de programas que ejecuten algoritmos que automaticen una estrategia de trading. Hemos ido viendo cada vez más gestores de activos y fondos de cobertura incorporan técnicas de aprendizaje automático para mejorar sus estrategias para realizar operaciones financieras. Y hemos visto incluso, fondos de cobertura completamente basados en programas informáticos que, de acuerdo con The Financial Times, ya se han unido a los que obtienen mejor rentabilidad [87].

Prueba del gran crecimiento del aprendizaje automático es que, hoy en día, los fondos de inversión y fondos de cobertura están contratando analistas cuantitativos para diseñar e implementar complejos modelos que permiten a instituciones financieras decidir el precio de los productos, realizar gestiones y analisis de riesgos, o incluso crear nuevas estrategias de comercio. Y la importancia de estos, se ve claramente reflejada en sus salarios de $250,000 anuales más bonus, o incluso algunos que superan los $500,000+ segun Investopedia. [116].

Durante el desarrollo de esta Tesis de Fin de Máster, estudiaré el estado actual de estas tecnologías, aprenderé acerca de la bolsa de valores, trading, aprendizaje automático, aprendizaje por refuerzo, trading algorítmico, etc. Y aplicaré dicho conocimiento para diseñar y desarrollar un sistema de trading algoritmico basado en técnicas de aprendizaje automático. Posteriormente, compararé los resultados obtenidos a otras técnicas de trading así como a los resultados obtenidos por otros trabajos relacionados. Por último finalizaré con posibles ideas de trabajo futuro.

Palabras clave:Trading Algoritmico, Aprendizaje Automático, Aprendizaje por Refuerzo, Redes Neuronales, Bitcoin, Criptomonedas

Abstract

Machine Learning (ML) is growing and it's rapidly making its way into almost every field. And the stock exchange and trading are no exception.

Day after day, algorithmic trading is gaining more and more popularity. This technology relies on computer programs to execute algorithms that automate a trading strategy. We have been seeing more and more asset managers and hedge funds incorporate ML techniques to improve their trading strategies. And we have even seen computer-driven hedge funds that, according to The Financial Times, have already joined the industry's top performers [87].

Proof of this ML growth is that, nowadays, investment banks and hedge funds are employing quantitative analysts to design and implement complex models that allow financial firms to price and trade securities, help with risk management, or even develop new trading strategies. And its importance can easily be seen reflected in their salaries; it is not uncommon to find positions with posted salaries of $250,000 or more and with bonuses, $500,000+ is achievable, according to Investopedia [116].

During the development of this Master's Thesis, I will study the current state of these technologies and learn about, Stock, Trading, Machine Learning (ML), Algorithmic Trading, Reinforcement Learning (RL), Deep Learning (DL), etc. And apply this knowledge to design and develop an Algorithmic Trading System using different ML techniques. I will later compare the obtained results with the ones obtained in previous Algorithmic Trading projects and with the usual trading performance. I will end with a quick overview of possible improvements of the system or future lines of work.

Keywords: Algorithmic Trading, Deep Learning, Machine Learning, Neural Networks, Bitcoin, Cryptocurrency

Contents

Resumen **VII**

Abstract **IX**

Contents **XI**

List of Figures **XV**

List of Tables **XVII**

List of Acronyms **XVII**

1 Introduction **1**

1.1 Context 2

1.1.1 Algorithmic Trading 2

1.1.2 Blockchain and Cryptocurrencies 3

1.1.3 Why Cryptocurrencies? 4

1.1.4 Why Bitcoin? 4

1.2 Project goals 5

1.3 Task Methodology 6

1.4 Structure of this document 6

2 Case Study and Related Work **9**

2.1 Case Study 10

2.2 Related Work 11

2.2.1 Vincent Poon and ThirstyScholar 11

2.2.2 PacktPub 14

2.2.3 Standford University 14

2.2.4 Rafael Schultze-Kraft 15

2.2.5 Adam King 16

2.2.6 Abhinav Sagar 17

3 Enabling Technolgies **19**
3.1 Artificial Intelligence and Machine Learning 20
3.1.1 Supervised Learning 21
3.1.2 Unsupervised Learning 22
3.1.3 Reinforcement Learning 23
3.2 Machine Learning Methods and Technologies 24
3.2.1 Deep Learning 24
3.2.2 Artificial Neural Networks 25
3.2.2.1 Multilayer Perceptron (MLP) 26
3.2.2.2 Backpropagation 26
3.2.2.3 Activation Function 26
3.2.2.4 Rectified Linear Unit (RELU) 26
3.3 Libraries 27
3.3.1 PyTorch 27
3.3.2 Numpy 27
3.3.3 Pandas 28
3.3.4 TensorFlow 28
3.3.5 Scikit-Learn 29
3.3.6 Ta-Lib 29
3.3.7 Zipline 30
3.3.8 Alphalens 30
3.3.9 OpenAI 31
3.3.10 OpenBlender 31

4 Model Building and Architecture **33**
4.1 Pipeline 34
4.2 Data Set 35
4.2.1 Data Fields 35
4.3 Assumptions 38
4.4 Preprocessing 39
4.4.1 Indicators 40
4.4.2 Cleaning the Dataset 41
4.5 Model Building 42
4.5.1 Algorithms 42
4.5.2 Q-Learning 43
4.5.3 Value Update Methods 44
4.5.3.1 Value Iteration 44

4.5.3.2 Policy Iteration . . . 45
4.6 Optimizing the model . . . 46

5 Training and Obtained Results **49**
5.1 Training . . . 50
5.2 Results . . . 50
5.2.1 15-Minute Candles . . . 53
5.2.2 1-Minute Candles . . . 54
5.2.3 1-Hour Candles . . . 55
5.2.4 12-Hour Candles . . . 56
5.3 Selected Grouping Method's Outcome . . . 57

6 Include News **59**
6.1 Dataset and Goals . . . 60
6.2 Pipeline . . . 62
6.3 Get Bitcoin Data and Preprocess . . . 63
6.4 Include Fox News . . . 64
6.5 Preprocess Blended data and Train model . . . 65
6.6 Results and Conclusions . . . 67

7 Conclusions and Future Work **71**
7.1 Conclusion . . . 72
7.2 Achieved Goals . . . 72
7.3 Problems Faced . . . 73
7.4 Future Work . . . 74

A Impact of this project **77**
A.1 Social Impact . . . 78
A.2 Economic Impact . . . 79
A.3 Environmental Impact . . . 79
A.4 Ethical and Professional Implications . . . 79

B Budget for this Project **81**
B.1 Technological Resources . . . 82
B.2 Licenses . . . 82
B.3 Computational Resources . . . 82
B.4 Human Resources . . . 82
B.5 Taxes . . . 83

List of Figures

2.1 Bitcoin's Price Graph Chart [9] . . . 10
2.2 Losses and Rewards Vincent Poon [57] . . . 12
2.3 Results ThirstyScholar [85] . . . 13
2.4 Results ThirstyScholar With Current Dataset . . . 13
2.5 Rafael Schultze-Kraft's results [74] . . . 15
2.6 Rafael Schultze-Kraft's results Zoomed In [74] . . . 15
2.7 Adam King's Rewards [37] . . . 16
2.8 Abhinav Sagar Predictions [69] . . . 17

3.1 Machine Learning Workflow [30] . . . 21
3.2 Supervised Machine Learning [26] . . . 21
3.3 Unsupervised Machine Learning [26] . . . 22
3.4 Reinforcement Learning [112] . . . 23
3.5 Artificial Intelligence (AI) in movies . . . 23
3.6 Deep Learning Workflow [30] . . . 24
3.7 Deep Learning Neural Network . . . 25
3.8 PyTorch . . . 27
3.9 NumPy . . . 27
3.10 Pandas . . . 28
3.11 TensorFlow . . . 28
3.12 Scikit-Learn . . . 29
3.13 Zipline . . . 30
3.14 Alphalens . . . 30
3.15 OpenAI . . . 31
3.16 OpenBlender . . . 31

4.1 Pipeline . . . 34
4.2 Timestamp in Unix time . . . 35
4.3 Bitcoin's Opening Value . . . 36
4.4 Bitcoin's Closing Value . . . 36
4.5 Bitcoin's Highest Value . . . 36

4.6 Bitcoin's Lowest Value 36
4.7 Sum of All Bitcoin Values 37
4.8 Sum of All US Dollar Values 37
4.9 Weighted Price 37
4.10 Q-Learning Update Equation 43
4.11 Value Iteration Algorithm [79] 44
4.12 Policy Iteration [79] 45
4.13 Policy Iteration Algorithm [79] 45
4.14 Adam Vs SGD 46

5.1 15-Minute Candles 53
5.2 1-Minute Candles 54
5.3 1-Hour Candles 55
5.4 1-Minute Candles 56

6.1 OpenBlender's Bitcoin Dataset Head Rows 60
6.2 OpenBlender's Fox News Dataset Head Rows 61
6.3 Word Count OpenBlender's Fox News Dataset Head Rows 61
6.4 Include News Pipeline 62
6.5 News Add Change Over 0 Attribute 63
6.6 News Change Over 0 Shifted 63
6.7 News Text Vectorizer 64
6.8 News and Bitcoin Blended Data Parameters 64
6.9 News Final Dataset 65
6.10 News Train and Test Sets 65
6.11 News Results Random Forest 66
6.12 News Results Decision Tree 66

List of Tables

4.1 Vincent Poon Indicators 40

5.1 Results 15-Min Candles 53
5.2 Results 1-Hour Candles 55
5.3 Results 12-Hour Candles 56

6.1 Confusion Matrix 67
6.2 Random Forest Confusion Matrix 68
6.3 Decision Tree Confusion Matrix 68

List of Acronyms

AD Automatic Differentiation

AI Artificial Intelligence

ANN Artificial Neural Network

API Application Program Interface

AUC Area Under The Curve

CPU Central Processing Unit

CUDA Compute Unified Device Architecture

DL Deep Learning

DNN Deep Neural Network

EM Expectation–Maximization

FP False Positive

FN False Negative

GPGPU General-Purpose Computing on Graphics Processing Units

GPU Graphics Processing Unit

IT Information Technology

KDD Knowledge Discovery in Databases

LSTM Long Short-Term Memory

ML Machine Learning

MLP Multilayer Perceptron

MDD Maximum Drawdown

NN Neural Network

NLP Natural Languaje Processing

OSS Open Source Software

PIT Point in Time

PR Precision-Recall

RELU Rectified Linear Unit

RL Reinforcement Learning

ROC Receiver Operating Characteristics

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

SVM Support Vector Machines

TP True Positive

TN True Negative

CHAPTER 1

Introduction

This chapter will introduce the context of this project, including a brief explanation of some cryptocurrency use cases, as well as a brief overview of all the different parts that will be discussed in this master's thesis.

It will also break down a series of objectives and goals to be reached during the realization of this project. This chapter will have a third section, explaining the main tasks that need to be accomplished for a correct development of this thesis.

Moreover, it will introduce the structure of the document with an overview of each chapter.

1.1 Context

1.1.1 Algorithmic Trading

Machine Learning (ML) is growing rapidly, and it's being used in many different fields. Forbes found that advanced initiatives related to data science and machine learning, including data mining, advanced algorithms, and predictive analytics are ranked the 8th priority among the 37 technologies and initiatives surveyed in a study they performed [23].

One of the fields ML is growing into is Trading. This is called Algorithmic Trading. Algorithmic trading has been gaining popularity and, not only have we seen big and small hedge funds insert ML techniques to their trading strategies, but we have seen new computer and quant-based ones rise. Even the world's largest asset manager, the American global investment management corporation BlackRock, "bets on algorithms to beat the fund managers", according to The Financial Times [86].

Algorithmic trading is part of the every-day news, making it clear that everyone is acquiring some sort of Algorithm to do compute trading in a faster and more efficient way. Looking no further, The Financial Times has multiple articles on the subject, such as: "When Silicon Valley came to Wall Street" [90], "Two Sigma rapidly rises to top of quant hedge fund world" [89], or "Fintech: Search for a super-algo" [88].

Therefore it seems like computers are quickly gaining over the jobs of trader's, or as Bloomberg's reporter Saijel Kishan puts it: "Robots Are Eating Money Managers' Lunch" [40]. And the same Bloomberg later dedicated a full article to getting the insights of the work of quants in Hedge Funds: "Rise of Robots: Inside the World's Fastest Growing Hedge Funds" [7], where he states that the fastest growing hedge funds are the ones using Algorithmic Trading.

In this regards, Christina Qi, founding partner of the hedge fund Domeyard said in an interview "These days we don't hire MBA's. If you want to get into trading study math, not finance." [29]. And Even The Wall Street Journal said: "For decades, investors imagined a time when data-driven traders would dominate financial markets. That day has arrived.", in their article called "The Quants Run Wall Street Now" [32].

Since this new technology is so important, this Master's Thesis created a system that implements RL applied to Algorithmic Trading. As we will see in the following Subsection 1.1.2, Section 1.2 and Chapter 2, most of the projects that tried this so far have a more hands-on approach, where a human's supervision is usually required. This project has a more hands-off approach, using Reinforcement Learning (RL) and Artificial Neural Networks (ANNs).

1.1.2 Blockchain and Cryptocurrencies

Blockchain is increasing popularity nowadays, and, along with it, so are cryptocurrencies.

Blockchain is a technology made of a growing list of records called blocks. It is resistant to modification of the data since it is "an open, distributed ledger that can record transactions in a verifiable and permanent way" [12]. In this system, everyone knows everything, since it's decentralized, and therefore, for someone to lie, he would have to get at least half of the network to lie with him, which is very hard, if not impossible.

Many different companies are now exploring and starting projects with blockchain. So much so that Michael del Castillo states, in his Forbes article, that the 10 largest companies in the world are now exploring blockchain [18]. And Kate Rooney, reporter for the CNBC says that 84% of companies are dabbling in blockchain [67].

One example of this in Spain is Santander bank. In April, 2018, it released its first international, blockchain-based, transfer service [71]. And again in July 2018 it released its first operations with we.trade [93], a blockchain platform whose banking partners are 13 big European banks: Caixa Bank, Deutsche Bank, Euro Bank, Erste group, HSBC, KBC, Natixis, Nordea, Rabobank, Santander, Societe Generale, UBS and UniCredit [72].

Cryptocurrencies are also gaining strength and becoming much more accepted as a regular payment method by many companies. Many different things can be payed for with cryptocurrency; from traveling the world (since travel agencies such as CheapAir [8] or Destinia [20] are accepting Bitcoins to pay for flights, car rentals, hotel bookings, etc.), to buying a ticket to space [5] or purchasing sports cars, such as a Lamborghini [43]. Even The Bank of England has not closed the door on a central bank-issued cryptocurrency [11].

So, as we have seen, cryptocurrencies are very strong nowadays, and getting stronger. Another proof of this is that Nasdaq [46] is now working with 7 cryptocurrency exchanges, as we see in Michael del Castillo's Forbes article [19].

Bitcoin does not have an intrinsic value, like gold. Nevertheless, value continues to exist and grow, since investors think they are worth billions. Of course, cryptocurrencies are good for some things and not so good for others, as we can see in Mike Orkut's article for the MIT Technology Review [51]. But one thing is for sure: Bitcoins, as well as many other cryptocurrencies, are here to stay.

So, in this project we created a system that, using Deep Learning (DL) techniques such as Neural Networks, is able to make trading decisions based on algorithms learned by RL.

As stated before, this task is usually done, or at least supervised, by humans, making it unable to adapt to change too well and being limited to what a human can do, realize or think. On the other hand, our system is much more "hands off" and is able to adapt to change, as well as to "think" of new strategies that a human might have missed.

1.1.3 Why Cryptocurrencies?

The impulse around cryptocurrencies is something revolutionary: to take the production and control of money, away from governments.

Another aspect about Bitcoin is that it is limited in quantity, thus functioning more like precious metals [77].

As we said before, cryptocurrencies could have a huge usage in e-commerce. As Randall Stephens states in his post for The Startup: "Although e-commerce is growing, the obsolete global financial system represents the biggest barrier to its expansion. Currently, banks act as intermediaries between buyers and sellers on the internet. That's not a big problem for people who already shop online. However, there are many people who could benefit from online shopping but can't open a bank account.

I'm talking about people from third-world countries where banking systems are undeveloped, as well as disadvantaged people from developed countries. According to some estimates, there are more than two billion people worldwide who fall under this category. [28]

"Cryptocurrency could connect those people with the world of e-commerce. After all, they just need an internet connection to get started. There's no list of requirements for downloading a wallet and using digital coins as a means of payment. You don't even need to provide your personal information." [80]

1.1.4 Why Bitcoin?

Bakkt, a company owned by the New York Stock Exchange, recently announced on their Tweeter [91] account: "Bitcoin today accounts for over half of total crypto market capitalization and has been deemed to be a commodity, and its derivatives are regulated in the US by the CFTC... As the world's most liquid and widely distributed cryptocurrency, and where we've seen the most customer demand, bitcoin's profile creates a liquid product on which to build a futures contract" [3].

Anthony Pompliano, the host of the popular crypto podcast and newsletter "Off The Chain," thinks that Bitcoin is still the best performing asset class in the last 5 years: "Bitcoin is down over 75% from its all-time high. It is still the best performing asset class in the last 5 years. It has dwarfed stocks, bonds, currencies and commodities since inception too. Ignore the noise, trust the code." [56].

Even Elon Musk, founder of many companies such as PayPal, Tesla Motors, and SpaceX, has recently said that bitcoin is "quite brilliant." He stated this during an interview, and added that crypto is a better way of transferring value than paper money [70].

Also, as we stated before, the number of things you can buy with Bitcoin is growing. According to CoinMap, 14,506 venues around the world currently accept it [10].

1.2 Project goals

As we explained before, even though Algorithmic Trading has been successfully implemented in the past, and has been around for a while now in some scenarios or companies, its performance is very human-supervised, thus limiting said performance from achieving its full potential. This project's main goal is to create a system capable of making trading decisions without any human supervision whatsoever. For that, this project will exploit and use Machine Learning (ML) techniques, specifically some Reinforcement Learning (RL) techniques combined with Neural Networks (NNs).

To accomplish this, our first goal could be to, after spending some time researching a little bit more about The Stock Exchange, trading, cryptocurrencies in general and Bitcoin in particular, become wiser in all these topics. We will need to gather all information possible regarding these topics in order to perform a good work as well as to be able to evaluate the agent's outcome.

Then, we will have to proceed to find a good database or dataset for our agent to work with. Our agent's decisions will be based on strategies and rules it comes up with based on such dataset, therefore we could conclude that a very important goal would be to select a correct dataset for our agent. Later, we will have to extract said data, and properly clean and pre-process it accordingly to our project goals.

And then comes our more obvious goal, the main goal mentioned above which would be to actually create the trading system that is able to successfully trade bitcoins based on its own strategies and decisions, without any human supervision whatsoever. In this part, it will also be very important to decide the correct input for our agent aside from the extracted data. So basically all calculated parameters as well as the optimizing parameters for our model.

After creating such a system, we will test it and we will evaluate its viability and performance. Afterwards, we will add new functionalities to our system trying to obtain some added value, such as a comparison of its performance using different trading methodologies, or study the viability of adding some extra data, such as external business news from newspapers.

Last, but not least, since ML, DL, RL, NN, etc. are so important nowadays, the last goal of this project is that its readers acquire some knowledge as to how these technologies work, their potential use, and how to develop such systems.

1.3 Task Methodology

This document can be branched into six phases that we needed to go through for a successful development of this project.

- ***Phase 1:*** Review and Study of Blockchain, Cryptocurrencies, Trading and Investing.
- ***Phase 2:*** Study and learn about RL and NN techniques to be able to develop this project correctly.
- ***Phase 3:*** Correct selection, of the cryptocurrency's data needed for our agent.
- ***Phase 4:*** Cleanse, organization and pre-processing
- ***Phase 5:*** Software development and implementation of an algorithmic trading system following the next steps:
- ***Phase 6:*** Feeding Data to our agent.
- ***Phase 7:*** Results analysis to check the system's response.
- ***Phase 8:*** Compare our system's results to other known trading strategies to evaluate its performance.
- ***Phase 9:*** Additional phase, to check the viability of including new data into our model. Tested with data from Business News from Fox.

All of these stages where accomplished successfully as we will see in this document.

1.4 Structure of this document

In this section we give a brief summary of this project's chapters. The remainder of this document is structured as follows:

Chapter 2 will have a brief case study, or description of the data used in this project. It will also include the state of the art, where we will analyze previous related projects done so far.

Chapter 3 will contain an overview and explanation of all the different technologies that made the realization of this project possible.

Chapter 4 will present the global architecture of the project as well as more detailed explanation of the chosen dataset and techniques used. It will provide a general description of our classifier and Neural Networks (NNs).

Chapter 5 will have a more detailed explanation of the training of our model as well as the results obtained in this project. We will compare our results with other projects and trading techniques.

Chapter 6 will show the research done on investigating if it would be possible and/or useful to include business news to our agent to improve its outcome.

Chapter 7 will discuss the different conclusions drawn from this project, achieved goals, problems faced and suggestions for future projects.

CHAPTER 2

Case Study and Related Work

In this chapter we will start by giving a quick overview of the chosen Data set. We will explain how Bitcoin's historic data is collected, how it is organized, as well as its characteristics.

Secondly, we will analyze related projects and papers that have been published in the past, regarding systems used for algorithmic trading using ML techniques, such as RL, DL and NNs. We will discuss the techniques they used, and the results they obtained with such techniques.

To summarize, we will briefly explain the conclusions extracted from all this research and past work that, as a result, led to the approach taken for the final development of this project.

2.1 Case Study

For a correct experimentation of this master thesis, and for our agent to properly learn how to trade, we have to feed it significant data. As we explained in Subsection 1.1.2, Blockchain, and therefore Bitcoin, are technologies where all users posses all information. Thus, all of Bitcoin's information and transactions are public knowledge. In Figure 2.1 we can see the price fluctuation in Bitcoin until today.

Although the data is public, we decided to use Kaggle's dataset, since it includes Bitcoin's historical data at 1-minute intervals from selected exchanges from January 2012 to the current date [34]. Kaggle's dataset not only contains Bitcoin's price at a certain time of the day, it actually provides more useful information.

We downloaded the data from 2012-01-01 to 2018-06-27. We will describe in more depth the data downloaded from Kaggle in Chapter 4, but it basically consist of Bitcoin's market data, stored in a table with the following eight fields:

1. Timestamp
2. Open
3. High
4. Low
5. Close
6. Weighted Price
7. Volume (BTC)
8. Volume (Currency)

We will later pre-process that raw data and arrange it in a way that works better for our agent as we will see in Chapter 4.

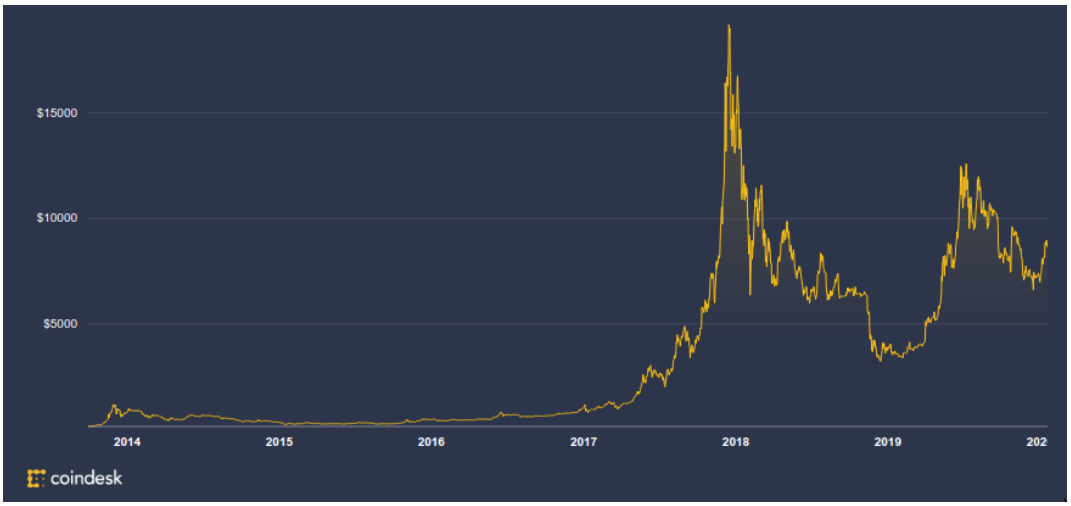


Figure 2.1: Bitcoin's Price Graph Chart [9]

2.2 Related Work

Although there has not been any groundbreaking projects, some research and small projects have been done.

It is key to mention that most of these projects were done before December, 2017, when Bitcoin reached its highest value. Until that moment, Bitcoin's price had a fairly straight rising trend, and therefore was easier to predict. Since the big price drop in December, 2017, Bitcoin's price has been more random as we can see in Figure 2.1, thus making it harder to predict it's future price.

This is a factor that should be taken into account when comparing results in Chapter 5. Since the results obtained by these other projects would be worst, had they used current data, and likewise, this project's results would be better if we just worked with data prior to 2017.

2.2.1 Vincent Poon and ThirstyScholar

Vincent Poon described how he applied RL algorithms to trade Bitcoins in his paper for Launchpad.AI [57]. While he gives implementation details, he doesn't provide any code. Because of this, ThirstyScholar tried to reproduce the results himself, posting the code he created on his GitHub profile [85].

This project is mostly based on their work, since it was the idea and the procedure we liked the most, and the one which proved to give better results when applying RL to an AI (ML) trading system. This is why we give more details about how they built their system.

As Vincent Poon explains: "RL is appropriate when the state space is large or even continuous. It may be especially useful when it is impractical to obtain labels for supervised learning. Trading is a good example of this where the correct actions aren't known and even if they were, would be nearly impossible to apply to every situation in which the agent has to act. RL is also appropriate when, as in trading, the actions have long term consequences and rewards may be delayed" [57].

As we will see later on in Chapter 3, RL works by having the agent decide the action from which it could get the maximum reward, given its current state. Poon decided to go for a policy based approach, he parameterized the policy, and then he finds the parameters that maximize the expected reward.

ThirstyScholar, based on what Vincent Poon described, grouped the one minute Bitcoin data into groups, or candles, of 15 minutes. Since every candle represented 15 minutes, it took a group of 96 candles to represent a whole day of Bitcoin trading (24h). These groups were defined as *episodes*.

Vincent Poon decided to take any random block of 96 15-min candles, to create the first episode. The agent would be given a random number of Bitcoins between 0 and 4 to start the episode. The agent had to choose between selling, holding, or buying Bitcoins, with a maximum of 4 Bitcoins.

In order for the RL agent to learn a proper policy, they inputted 5 variables that represented the 5 possible Bitcoin holdings (from 0 to 4) along with 18 market indicators, making use of the open-source software library Ta-Lib [83], which we will described in Section 3.3.6.

At the end of the episode, they stored the inputs, the actions that were taken, and the calculated return for those actions. The RL Agent's mission was to learn from its actions and modify the NN by adjusting the weights of the neurons so its future actions granted it the highest reward, as we will see in Chapter 3.

Poon chose a policy gradient agent. As Thomas Simonini explained it: "In policy-based methods, instead of learning a value function that tells us what is the expected sum of rewards given a state and an action, we learn directly the policy function that maps state to action (select actions without using a value function)" [78].

For the structure of their Multilayer Perceptron (MLP) they had three layers: one Input layer, one Output layer and one Hidden layer. We will see in more detail the technical parts of NN and MLP in Subsection 3.2.2, but we wanted to comment that he used 23 neurons in his Hidden layer and decided to use a RELU activation for his model.

After training their model agent, they achieved the losses and rewards that, after being smoothed, are shown in Figure 2.2.

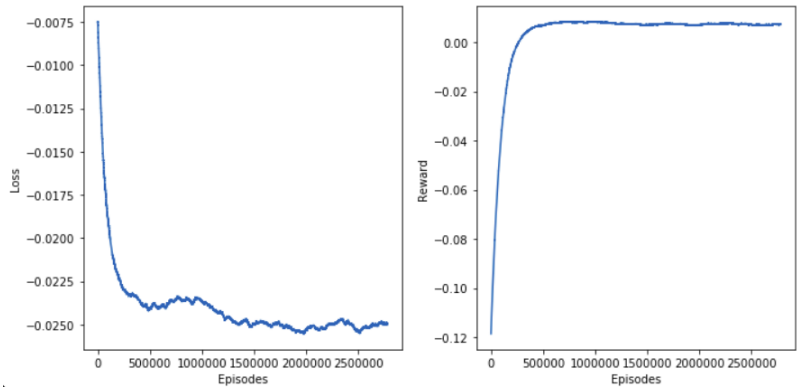


Figure 2.2: Losses and Rewards Vincent Poon [57]

As it can be seen, the results were pretty satisfactory. The losses rapidly decrease, thus having the reward increase exponentially. With this, they concluded that RL could also be successfully applied to algorithmic trading.

The results ThirstyScholar uploaded are the ones shown in Figure 2.3. Looking at them, we can see that the results seem really good.

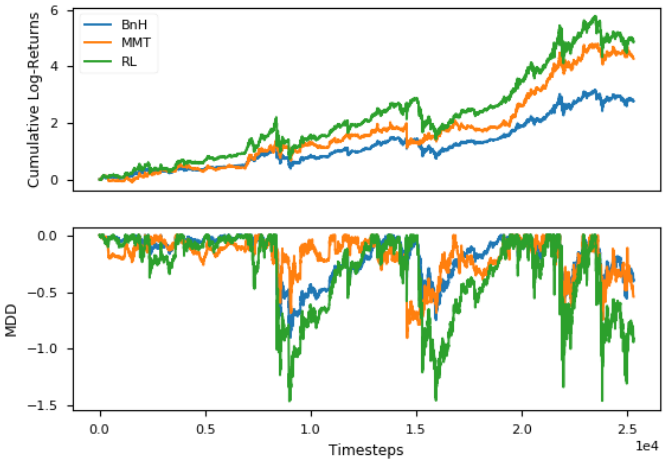


Figure 2.3: Results ThirstyScholar [85]

However, ThirstyScholar himself said: "I observed substantial variability in the test result", which means that these might be the best results he achieved, and it is also important to note again that these results are achieved because the data they used only goes to 2017; which means that they didn't have to deal with our famous tipping point in 2018.

Figure 2.4 shows the results obtained by running their code with the data used in this project, which includes data up to 2019. We can see results are pretty bad compared to other trading methods and to the results they posted in their paper.

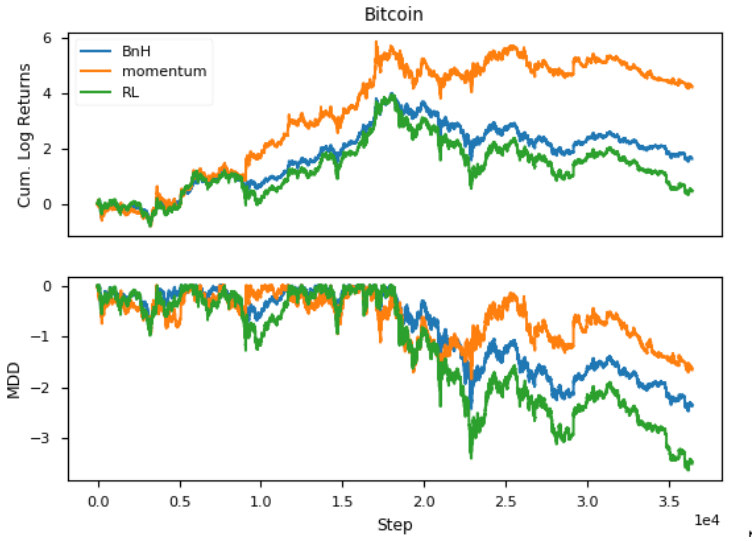


Figure 2.4: Results ThirstyScholar With Current Dataset

2.2.2 PacktPub

The print-on-demand and famous e-book company Packt (Packt Pub) [52] chose Scala [73] to also try predicting Bitcoin prices from historical data [76].

They used the same dataset from Kaggle [34] and they also used real time data obtained from Cryptocompare API [15].

To evaluate the results of their system, they used the area under the Receiver Operating Characteristics (ROC) curve and these are the results they obtained:

- Area under ROC curve: 0.6045355104779828
- Area under the Precision-Recall (PR) curve: 0.3823834607704922

Therefore, they did not achieve very high accuracy.Since it was a short article, not a big paper, they didn't try to improve said results, but as they stated, they could've gotten better accuracy, if they had tuned the hyperparameters.

2.2.3 Standford University

Isaac Madan, Shaurya Saluja and Aojia Zhao from the Department of Computer Science of Stanford University, CA, did a project also attempting to use ML algorithms to predict Bitcoin prices [44]. They divided the project into two phases.

During the first phase they used a data set that consisted of 25 features related to the price of Bitcoin over the course of five years, recorded daily. They focused on predicting whether the price would go up or down the next day. For this, they accomplished an accuracy of 98.7% while predicting the sign of the daily price change.

For the second phase they used data recorded in 10-minute and 10-second intervals. These results had 50-55% accuracy in predicting the sign of future price change using 10-minute time intervals. They didn't provide any code for their project, but, after reading some reviews in different forums, their results may not be very realistic. First of all, 98.7% seems too good to be true to almost everyone. Secondly, they might have had a case of overfitting in their Random Forest algorithm. Thirdly some say that the 50% accuracy result in the 10-minute interval case is not really any better than a coin toss, and, therefore, should not be used for trading at all. Another concerned developer also stated, "Another concern here is the lack of confusion matrices and model-free baselines; what performance do you get by guessing "positive" or "negative" every time, or by guessing the same sign as the last interval?" [24].

The approach itself also seems quite shallow, making the results even more suspicious, but again, without the code to look at, we can't really be sure whether they made one or more "cardinal sins" or not.

2.2.4 Rafael Schultze-Kraft

Rafael Schultze-Kraft did a similar project, regarding using DL to predict cryptocurrency and actually using it for real trading [75], and wrote an article explaining his work [74].

Schultze-Kraft explains he used Python and Keras [35] to create a multidimensional Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) with 20 neurons, to predict the price of Bitcoin, that yielded the results in figure 2.5.

Figure 2.5: Rafael Schultze-Kraft's results [74]

As he himself explains, these results look very good: almost too good to be true. This is because they are not true. There is something utterly deceptive about these results.

Taking a closer look, we can see in Figure 2.6 that all his model learned was to use the value of a given day to predict the value of the following day. As Schultze-Kraft said: "Yes, the network is effectively able to learn. But it ends up using a strategy in which predicting a value close to the previous one turns out to be successful in terms of minimizing the mean absolute error."

Figure 2.6: Rafael Schultze-Kraft's results Zoomed In [74]

2.2.5 Adam King

Adam King, in his article "Creating Bitcoin trading bots don't lose money" [38] creates deep RL agents that learn to make money trading Bitcoin. He decided to use OpenAI's Gym and the PPO agent from the stable-baselines library, a fork of OpenAI's baselines library.

He followed three simple steps: Create a Gym environment. Render a visualization of such environment. And train the agent to learn a profitable trading strategy.

Like this Master's Thesis, he did take into account that It's important to only scale the data the agent has observed so far to prevent look-ahead biases, which is an important factor to take into account. Therefore he did not use cross validation (K-Fold validation, etc.).

After tuning all parameters and fixing bugs, he accomplished the rewards that can be seen in Figure 2.7.

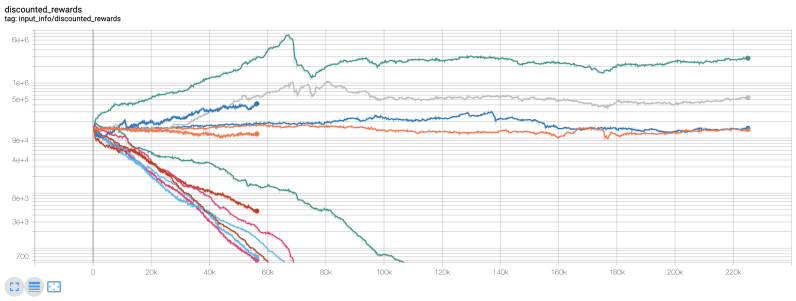


Figure 2.7: Adam King's Rewards [37]

As it can be observed, a couple of agents did well, but the rest traded themselves into bankruptcy. However, he continued his work in a following article where he used Bayesian optimization to zone in on the best hyper-parameters and improve the agent's model. He switched from a Multilayer Perceptron (MLP) network to a Long Short-Term Memory (LSTM) network.

Although he improved his results, he wasn't able to improve them by much. Most of his agents kept loosing money. This time he compared it with another three common trading strategies: Buy and Hold, RSI Divergence and SMA Crossover. But the rewards his agent's got were always lower than the other three common techniques.

2.2.6 Abhinav Sagar

Although this next project is not trying to accomplish the same thing as ours, Abhinav Sagar, in an article called: "Cryptocurrency Price Prediction Using Deep Learning" [69] talks about how, by using a LSTM NN, managed to predict bitcoin prices using real-time data.

The Bitcoin data used was gathered from CryptoCompare [15]. He provides also the code of his project for free in a Git Hub repository [68]. Sagar went though four stages in his project: Getting the data, preparing it for training, training and testing, and predicting the prices.

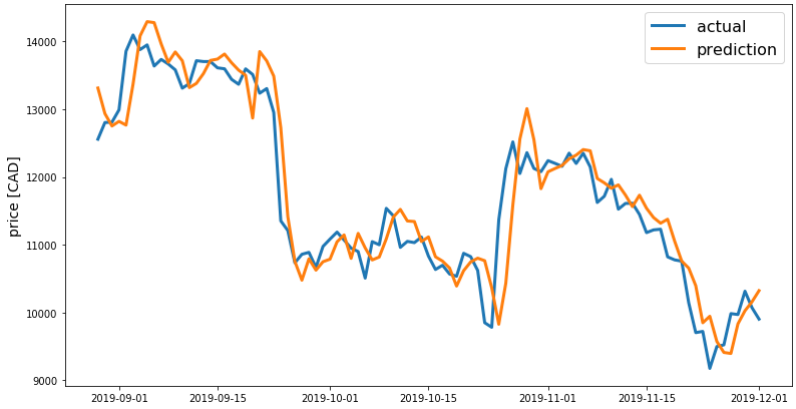


Figure 2.8: Abhinav Sagar Predictions [69]

We can see the obtained results in Figure 2.8, but, as it can be observed, they are not very accurate. Although this is supposed to be a short paper and by playing with the hyper-parameters the results could be improved.

Therefore we can conclude that, this last project is still not a good enough approach to our problem. Also, as we said in the begging, this thesis will try to create a RL agent that si able to trade bitcoins, not just predict what the bitcoin's price is going to be.

CHAPTER 3

Enabling Technolgies

This chapter offers a brief review of the main technologies that have made possible the development of this project.

This project is based on AI. More specifically, based on DL and Neural Networks (NNs), or ANNs, to differentiate them from Biological Neural Networks.

This project will use Python as the programming language. With Python, we will use PyTorch, an open source deep learning library for Python based on Torch. PyTorch provides two high-level features: Tensor computation (we will use NumPy for that) and Deep Neural Networks (DNNs). We will also use TensorFlow, another open source library for data flow used for NN in ML.

Another library we will use is Pandas. It provides easy-to-use data structures and data analysis tools. And last, but not least, we will also use the Ta-Lib library. Ta-Lib is a library of Technical Analysis indicators. "In finance, technical analysis is an analysis methodology for forecasting the direction of prices through the study of past market data, primarily price and volume" [113].

3.1 Artificial Intelligence and Machine Learning

In computer science, Artificial Intelligence (AI) is intelligence demonstrated by machines. The term is usually associated to machines that mimic "cognitive" functions that humans associate to the human mind, such as "learning" and "problem solving" [102].

A thing to keep in mind is that, as machines become more and more capable, tasks considered to require "intelligence" are often removed from the definition of AI - a phenomenon known as the AI effect [101].

This project will address an AI that's focused on a machine having the ability and power to learn over time - ML.

Machine Learning (ML) is an application of AI that, without being explicitly programmed (having explicit instructions), provides systems the ability to automatically learn.

It usually begins with observations, or data, as a sample input. This data is then processed by a human to extract features for the system to use to improve, or create, different algorithms that will later be used to make decisions, classify, predict future outcomes, etc. In Figure 3.1 we can see a diagram of a ML process to classify cars.

In other words, ML aims to automatically put the human process of reasoning, and coming to reasoned conclusions, into code.

There are various requirements which must be met to attempt to use Machine Learning for making a decision or solving a problem. First, there must be a pattern in the input data for the algorithm to understand and form an opinion about. It could even be a pattern almost impossible to be detected by a human mind, but the system might be able to realize that there is such a pattern. Second, there must be a large amount of input data, the larger the data set, the more accurate the prediction is going to be. Third, because a human cannot create a mathematical formula which describes and classifies a problem, ML is used for two things: to "understand" the data, and to "learn" in a structured way, getting to a mathematical approximation that describes the way a problem behaves. So we should use ML if that is the kind of problem we are trying to solve. [53]

In this project we will have our system try to figure out a pattern in cryptocurrency values. We will have all the historical data of such values, and therefore, a large dataset. And lastly, we want the system to "learn" by itself, and not just execute pre-coded instructions.

The middle component of Figure 3.1, the ML classification algorithm, is the key part. This ML process can be one of three types: Supervised Learning, Unsupervised Learning or Reinforcement Learning.

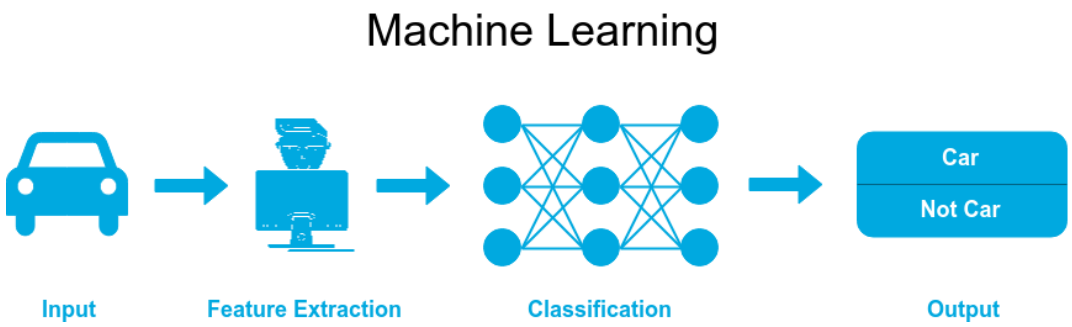


Figure 3.1: Machine Learning Workflow [30]

3.1.1 Supervised Learning

Supervised ML algorithms can map an input to an output applying what has been learned in the past to new data, based on labeled data; this is, example input-output pairs.

Supervised learning is used for cases where a label or classification is available for the data set that is going to be used to "train" the algorithm (Figure 3.2) [65, 26].

After looking at previous data and its corresponding label, various types of algorithms can be applied to obtain the expected label. Some of the most popular ones are: Decision Tree, Ordinary Least Square Regression, Logistic Regression, Support Vector Machines (SVM) and Naïve Bayes Classification.

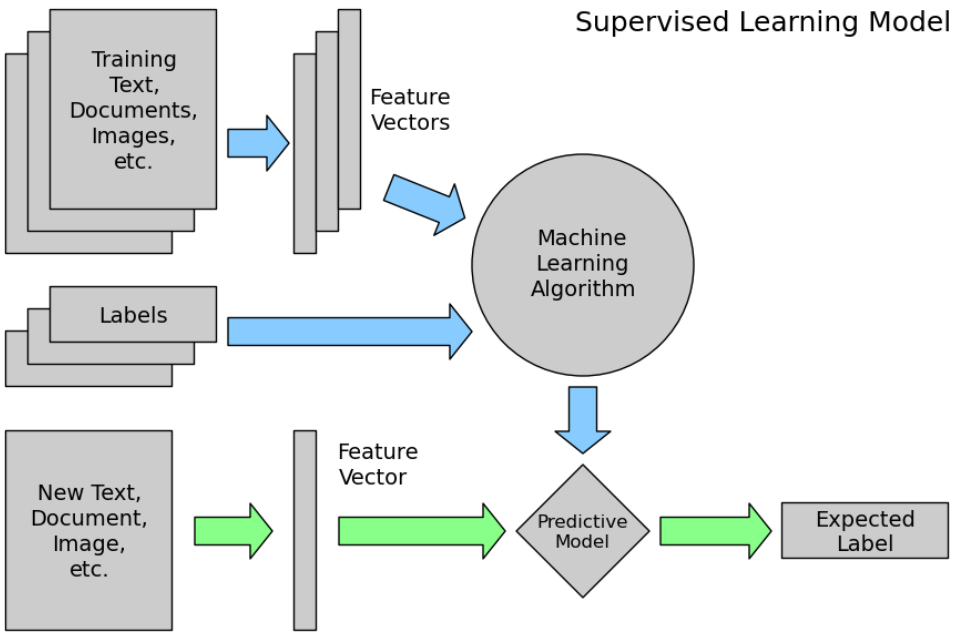


Figure 3.2: Supervised Machine Learning [26]

3.1.2 Unsupervised Learning

By contrast, Unsupervised Learning is useful in cases where input data is neither labeled nor classified. Therefore, the system must deduce or imply a relationship between the data and come up with different algorithms, as explained in figure 3.3. It must learn relationships between elements in a data set and classify the raw data without "help". [114]

As explained before, this would be useful if the pattern that follows the data is very hard, or even impossible, to be detected by a human being. Unsupervised Learning studies how systems can infer a function to describe a hidden structure from unlabeled data [82]

It can also be used to discover the representations needed to detect features or classifications from input data. All of this means that Unsupervised Learning is a means to an end [65].

Some of the most common algorithms used in Unsupervised Learning include: Clustering (Hierarchical, k-means, mixture models, etc.), Anomaly Detection (Local Outlier Factor), Neural Networks (NNs), Expectation–Maximization (EM) algorithm, Method of Moments, and bind signal separation techniques.

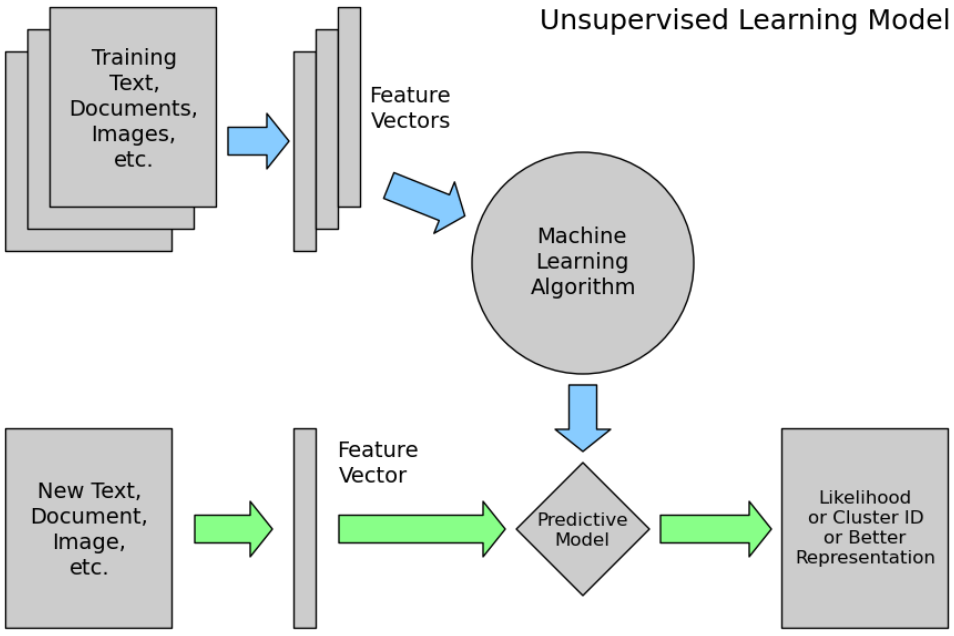


Figure 3.3: Unsupervised Machine Learning [26]

3.1.3 Reinforcement Learning

Reinforcement Learning (RL) is another area of ML. It allows machines and software agents to automatically decide the best behavior in a specific context, to maximize its performance.

Some kind of reward feedback is required for the agent to learn from its behavior. This way, our agent interacts with the environment by taking an action. After that, an interpreter, either a human being or by a programmed rule that checks if the decision has been good or bad, decides whether to give a positive or negative reward, as can be seen in Figure 3.4.

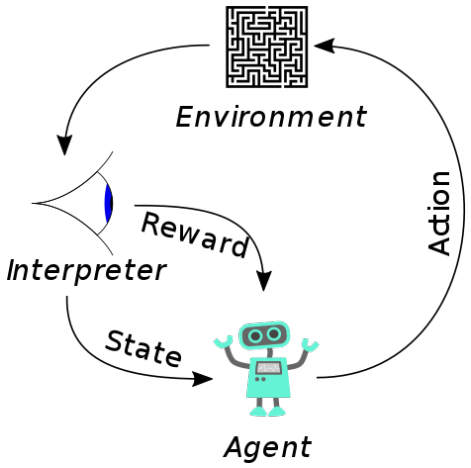


Figure 3.4: Reinforcement Learning [112]

If the agent makes the right decisions and accomplishes the desired outcome, it will get a reward. If not, it will receive a penalty. Its goal is to achieve the maximum reward possible.

RL's concern is to determine how the system should react or which action it should take in such an environment, in order to maximize the cumulative reward. As it happens with Unsupervised Learning, the input/output pairs need not be labeled. Instead, the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). [33]

Therefore, RL algorithms attempt to find a policy that maps states of the environment to the actions the agent ought to take in those states. Reinforcement Learning is used where the data is not labeled nor grouped by a property, but there is some type of feedback available for the predictive steps or actions. This is the AI we usually see in movies such as "Jarvis" from Iron Man (Figure 3.5a) or Wall-E from Disney's movie (Figure 3.5b). [53]

(a) Jarvis [45]

(b) Wall-E [92]

Figure 3.5: AI in movies

3.2 Machine Learning Methods and Technologies

3.2.1 Deep Learning

Deep Learning (DL) is a ML method that is based on using a cascade of layers, for feature extraction and classification. These layers are the ones used in Artificial Neural Networks (ANNs), which is why DL methods are usually called Deep Neural Networks (DNNs). Just like ML, its learning can be supervised or unsupervised.

As MathWorks describes it: "Deep Learning is a Machine Learning technique that teaches computers to do what comes naturally to humans: learn by example" [115].

The term "Deep" refers to the number of these so called "layers". We will get into more detail in Subsection 3.2.2 but, usually DL models can range from having from 2 to up to 150 hidden layers, and every one of these layers learns to transform its input data into a slightly different data.

The main differences with classical Machine Learning are:

- With a DL workflow, ***relevant features are automatically extracted***. There is no longer the need for a human to extract features manually like we saw in Figure 3.1. Instead, the system extracts the features for us, as we can see in Figure 3.6
- DL performs ***"end-to-end learning"***, where a network is given raw data and a task to perform - such as classification, or trade Bitcoins - and it learns how to do this automatically. This provides it with a much more "hands-off" approach.
- The key difference and advantage is that ***DL scales with data***, whereas other methods converge. DL algorithms usually ***improve as the size of the data increases.***

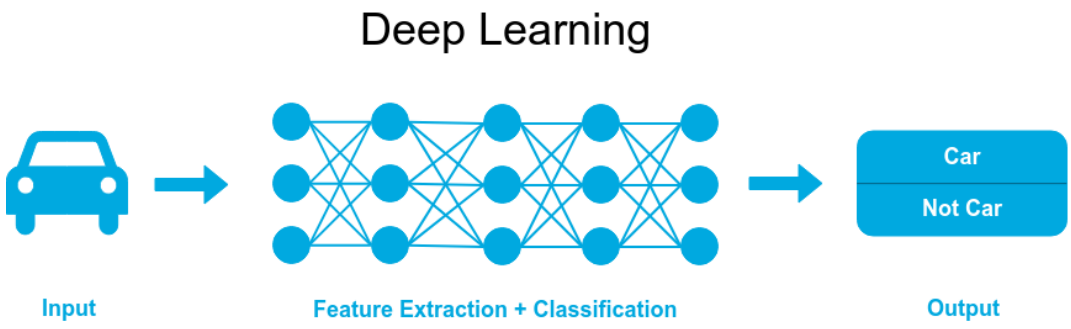


Figure 3.6: Deep Learning Workflow [30]

In Subsection 3.2.2 we will explain in more detail the middle part of Figure 3.6 which will be our ANN: What it does, and how its layers work.

3.2.2 Artificial Neural Networks

Neural Networks (NNs) were originally inspired by information processing and communications in biological systems, which is why, to discern between the two, they are usually referred to as Artificial Neural Networks (ANNs).

The ANNs act as a framework, where many different ML algorithms and functions work together to process complex data inputs. It can easily be explained using Figure 3.7.

Taking a look at the middle fragment of Figure 3.6, the first light colored column we see would be the *Input Layer*. This would be where our cleaned input data would be inserted into the ANN.

After that, we can see three *Hidden Layers*. These are the layers between the input and the output. Each one of these layers is made up of *Neurons*, every one of which is a function that transforms its inputted data, and inserts its output into the next neuron.

After the data has passed through all neurons, the new processed information is added up in the *Output Layer*.

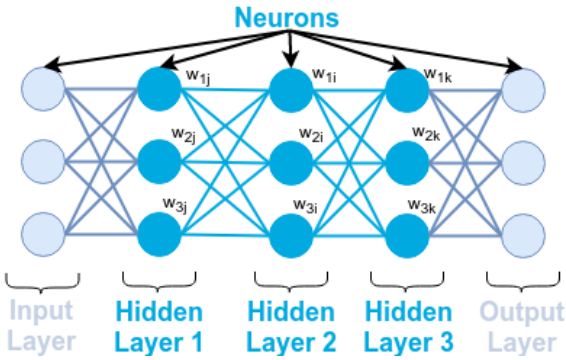


Figure 3.7: Deep Learning Neural Network

The key part about the middle Hidden Layers is that the functions can be "weighted" in order to give more relevance to some functions than others, that way varying its impact in the final outcome. These "weights" are represented by every "w" next to each neuron. Essentially, what the ANN should do, is find the optimal weights for each neuron, in order to obtain the best results, which, if it is using RL like in our case, means the biggest reward.

Pattern Recognition

Pattern recognition is a very important component of ANNs. It is usually used in speech recognition, text classification and computer vision. It consists of making the system able to recognize patterns and regularities in data, or as Christopher M Bishop describes it: *"The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories."*[4]. Knowledge Discovery in Databases (KDD) or data mining, also involves pattern recognition, since its goal is to discover patterns in large data sets like ours [22].

In this project, we will use it to discover fluctuating patterns in cryptocurrency prices that would've been missed by humans, so our RL system can make good trading decisions.

3.2.2.1 Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP) is a kind of Feed Forward Neural Network [105]. MLP consists of an ANN composed by at least three node layers.

Each one of the nodes uses a non-linear activation function [3.2.2.3]. MLP uses a ML technique called *backpropagation*[3.2.2.2]. Another key characteristic about MLP is that it can distinguish data that is not linearly separable. [109]

3.2.2.2 Backpropagation

Backpropagation involves a set of algorithms that are used to train an ANN. They follow a gradient descent method [106] that exploits the chain rule [104].

Its main feature is that it is an iterative, efficient and recursive way of calculating the optimum value for weights to improve the networks performance and results.

3.2.2.3 Activation Function

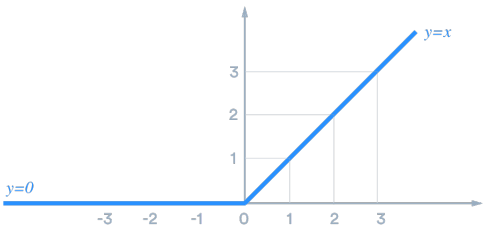
The Activation Function of a node is a function that determines the output of such node given its input. Linear activation functions are simple and useful to resolve easier computational problems. Non-linear activation functions help resolve more complex problems. [100]

The activation function we are going to use in this project is the Rectified Linear Unit (RELU) function which we will explain in Subsection 3.2.2.4.

3.2.2.4 Rectified Linear Unit (RELU)

The Rectified Linear Unit (RELU) is the most commonly used activation function in DL models. The function is characterized as the positive part of its argument. It returns 0 if it receives any negative input, and it returns the same value back if it happens to be positive. It could be written as Function 3.1.

$$f(x) = x^+ = max(0, x) \qquad (3.1)$$



Rectified linear units, allow for faster and effective training of deep neural architectures on large and complex data sets. [99]

As DanB in his article for Kaggle states: "It's surprising that such a simple function (and one composed of two linear pieces) can allow your model to account for non-linearities and interactions so well. But the ReLU function works great in most applications, and it is very widely used as a result." [17]

3.3 Libraries

For the development of this project, various open-source software libraries have been used. In this section we will do a quick overview of some of them.

3.3.1 PyTorch

PyTorch [58] is an open-source machine learning software library for Python, based on Torch [36]. The project is primarily developed by Facebook's [21] artificial-intelligence research group.

PyTorch provides two high level features:

- ***Tensor computation*** (like Numpy in Subsection 3.3.2) using General-Purpose Computing on Graphics Processing Units (GPGPU) in systems that support CUDA (Compute Unified Device Architecture) [16].
- ***DNNs*** built on a tape-based Automatic Differentiation (AD) system.

Figure 3.8: PyTorch

PyTorch's modules provide some extra functionalities that help in the development of ANNs: *Autograd* uses AD that saves time on one epoch. Even though you can implement your own algorithms, *Optim* is a module that provides various optimization algorithms used when building ANNs. Last, but not least, PyTorch's *nn Module* is the base class for all NN modules. We will use this, and other modules, in this project [111]

3.3.2 Numpy

NumPy [47] is an open-source software library for Python. It provides support for large, multi-dimensional arrays, along with a large collection of sophisticated high-level mathematical functions to operate on these arrays; as well as useful linear Algebra, Fourier transform and random number capabilities. The core functionality of NumPy is its *"ndarra"*, for n-dimensional array, data structure. [110].

Figure 3.9: NumPy

Therefore, we will use NumPy to manage our cryptocurrency data sets, store them in large arrays, and make it easier for our system to operate on them using NumPy's functions.

3.3.3 Pandas

Figure 3.10: Pandas

Pandas [54] is an open-source software library, written as a Numpy's extension, for data manipulation and analysis in Python.

Pandas provides tools to manage numerical tables and time series, which will come in handy, since our dataset is a time series with numbers for the prices over the years. It also provides ease for grouping, merging and querying pandas data structures, as well as visualization mechanisms.

Pandas' two main data structures are:

- ***Series:*** One-Dimensional labeled object, capable of holding any data type. It would be similar to an array list, a dictionary or a column in a table. It consists of indexed values, making it easy to store and retrieve data.

- ***Data Frames:*** Two-Dimensional labeled objects of potentially different types. It would be similar to a database or a spreadsheet. It could be seen as a dictionary of series that share the same index.

3.3.4 TensorFlow

TensorFlow [84] is an end-to-end open source platform for numerical computation using data flow graphs, used for ML applications such as ANN. It was developed by Google's [25] AI research team. It provides a stable Python Application Program Interface (API).

As they describe in their repository, "The graph nodes represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture enables you to deploy computation to one or more Central Processing Units (CPUs) or Graphics Processing Units (GPUs) in a desktop, server, or mobile device without rewriting code. TensorFlow also includes TensorBoard, a data visualization toolkit."

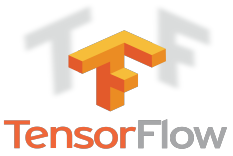


Figure 3.11: TensorFlow

It has a comprehensive and flexible ecosystem of tools and libraries that helps developers build and deploy ML powered applications. It supports training and deployment of models in the cloud, on-premises, in browser and on-device.

3.3.5 Scikit-Learn

Scikit-Learn [55] is part of a small core of open source packages for computing in Python, called the SciPy Stack (or The SciPy Ecosystem). Each extension or module is considered, and named, a SciKit. And Scikit-Learn is the module that provides learning algorithms.

Figure 3.12: Scikit-Learn

Scikit-Learn provides a mixture of supervised and unsupervised learning algorithms through an interface for Python. This is why it is considered an open-source library for data analysis and data mining [6].

The SciPy Ecosystem has some general and specialized tools to manage and compute data, to accomplish high-performance computing and productive experimentation.

Although there are many more, these are some of the key packages used in this project from said ecosystem:

- Productivity and high-performance computing packages:
  - ***The Jupyter Notebok:*** provides iPython functionality and other items on a web browser, for easy documentation of computations in a easily reproducible form.
  - ***Matplotlib:*** a popular and mature plotting package which generates 2D plotting with the quality for publications, and simple quality 3D plotting.
- Data and computation packages:
  - ***NumPy:*** the fundamental package for numerical computation. It defines the numerical array and matrix types, and does basic operations on them.
  - ***Pandas:*** provides high-performance data structures which are easy to use.
  - ***SciKit-Learn:*** a collection of algorithms and tools for machine learning.

3.3.6 Ta-Lib

Ta-Lib [83] is an open-source software library of Technical Analysis Indicators. "In finance, technical analysis is an analysis methodology for forecasting the direction of prices through the study of past market data, primarily price and volume" [113].

This library is widely used by trading software developers who need to perform technical analysis of financial market data. It provides about 200 functions, some of which will be used to acquire some key parameters that we will see ahead. It incorporates an API for Python, and it includes a Candlestick pattern recognition.

3.3.7 Zipline

Zipline [64] is a Pythonic, open source, algorithmic trading library. It is an event-driven system for backtesting systems. It is currently used in production as the backtesting and live-trading engine powering the crowed-sourced quantitative investment fund Quantopian [63]. As Stephan Jansen defines it in his book [31] "It automates the algorithm's reaction to trade events and provides it with current and historical point-in-time data that avoids look-ahead bias."

Some of the key features of this library include:

- Ease of Use.
- Includes many common statistics.
- PyData Integration. Input and output data are based on Pandas (3.3.3) DataFrames
- Statistics and ML libraries can be used to support development, analysis and visualization of trading systems

Figure 3.13: Zipline

3.3.8 Alphalens

Alphalens [61] is a Python Library for performance analysis of predictive (alpha) stock factors, as defined in their GitHub page. Alphalens works great with Zipline 3.3.7, and you can also find it at Quantopian [63].

Quantopian defines its use like: "Alphalens is a Python package for performance analysis of alpha factors which can be used to create cross-sectional equity algos. Alpha factors express a predictive relationship between some given set of information and future returns. By applying this relationship to multiple stocks we can hope to generate an alpha signal and trade off of it. Developing a good alpha signal is challenging; so where can we (Quantopian) make things easier, and where do you (the quant) add the most value? We figured that a common set of tools for analyzing alpha factors would have a big impact." [62]

Figure 3.14: Alphalens

3.3.9 OpenAI

OpenAI in a research laboratory. It's an open source, free organization, which purpose is "to ensure that artificial general intelligence benefits all of humanity". [1]

OpenAI is a non profit with investors such as Microsoft and founders such as Elon Musk. The corporations conducts research in artificial intelligence with the goal to develop a friendly AI development system.

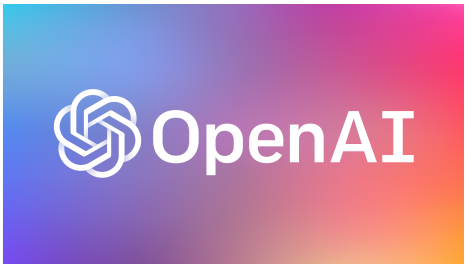


Figure 3.15: OpenAI

It has various different products and applications. Some of the most famous ones are:

- ***Gym:*** An easy to set up general-intelligence benchmark with many different environments.
- ***OpenAI Five:*** Is the name of the team of bots that play in the video game "Dota 2".
- ***RoboSumo:*** Bots that initially don't even know how to walk, that have the goal of learning to move around and interact with different environments.
- ***Retro:*** A platform of RL that is used to conduct research on RL algorithms.

3.3.10 OpenBlender

Figure 3.16: OpenBlender

OpenBlender [48] is a Self-service online platform that enables data scientists to enrich datasets with correlated variables from thousands of live-streamed open sources in an automated way.

OpenBlender profiles and transforms both structured and unstructured data (in the form of text) into a common numeric format.

It allows you to upload a dataset to their portal, merge it with different datasets that overlap in location or time interval and then download the enriched data to boost the performance of a ML project.

They also provide an API to do all these requests and actions.

CHAPTER 4

Model Building and Architecture

This chapter presents the methodology that this thesis followed.

First, we will start of by presenting the pipeline built for the correct development of this project.

Then, it will present the data selected to form our dataset, and what all the different fields represent. This will be the input of our project's pipeline.

Later, this chapter will describe the overall architecture of the project, explaining the connections between the different components involved in the development of the project. This can easily be identified by examining our project's pipeline represented in Figure 4.1.

It will also specify the model chosen for this project, along with the description of how it was built.

Last, but not least, it will characterize the different procedures adopted to achieve our solution, including the tuning of parameters to optimize our model.

4.1 Pipeline

In this section we will talk about the pipeline developed for the RL agent build during the development of this project.

In Figure 4.1 we can see a representation of the previously described pipeline with all the different steps necessary to obtain our agent's final outcome.

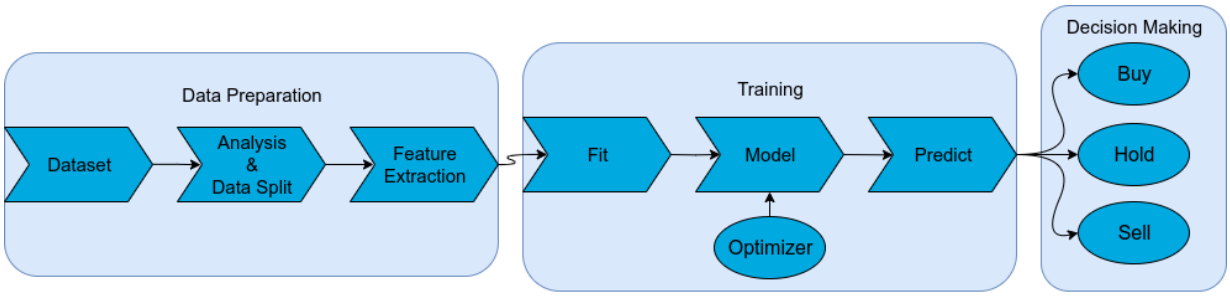


Figure 4.1: Pipeline

Data Preparation

As with every ML related project, the most important part would be to gather the correct input for out model. In this case, we selected the best data we thought could fit our purpose. We analyzed the dataset as we will present in Section 4.2.

Then we preprocessed the data as shown in Section 4.4. We tuned the parameters we thought would be more useful if presented in a different way, and we got rid of the ones that we deemed unnecessary or useless.

After that, we get to the last part of the data preparation. We decided to extract and calculate some new features that will be explained in Subsection 4.4.1

Training

When all the data has been gathered and processed, we move on to fit and train our model.

To improve its performance we decided to add an optimizer. We tested with a couple and we used the one that obtained the best results as we will see in Section 4.6.

Lastly, our model will move on to predicting what the Bitcoin's price will be the following day.

Decision Making

Finally, when it has the prediction, it will do the most important part, decide on one of the 3 options presented to it: whether to buy, sell, or hold up to 4 bitcoins.

4.2 Data Set

We'll start off by describing in more depth the dataset used in this project.

In order to correctly train our model, we wanted to use as much information as possible. For that purpose, we downloaded Bitcoin's historical data from Kraggle [34]. The downloaded dataset covers from 2012-01-01 to 2018-06-27.

We saw in figure 2.1 the graphical representation of such a data set. There are two important aspects to highlight about this dataset:

1. Data fluctuates a lot, in short periods of time, in an apparent random way. Bitcoin's price for a given day has no apparent relationship with the previous or following days.

2. 2017's peak. As we stated in Chapter 2, in December 2017, Bitcoin reached its highest value at $19,783.06. It was at that moment when its value started to decrease and, with a few up-spikes along the way, has continue to decrease ever since. This event made predicting the price, harder for our system, compared to related work previous to this tipping point.

4.2.1 Data Fields

Our data set, downloaded from Kraggle [34], is organized in a table with eight columns, which are: *Timestamp, Open, Close, High, Low, Volume_(BTC), Volume_(Currency) and Weighted_Price*. We will now describe what each one of these features means and show its properties.

- ***Timestamp:*** This column represents the Unix time (also known as POSIX time or UNIX Epoch time). This is the time elapsed since 00:00:00 UTC Thursday, 1 January 1970, in seconds [13].

  It is just a way of measuring time, so, in other words, this column represents the time when the bitcoin had the values in that row.

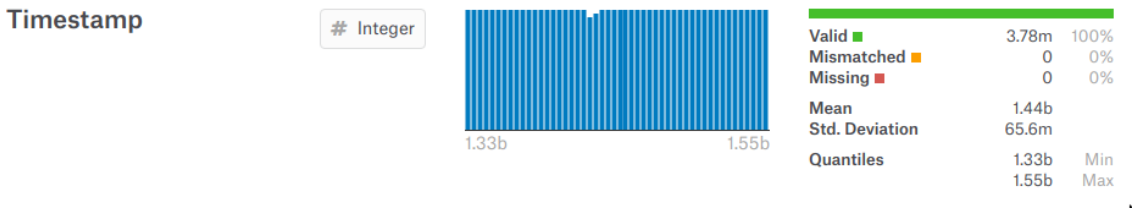


Figure 4.2: Timestamp in Unix time

- ***Open:*** This column represents the price of Bitcoin at the opening of the time interval, the price of the first trade that happened after the timestamp of the previous interval.

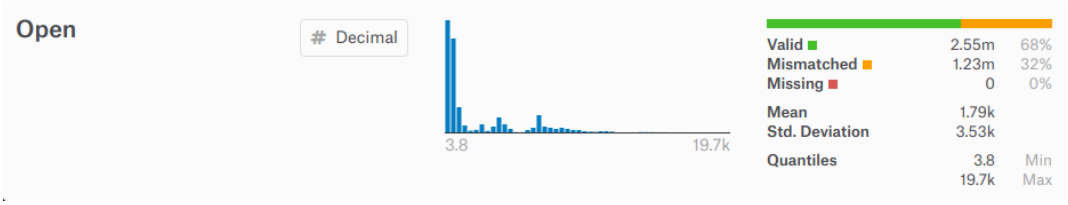


Figure 4.3: Bitcoin's Opening Value

- ***Close:*** This column represents the price of Bitcoin at the closing of the time interval.

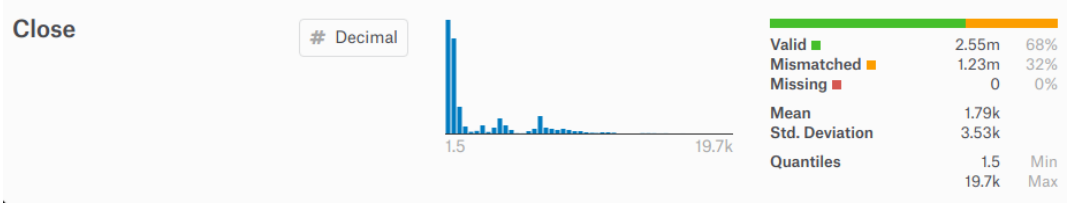


Figure 4.4: Bitcoin's Closing Value

- ***Highest:*** This column represents the highest Bitcoin price from all transactions executed during the interval.

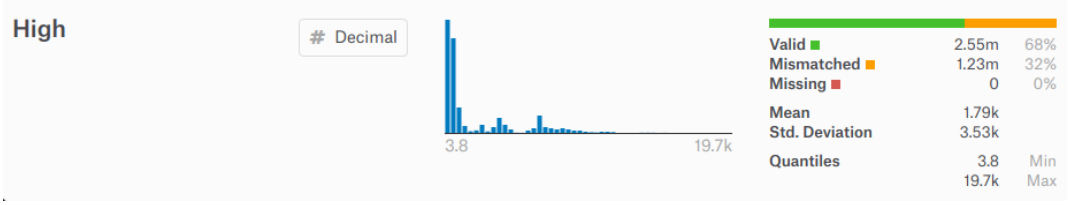


Figure 4.5: Bitcoin's Highest Value

- ***Low:*** This column represents the opposite to the previous one. It represents the lowest Bitcoin price during that interval.

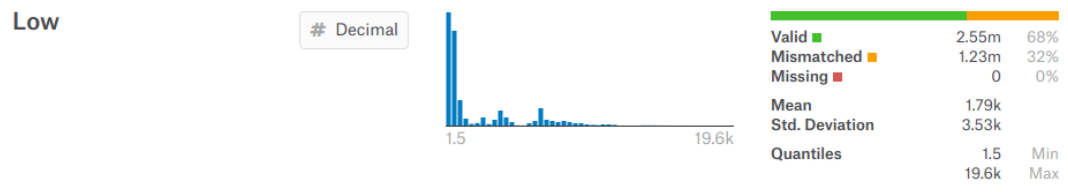


Figure 4.6: Bitcoin's Lowest Value

- ***Volume_(BTC):*** This column represents the sum of the value of all Bitcoins that were transferred during the selected time interval.

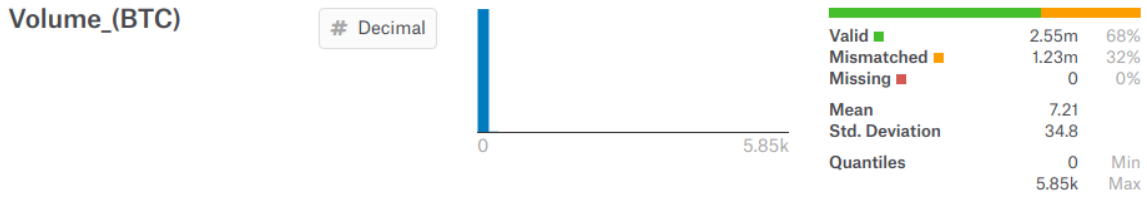


Figure 4.7: Sum of All Bitcoin Values

- ***Volume_(Currency):*** This column represents the sum of all US Dollars transferred during the selected time interval.

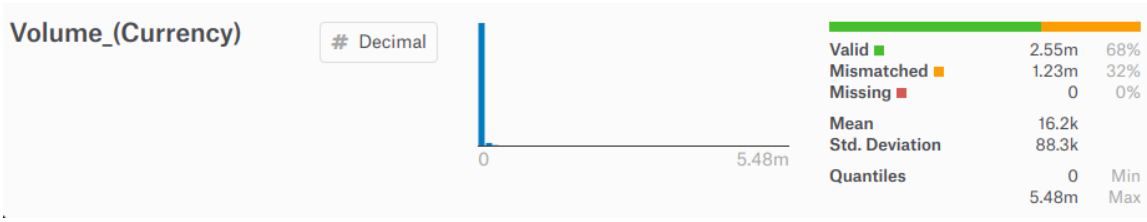


Figure 4.8: Sum of All US Dollar Values

- ***Weighted_Price:*** This column is derived from the two previous ones: *Volume_(BTC)* and *Volume_(Currency)*. By dividing the sum of all dollars traded by the sum of the value of all Bitcoins, we can get the weighted average price of Bitcoin during a selected time interval. As we can see in Ecuation 4.1.

$$Weighted\_Price = \frac{Volume\_(Currency)}{Volume\_(BTC)} \tag{4.1}$$

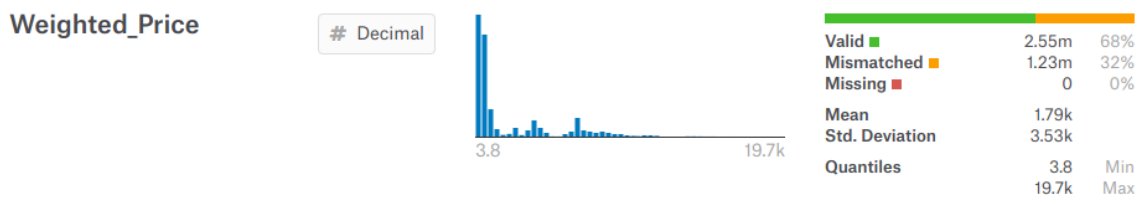


Figure 4.9: Weighted Price

4.3 Assumptions

A key aspect about all ML projects, is to find a balance between complexity and performance, in order to obtain the best desirable outcome by the easiest method possible. The more complex the system is, the better the results it will obtain, but this will be by sacrificing performance; the system will be slower and harder to build.

During the development of this project some assumptions were made in order to simplify it, making it as least complex as possible, obtaining the best possible outcome. We will describe some of this assumptions in this section:

- The first assumption this project made, is the following: **People will trade in similar patterns regardless of Bitcoin's price**. People's decisions will stay the same regardless of whether we are in November 2016, with a price of about $700, or in December 2017, with a price of about $20.000.

  This assumption was made for two main reasons:

  First, not all people use the same trading patterns. An event, or price, that for one person could be a sign to sell, could be a sign to buy for another person. People that sell, hold or buy will always find an excuse to sell, hold or buy. The same amount, could be a high value for some people, and a low value for others.

  Second, since, as we explained earlier, our system will be using RL, therefore being much more "hands-off", it may learn some price-depending patterns on its own.

- The second assumption made in this project was to believe that **the closing price for a given time period (*Close*) will not differ by much from the opening price of the following time period (*Open*)**. By comparing the opening and closing values for a certain time interval, we can know if Bitcoin's price went up or down during such time interval. By comparing several price variation in succeeding intervals we can come to the conclusion that if it has been going up or down for some consecutive intervals, it has gone up or down during the time containing those intervals. This would of course be a false conclusion, if the closing value of a given interval is significantly different from the opening value of the following one.

  Since this case rarely happens, this project will therefore make the assumption that this doesn't happen at all.

- Since we assumed that the actual Bitcoin value is irrelevant regarding peoples trading choices, and that the closing value of an interval will not differ from the opening value of the following one, we can focus on looking at the value change explained in the previous point. **We will focus on the delta between opening and closing prices within a time interval**. If the delta is positive, we will know the value went up during that interval. If it is negative, the value will have gone down. Last, if delta equals zero, the value will have remained the same.

- As Packtpub realized: "**Not all data in the dataset is valuable**. The first records are not informative, as price changes are rare and trading volumes are small. This can affect the model we are training and thus make end results worse." [76] That is why the first rows are eliminated from the dataset. Also as ThirstyScholar correctly pointed out in his last commit, there's an error in the data for 2017-04-15 23:00:00 so we also got rid of that time entry.

- The last, assumption made in this project is that the **prediction is made in an isolated system**. The system does not take any other feedback from the outside, such as news, stock exchange feeds, or any others. The only data used is the one specified in Subsection 4.2.1. (This, as explained in Chapter 6 and Section 7.4 could change is we decided to include more information, such as business or Bitcoin news feeds, or any others such as values of stock from other companies or products)

4.4 Preprocessing

One of the most important parts of the data-science pipeline after data collection (which is, in a sense, outsourced; we use data collected by others) is data preprocessing — clearing a dataset and transforming it to suit our needs.

It is very important that we carefully manage and curate data to avoid look-ahead bias by adjusting it to a Point in Time (PIT) basis. This means that data may only reflect information available and known at the given time. Therefore, we cannot use the simple K-Fold validation since this would take chunks of future data to predict past data.

In order for our RL agent to lean in the best way possible, it needs more input data than just bitcoin's prices. To help with this we used 18 different technical indicators, also used by Vincent Poon, that are representative of the state of the market and are somewhat predictive. These indicators are the ones shown in table 4.1.

We also included 5 state variables which represent the five possible bitcoin actions, between hold 0, 1, 2, 3 or 4 bitcoins for each trading action.

4.4.1 Indicators

Name	Description	Type
r	Bitcoin Return	Return
r_1	Return from 1 period prior	Lagged Return
r_2	Return from 2 periods prior	Lagged Return
rZ12	Zscore(r,12)	Price Level for 3 hours
rZ96	Zscore(r,96)	Price Level for 1 day
pma12	Zscore(p / avg(p,12) - 1, 96)	Change in Price for 3 hours
pma96	Zscore(p / avg(p,96) - 1, 96)	Change in Price for 1 day
pma672	Zscore(p / avg(p,672) - 1, 96)	Change in Price for 1 week
ma4/36	Zscore(p,4) / avg(p,36) - 1, 96)	Change in Price 1 hour / 9 hours
ma12/96	Zscore(p,4) / avg(p,36) - 1, 96)	Change in Price 3 hours / 1 day
ac12/12	Zscore([p / avg(p,12)] / avg[p / avg(p,12), 12], 96)	Acceleration in price for 3 hours
ac96/96	Zscore([p / avg(p,96)] / avg[p / avg(p,96), 12], 96)	Acceleration in price for 1 day
vZ12	Zscore(v, 12)	Volume Level for 3 hours
vZ96	Zscore(v, 96)	Volume Level for 1 day
vZ672	Zscore(v, 672)	Volume Level for 1 week
vma12	Zscore(v / avg(v, 12) - 1, 96)	Change in Volume for 3 hours
vma96	Zscore(v / avg(v, 96) - 1, 96)	Change in Volume for 1 day
vma672	Zscore(v / avg(v, 672) - 1, 96)	Change in Volume for 1 week
vol12	Zscore(std(r, 12), 96)	Volatility Level for 3 days
vol96	Zscore(std(r, 96), 96)	Volatility Level for 1 day
vol672	Zscore(std(r, 672), 96)	Volatility Level for week
dv12/96	Zscore(std(r, 12) / avg[std(r, 12), 96], 96)	Change in Volatility Level 3 hours / 1 day
dv96/672	Zscore(std(r, 96) / avg[std(r, 96), 672], 96)	Change in Volatility Level 1 day / 1 week

Table 4.1: Vincent Poon Indicators

In finance, the ***return*** parameter represents the profit on an investment. Accordingly, the *rate of return* represents the profit on an investment over a period of time, expressed as a proportion of the original investment. This is why we chose to use it in our project. The return $\boldsymbol{r}$ represents the return rate obtained after one trading period, and it is calculated as shown in Formula 4.2. We chose to use the logarithmic return over price or raw returns because, as Quantivity explains in their post *Why Log Returns* [60] "the benefit of using returns is *normalization*, [...] *log-normality*, [...] *approximate raw-log equality*, [...] *time-additivity*, [...] *mathematical ease*, [...] and *numerical stability*."

$$Bitcoin's\_Return = \ln \frac{Price\_Period\_1}{Price\_Period\_2} - 1 \tag{4.2}$$

The ***Zscore*** function represents the number of standard deviations from the mean, a data point is. The ***Zscore*** function is calculated as shown in in Formula 4.3. With the Zscore calculated, we calculate the mean and the standard deviation using pandas' rolling function. Using different window sizes (the last argument of the Zscore formula) we get 4, 12, 36, 96, 672 periods which correspond to 1 hour, 3 hours, 9 hours, 1 day and 1 week worth of data.

$$Zscore = \frac{Score - Mean}{Standard\_Deviation} \tag{4.3}$$

4.4.2 Cleaning the Dataset

This dataset contains eight columns or fields as we saw in Subsection 4.2.1. From this eight, we will only use six of them, thus, we will remove the two unused ones. We decided to delete the field "Volume_(Currency)" and the field "Weighted_Price". Since the only volume remaining in the dataset is "Volume_(BTC)" we will rename it to simply "Volume".

To make the dataset even smaller, we decided to take advantage of the index of every row to store the "Timestamp" in UTC format, thus being able to also remove the first column of our dataset "Timestamp" and having only five columns left.

We also noticed that, at least at first glance, the date and time of the data is not related to the price. Regardless, it was decided to keep the field "Timestamp, since, first, the agent might pick up some relationship we are currently not thinking about. And second, one of the thought future lines of work, would be to include also data from different markets, or news from different days, so this way the agent will be able to establish relationships based on the time and date. We changed the data from this column to a more readable format (DateTime format instead of UnixTime).

4.5 Model Building

After preprocessing the data and calculating the previously mentioned indicators, we create the RL agent.

Taking into account everything we have mentioned up until now, we have as the input of our ANN the 18 indicators, 5 columns of our dataset, and the 5 state variables representing the possible actions, which makes a total of 28 input features for our model.

As we commented in Chapter 2, we define the input layer as a Linear module, and a RELU module, sequentially applied. After that, the hidden block would consist of 32 hidden layers, with 23 neurons each.

4.5.1 Algorithms

As Kung-Hsiang, Huang (Steeve) explains it [41] there are different types of RL algorithms:

- ***Model-Based Algorithms:*** When speaking of RL, the model stands for the simulation of the dynamics of the environment. With a model based algorithm, the model learns the transition probability from the pair of current state and action to the next state, ergo the agent will know how likely to enter a specific state given a new current state and action. This Model-Based algorithms, however, will need space to store all the previous states.
- ***Model-Free Algorithms:*** Model-Free Algorithms on the other hand, rely on trial-and-error to update its knowledge. Therefore not requiring space to store all the combinations of states and actions. Inside this model-free category we can find two types of algorithms:
  - ***On-Policy:*** An on-policy learns the value based on its current action a derived from the current policy.
  - ***Off-Policy:*** An off-policy learns the value based on the action $a*$ obtained from another policy. In Q-Learning, such policy is the greedy policy, as we will see in Subsection 4.5.3.2.

For the development of this project we have decided to use an *Off-policy* by using the Q-Learning technique.

4.5.2 Q-Learning

Q-Learning is an off-policy, model-free RL algorithm based on the Bellman Equation:

$$v(s) = \mathbb{E}[R_{t+1} + \lambda_v(S_{t+1})|S_t = s] \tag{4.4}$$

Equation 4.4: Bellman Ecuation [81]

The goal is to maximize the Q-value, which can be re-written in the form of:

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + ...|s,a] \tag{4.5}$$

Equation 4.5: Q-Value [81]

Before the learning begins, Q is initialized to an arbitrary fixed value. Then, at each time $\mathbf{t}$ the agent selects an action $\mathbf{a_t}$, observes a reward $\mathbf{r_t}$, enters a state $\mathbf{s_{t+1}}$ and $\mathbf{Q}$ get's updated. Hence, the Q-Learning algorithm is a simple value iteration update that uses the weighted average of the old value and the new information [97]:

$$Q^{new}(s_t,a_t) \leftarrow (1-\alpha)\cdot \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Figure 4.10: Q-Learning Update Equation

is the learning rate [95].
Where:

- $\mathbf{r_t}$ is the reward received when moving from the state $\mathbf{s_t}$ to the state $\mathbf{s_{t+1}}$.
- α is the learning rate. It determines to which extent, new information overrides old information. It ranges from a factor of 0, which would make the agent only take into account past information, to a factor of 1, which would make the agent consider only the most recent information.
- γ is the discount factor. It determines the importance of future rewards. It ranges from a factor of 0, which will make the agent "short-sighted" by only taking into account present rewards, to a factor of 1, which will make the agent fight for the highest reward in the long run.

4.5.3 Value Update Methods

The two most common value update methods, closely related to Q-Learning are Policy Iteration and Value Iteration.

4.5.3.1 Value Iteration

Value Iteration is based in one simple principle. It tries to update the value function V based on the optimized version of the Bellman Equation we saw in 4.4

$$v_*(s) = max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{4.6}$$

Equation 4.6: Optimized Bellman Equation [79]

And the algorithm of value iteration is as follows:

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Figure 4.11: Value Iteration Algorithm [79]

And after the iteration finally converges, we can straight-forwardly derive the optimal policy by just applying an argument-max function for all states.

The main goal is to update the value using the Optimal Bellman equation. The value that is finally converged is the optimal value in the current state. Therefore, as long as it converges, we obtain the optimal policy. Thus, it is called value iteration because it is based on updating the value.

This differs from the Policy Iteration we will see in Subsection 4.5.3.2, even though it also uses the Bellman equation, because the final converged value is the optimal value of the current policy (so called, evaluation policy), where the purpose it to get a new policy for subsequent policy improvements.

4.5.3.2 Policy Iteration

Policy Iteration is usually divided into two steps: Policy Evaluation, which purpose is to update the Value Function. And Policy Improvement, which uses the greedy policy to generate new samples for the policy evaluation.

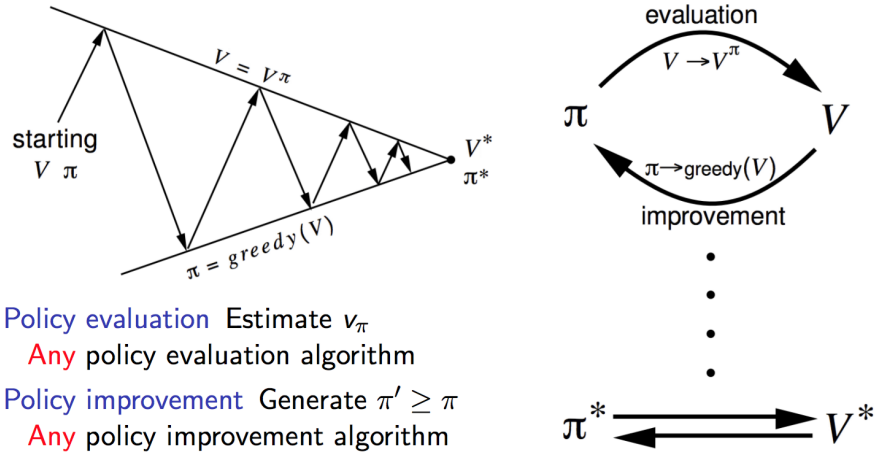


Figure 4.12: Policy Iteration [79]

It essentially uses the policy evaluation estimate a value V. Then, using the greedy policy, it obtains a policy improvement, which is used again to get a new value V. This process iterates until it converges to the optimal value.

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\left[r + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $a \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
   If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2

Figure 4.13: Policy Iteration Algorithm [79]

4.6 Optimizing the model

As we saw in Section 4.1, we wanted to use an optimizer in order to improve our RL agent's performance.

We decided to use PyTorch's package, Torch.Optim [59]. This package implements various optimization algorithms. We tested out the two most common algorithms with our model:

- ***Stochastic Gradient Descent (SGD):*** This is an iterative method for optimizing an object with suitable smoothness properties. It replaces the actual gradient, calculated from the entire data set, by an estimate thereof, calculated from a randomly selected subset of the data [98].
- ***Adam:*** The Adam optimization algorithm is an extension to SGD that has recently seen broader adoption for deep learning applications.

After some research and testing, we proved that Adam works better than SGD. In figure 4.14 we can see a comparison of the results obtained using both algorithms. The upper graph represents the Cumulative Logarithmic Returns and, the lower one, the Maximum Drawdown profile.

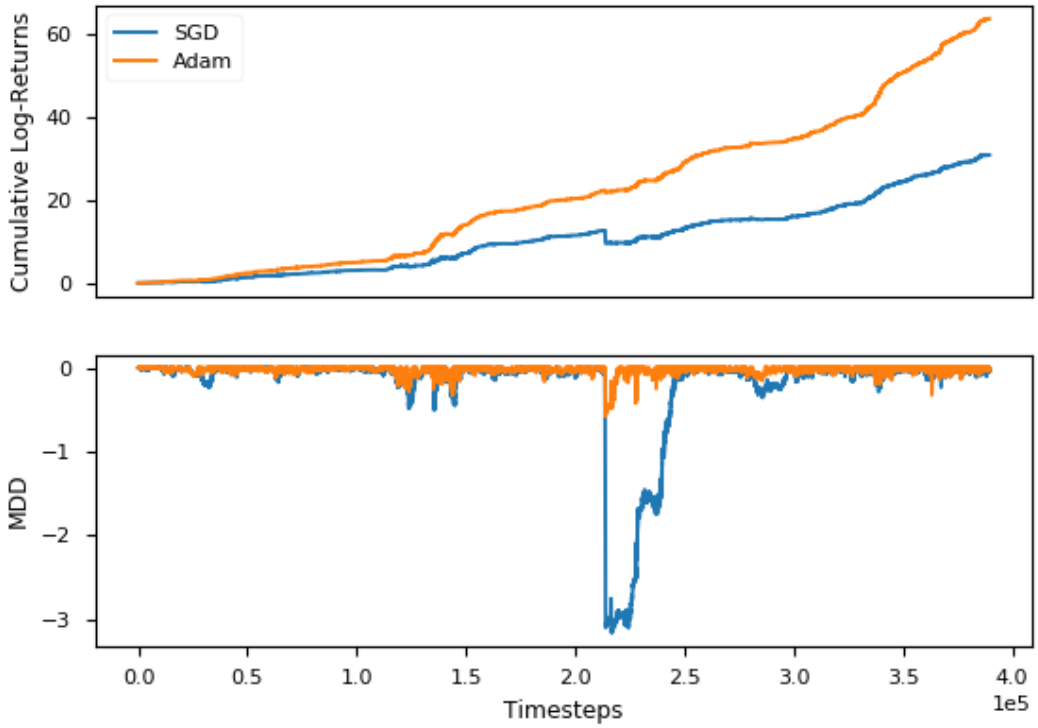


Figure 4.14: Adam Vs SGD

Therefore, we decided to establish Adam [39] as the optimizer algorithm we were going to use for our model.

Adam's optimizer has numerous parameters that can be tuned to obtain a better result. After testing with several tuning parameters and checking the obtained results, we decided to use an optimizer with a weight decay of 10^{-6} and a learning rate of 10^{-3}.

CHAPTER 5

Training and Obtained Results

This chapter explains the training process followed by our RL agent.

It also focuses on analyzing the obtained results using different techniques, comparing them among themselves and explaining their meaning and any possible outliers or pieces of data worth mentioning.

Later, it presents a comparison between our obtained results, and the results obtained by other projects, strategies, techniques or approaches.

5.1 Training

As it is usual in ML, we chose to divide the dataset into two groups, one to train our model and one to test it's performance afterwards. We used 70% of the dataset to train our model and 30% for testing purposes.

Then we create our agent as explained in Section 4.5. As we said previously, we used Q-Learning as our algorithm. We decided to use the following parameters:

1. In Features = 28
2. Hidden Layers = 32
3. $l_r = 10^{-3}$
4. $Weight\_Decay = 10^{-6}$
5. Discount_Factor = 0.99
6. Low
7. Close
8. Weighted Price
9. Volume (BTC)
10. Volume (Currency)

We establish to train the model with 5000 random episodes and we tested our model with different periods for out candles.

5.2 Results

After having trained our model with these parameters and algorithms, we decided to test out results by grouping our data into 15-minute candles. Each candle will then contain the information of all the attributes we saw in Chapter 4 about the price fluctuation only in the last 15 minutes.

As we have stated before, given the nature of this project and the type of ML that is being applied, it is hard to know whether the agent is performing well or not, since there's no right or wrong choice per se. It will obviously never obtain all profits without loosing any money ever.

It would also be impractical to compare it with a real-person's performance when it comes to trading, firstly for obvious reasons, we will not have a real-person trade simultaneously just to compare the obtained results. Also, who's trading decisions should we choose as "correct"? Each specific person has his or her own trading strategies, secrets and techniques, and all of them could argue that theirs is better than anyone elses.

We would then need to use known fixed trading strategies for comparison. Strategies that have been proven to work and that are "hands-free", with none or very very little human interaction, so that our agent can also follow them without any human supervision and just following a set of predefined rules.

Therefore we decided to compare the results obtained by our RL agent in a graph along with two other trading strategies, that we consider are two of the most popular trading strategies or techniques among traders and two of the ones that give the best results, needing the lease amount of interaction from any human being:

- ***Buy and Hold***: It's a passive investment strategy. The investor buys stocks(or, in our case bitcoins) and holds them for a long period, regardless of fluctuations in the market. If an investor uses a buy-and-hold strategy, he or she has no concern for short-term price movements and/or technical indicators. It's a good strategy that requires little involvement from the investor. Many legendary investors such as Warren Buffett and Jack Bogle praise the buy-and-hold approach as ideal for individuals seeking healthy long-term returns. In our case, we programmed our agent to always hold at least 2 bitcoins (out of the 4 bitcoins available to him) regardless of the price changes on a daily basis.

- ***Momentum***: A momentum strategy is another passive trading technique in which traders, buy and sell stocks (bitcoins in our case), according to the recent price trends. Momentum traders bet that an asset price that is moving strongly in a given direction will continue to move in that direction until the trend loses strength, meaning that while the trend of growth on a stock is stable, it will continue to be that way, therefore, if it's growing, the trader should hold on to his or her stock, and sell it otherwise. In this case, our agent has being programmed to hold all 4 bitcoins if the price is above the average price of the previous thirty periods. Otherwise, it will sell everything, holding zero bitcoins.

Since this is a big dataset and the resources were limited, it took a long time to process each time we executed our code to run our agent. We realize that, both time and computing resources, were a key factor to be considered when building and executing any project. Therefore, in an attempt of saving both, time and computing resources, we decided to test our results by creating candles with more data compressed in them, thus having less candles to process.

Before trying this, first we tried out with 1-min candles. We know that this creates a greater number of candles, therefore increasing the time and computing resources, but we wanted to do it just to be able to later try comparing the obtained results, since sometimes, time and computing resources are not an issue or even if they are, good results are more important. Then, we tested our model's performance with 1-hour candles and 12-hour candles, saving our two valuable resources. We will now see the obtained results with all of these scenarios, in the following Subsections.

We decided to include two examples of each type of organization of candles. Each one of those examples shown in the following Subsections contains two different graphs:

- The upper one represents the **rewards** gained by the agent following its corresponding technique. The reward is calculated as the sum of the discounted returns from the step in question to the end of the episode. The returns are calculated as we saw in Equation 4.2. This would be the result that receives the most attention since the final goal of our agent would be to let it trade and make money for us in the long run without interaction needed in our side.

- The lower graph represents the **Maximum Drawdown (MDD)** which is the maximum observed loss from a peak to a trough. This decline during a specific period of time, is usually represented as the percentage between the peak and the subsequent though. It is an indicator of downside historical risk over said specified time period. The MDD, is also the negative half of standard deviation in relation to the stock's price. One of the ways it can be calculated, would be with the formula we see in Equation 5.1.

$$\text{MDD} = \frac{TroughValue - PeakValue}{PeakValue} \tag{5.1}$$

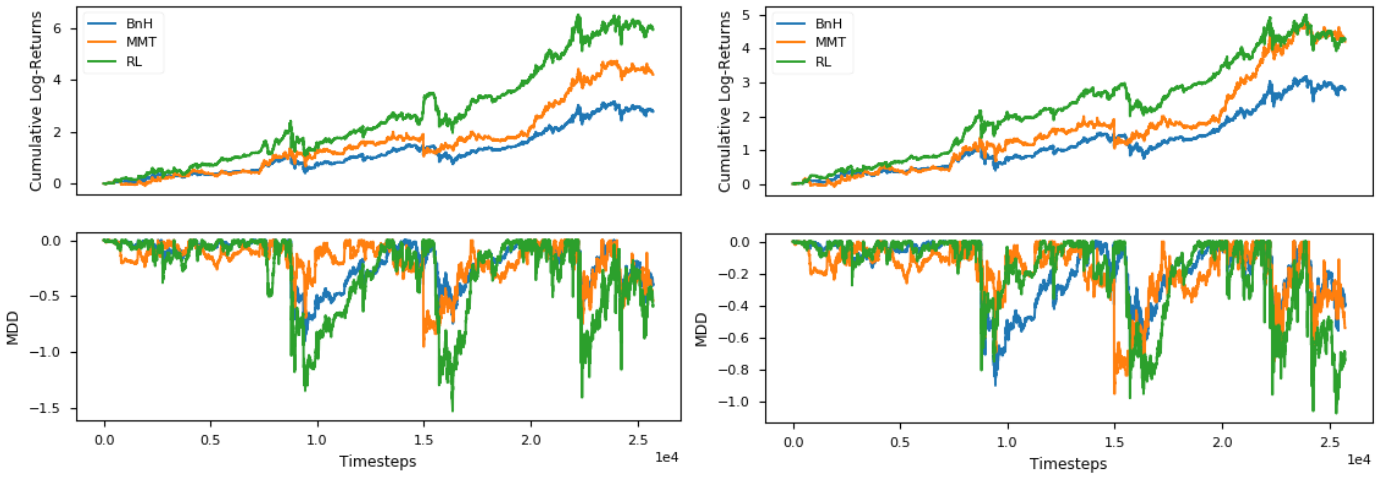
As we will see in Subsections 5.2.1 and 5.2.2, when grouping our data into the 1-Minute and 15-Minute candles, our RL agent obtained better results, when compared to the results obtained when we just followed the, previously explained, Buy and Hold or Momentum strategies. It obtained higher rewards, although, sometimes, it got a bigger MDD. Noticeably, the best results were obtained with the 1-Minute candle grouping, but, as we would have expected, this took too much time and resources to process, and since the difference between the rewards achieved with the 15-Minute candle grouping, was not very high, and, notwithstanding, a lot of computational time and resources were saved with the latter, we concluded that the optimal solution is the 15-Minute candle grouping approach.

Contrariwise, the results obtained when using the 1-Hour and 12-Hour groupings, were, as expected, worst, since it grouped more data and wasn't as accurate. It did save a lot of time and computational resources, but its performance wasn't as good as the two common techniques for which we would not need any kind of ML. Therefore we decided not to use it for our agent.

On the other hand, it was quite surprising that if we kept trimming data, by only taking into account the 1-Hour or 12-Hour candles, the best results were obtained by the Momentum strategy. Our RL agent did good as well (better in fact, than the buy and hold strategy) but the Momentum technique obtained better rewards and less MDD as we will see in the following subsections.

5.2.1 15-Minute Candles

As we said before, this was our preferred optimized grouping method, or technique. When taking into account the relationship between results obtained versus the time and computational resources spent, this 15-Minute candle grouping obtained the best results in the least amount of time and using the least resources. Our RL agent really surpassed our other two strategies almost every time throughout all the tests. We can see two examples of these obtained results in Figure 5.1.



(a) Example A

(b) Example B

Figure 5.1: 15-Minute Candles

It can be seen in the two examples, at first it might seem that the three techniques start with a similar performance, but later on, our RL agent, by the use of all the previously discussed ML and ANN methods, was able to obtain a reward significantly higher than the momentum strategy (described as MMT in the graph's legend) and almost doubling the one obtained by the buy and hold strategy (described as BnH in the graph's legend).

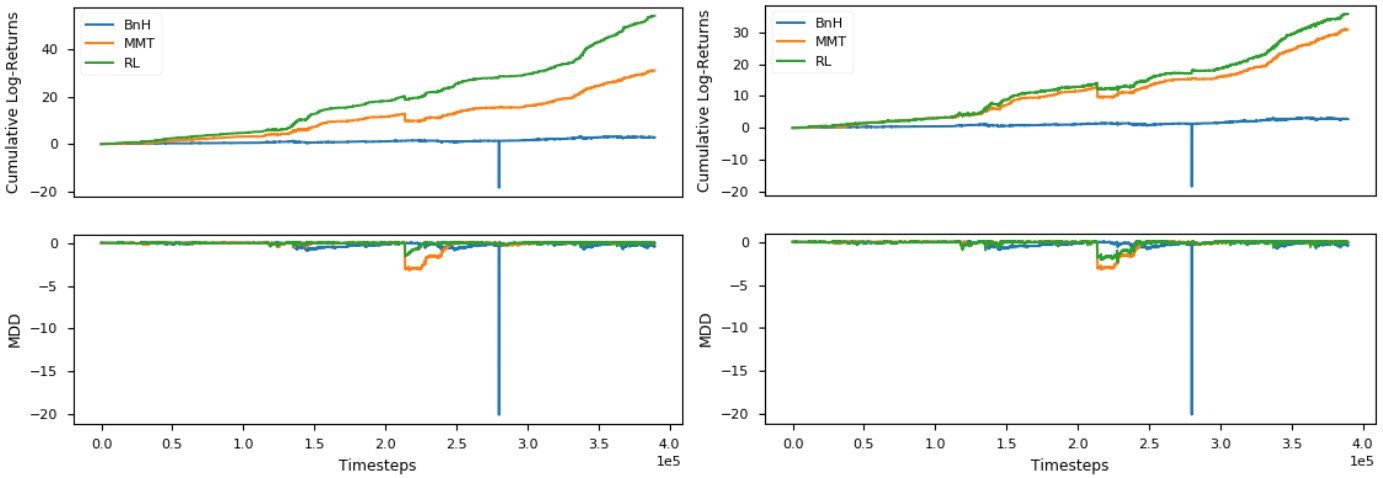
In Table 5.1 we can see the results obtained by these three strategies for the two examples shown in Sub-figures a and b.

(Subfig a / Subfig b)	**Buy and Hold**	**Momentum**	**RL**
Annual Return	620% / 620%	955% / 955%	1252% / 990%
Max Drawdown	-89% / -89%	-90% / -90%	-143% / -103%

Table 5.1: Results 15-Min Candles

5.2.2 1-Minute Candles

This second approach, while obtaining the best rewards, it also obtained the highest MDD percentage. As we expected, it was also the one that took longer time and used more resources to process and obtain the desired results, since it has more candles that need to be processed. We can see two examples of this grouping strategy's obtained results in Figure 5.1.



(a) Example A

(b) Example B

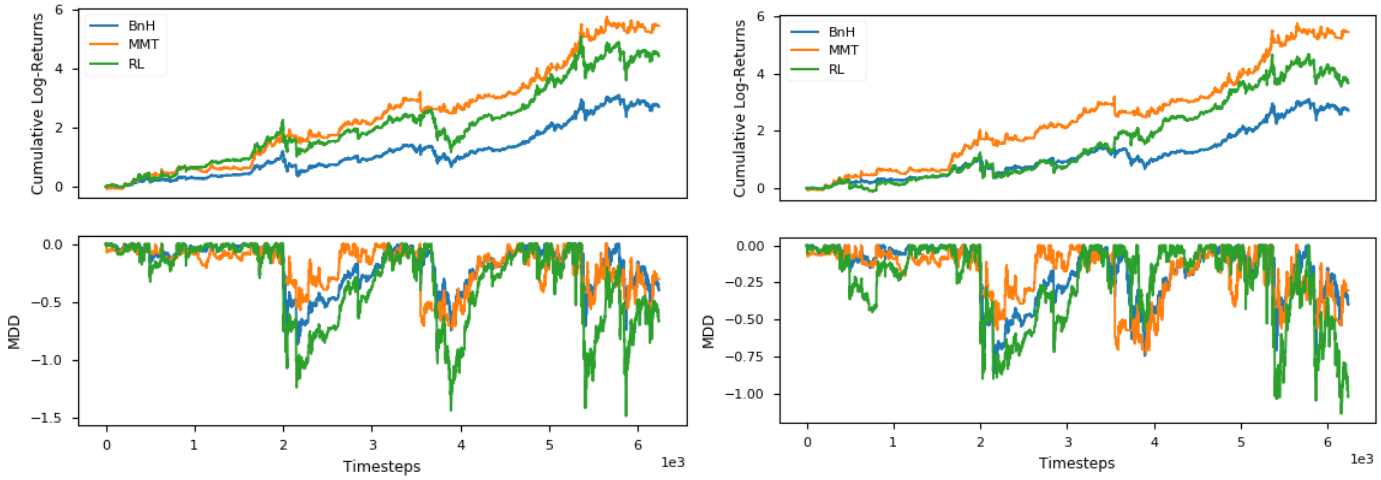
Figure 5.2: 1-Minute Candles

Nonetheless, when looking at the relationship of the rewards obtained by our RL agent, with the ones obtained by, for instance, the momentum strategy, it is very similar to the one we obtained in Subsection 5.2.1 with the 15-Min grouping. Thus, since we really would like to evaluate the performance of our "hands-off" RL agent in comparison the other well known "hands-off" strategies, we can conclude that the preferred grouping method would be the 15-Minute candle grouping.

Furthermore, as we can see in both MDD graphs, there was an outlier which got an extremely high MDD for the buy and hold strategy (around -20 in comparison with -3 and -2 that got the RL and momentum strategies). It also had a very bad effect in the obtained rewards, where it ended up loosing money (It got to a -18 cumulative logarithmic return). Taking also this into account, we reaffirmed our decision of not using this grouping formation.

5.2.3 1-Hour Candles

The 1-Hour candle approach is a good alternative if we don't have the necessary time or resources to process a big amount of candles like we did with the groupings in Subsections 5.2.1 and 5.2.2. Obviously the results are not as good as the ones obtained with previous techniques. We can see in Figure 5.3 that the return at the end is a little bit lower than the momentum strategy but, it's still better than the buy and hold strategy.



(a) Example A

(b) Example B

Figure 5.3: 1-Hour Candles

Nonetheless, they all start with pretty similar performances and the RL agent's performance even surpasses the momentum strategy in a couple of occasions. Of course, we should also look at the MDD, and it can be seen that the performance of our RL agent is not so good either.

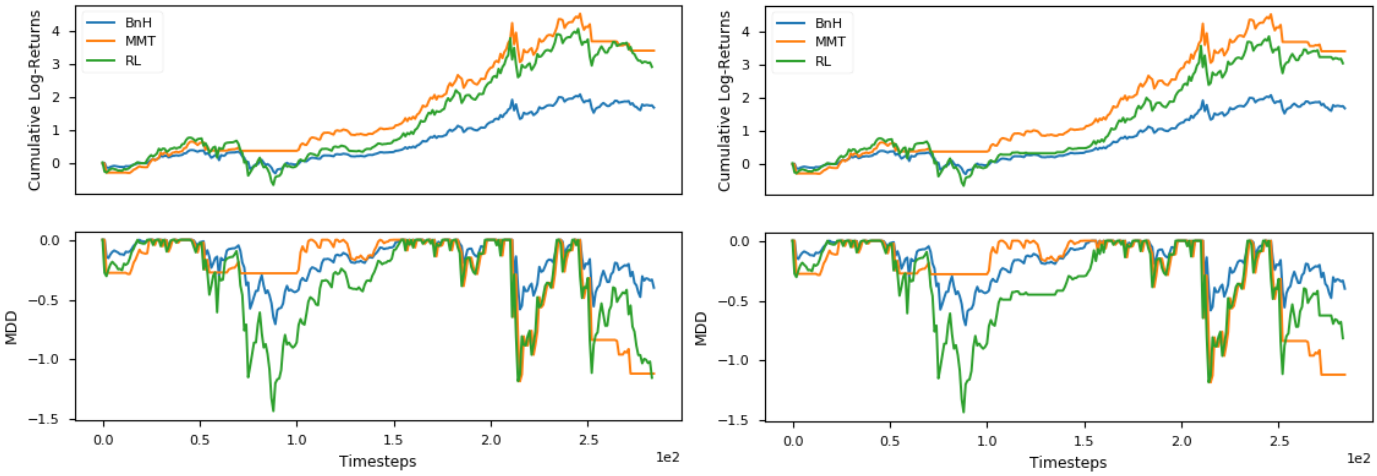
We can see in Table 5.2 the results that were obtained by our three grouping strategies in both examples.

(Subfig a / Subfig b)	**Buy and Hold**	**Momentum**	**RL**
Annual Return	580% / 580%	1180% / 1180%	1040% / 995%
Max Drawdown	-80% / -80%	-73% / -73%	-143% / -103%

Table 5.2: Results 1-Hour Candles

5.2.4 12-Hour Candles

This 12-Hour candle approach, as expected, is the one that got the worst results of all the grouping strategies. It was, however, the one that used the least amount of time and the least computational resources. In this case, the rewards obtained when using all three trading strategies (Buy and Hold, Momentum and RL) is lower than with the other grouping techniques, but the MDD is better.



(a) Example A

(b) Example B

Figure 5.4: 1-Minute Candles

If we take a closer look at the graphic, we can see that the RL was in fact very close to the Momentum grouping strategy, and better than the Buy and Hold approach, so the results are not really bad, it was still able to trade successfully and obtain benefits.

In Table 5.3 we can see the results obtained by these three strategies for the two examples shown in Sub-figures a and b.

(Subfig a / Subfig b)	**Buy and Hold**	**Momentum**	**RL**
Annual Return	395% / 395%	964% / 964%	780% / 705%
Max Drawdown	-70% / -70%	-125% / -125%	-143% / -143%

Table 5.3: Results 12-Hour Candles

5.3 Selected Grouping Method's Outcome

As we have seen in the previous sections of this chapter, after examining the results obtained by the three different methods, comparing the obtained rewards and MDD with another two known and commonly used "Hands-Free" trading strategies, the Buy and Hold strategy and the Momentum strategy, we decided that the best grouping technique to be used would be to group the candles in 15-Minute periods.

Our RL agent, with the organized 15-Minute candles and the selected inputs has shown to be effective in algorithmic trading of Bitcoins, creating a technique that is both unique and outperforms other commonly known trading strategies.

CHAPTER 6

Include News

After achieving the results shown in Chapter 5 we kept looking for ways to improve our results and data we could add to improve our agent's performance.

Since Bitcoin's value, and cryptocurrencies' in general, are not linked to the value of any other currency, company, asset or even country, we concluded that one of the factors that influence Bitcoin's price the most, might be business news from well-known papers or tv channels that make people want to trade (Buy or Sell) depending on what the media says about Bitcoins and the market.

Therefore we search for ways to include some sort of news feed regarding Bitcoin into our model. We found that Federico Riveroll used Python to attempt to predict Bitcoin Prices using news from Fox [66]. So we decided to try and replicate his work to see how using News would impact our Bitcoin price prediction.

This chapter will illustrate the obtained results and conclude whether we think this would be useful new information to add to our model or not.

6.1 Dataset and Goals

As we saw in Chapter 3, OpenBlender [48] contains many different datasets. With just creating a free account in their portal you have access to may datasets with lots of different information and you even have the ability to merge several dataset together or even with a dataset of your own.

There were many bitcoin-related datasets but for this chapter we decided to use OpenBlender's Bitcoin dataset [49]. We decided to use this dataset because, aside from the parameters we had in the dataset used in this project, (Volume, Timestamp, Price, High, Low, Open, Close (renamed Price)) we have an extra parameter named "Change". As we can see represented in Figure 6.1, this value represents the percentage change of the closing price of the day (Price) with respect to the opening price of the day (Open). Therefore it represents the price's increment over a specific day, having negative percentages for when we have a price decrement.

	volume	timestamp	price	high	low	open	change
0	31.08K	1584723600	6205.8	6897.0	5687.6	6173.5	0.52%
1	21.13K	1584550800	5416.2	5452.7	5033.9	5339.0	1.33%
2	41.78K	1584378000	5058.5	5381.9	4468.0	5381.9	-6.03%
3	18.64K	1584205200	5218.2	5673.2	5099.0	5621.8	-7.24%
4	116.68K	1584032400	4927.0	8001.7	4612.1	7975.3	-38.18%

Figure 6.1: OpenBlender's Bitcoin Dataset Head Rows

This chapter's objective would be to be able use some kind of news feed to predict price variations in Bitcoin.

To keep things simple, in this chapter we will focus only on predicting if the price will rise (Change value greater than 0) or fall (Change value less than 0) the following day. We would like to do this to know if it would be possible to predict a binary outcome (increment or decrement in the price) by relating the price with the business news of the previous days. If this value is well-enough predicted, we could insert it as a new parameter for our RL agent, so it can make a decision on how to trade having not only the past values of the price of Bitcoin but a prediction on how it will vary in the following day, taking into account the news of the day.

As we said in this chapter's introduction, we decided to use a second dataset from OpenBlender, since we want to include some sort of news feed to our ML process. There are many datasets containing news in OpenBlender's portal. From news in general, to news regarding any specific topic, mix of a handful of selected different topics, or news from a specific paper.

We wanted to use news that we thought had some sort of relationship with cryptocurrencies' price, but we didn't want to limit ourselves to one specific asset or company. We decided therefore to use all news that fell under the "business" category. We were inclined to use "Fox News Business Headlines" [50].

This dataset contains a "Timestamp" value, which will be used as the key to merge it with our Bitcoin's price dataset. It also contains two other useful columns, "Headline", which contains the headlines of a specific news page, and "Title" which indicates the titles of such news. We can see an example of a couple of rows of this dataset in Figure 6.2.

TIMESTAMP	HEADLINE	TITLE
2020/03/24 18:05:32	Simple Trading Director of Operations Danielle Shay, Fairfax Global Markets CEO Paul Dietrich and "Barron's Roundtable" host Jack Otter discuss the possibility of working at home amid rising coronavirus concerns	Productivity threatened as telecommuting challenges hinder workers
2020/03/24 18:05:27	Citymeals on Wheels executive director Beth Shapiro discusses how her organization is helping feed the elderly during the coronavirus outbreak	Coronavirus cripples volunteering in America
2020/03/24 18:05:20	Association of Nurse Practitioners president Sophia Turner says patients are stealing and hoarding supplies in hospitals in the midst of the coronavirus pandemic, causing health care providers to improvise on how to protect themselves	Volunteers sew coronavirus masks for health workers amid shortage
2020/03/24 18:05:15	FOX Business' Lauren Simonetti breaks down streaming specials to watch over the weekend	YouTube reduces video quality to minimize lag

Figure 6.2: OpenBlender's Fox News Dataset Head Rows

As we will see in Section 6.4, in order to have a simpler, and more manageable dataset, we decided not to use this dataset as it is shown in Figure 6.2. But to pre-process it, to twist it a little bit, and, with its information "create a new dataset". This newly formatted dataset is based on the number of times certain words appear in Fox's business headlines. As it is shown in Figure 6.3, we have 4995 Columns, each containing the count of the number of times a certain word appear in that day's headlines throughout the whole Fox Business News dataset.

	fake	revolutionary	economies	contempt	property	business grady	unexpected	tesla	radio host	agree	...	material	idea	correspondent	political	propaganda
1	1	0	0	0	0	0	1	2	0	0	...	0	0	1	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	4	0	0
3	0	0	0	0	0	0	0	1	0	1	...	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
5	0	0	0	0	0	3	0	2	0	0	...	0	0	1	4	0

5 rows × 4995 columns

Figure 6.3: Word Count OpenBlender's Fox News Dataset Head Rows

6.2 Pipeline

In Figure 6.4 we can see the pipeline of the module built for this chapter.

As we explained in Section 6.4, we are going to extract data from two different data-sets, after extracting the data, we preprocess it to alter its parameters, and then we will merge both into just one "blended" dataset that we will use to feed our ML model:

1. The first dataset is OpenBlender's Bitcoin's dataset that we saw in Figure 6.1. Containing information on Bitcoin's price and price fluctuations.

2. The second dataset is Fox's Business News, as seen in Figure 6.2. It contains the business news of Fox for each day.

We then use this data to train our model using two different classifiers: Random Forest Classifier and Decision Tree Classifier. After training our model, we will try to predict if Bitcoin's price will rise or fall based on the cumulative information of news and Bitcoin prices from previous days.

Last, but not least, we will try to ascertain that the ML model predicted correctly, by getting the precision and accuracy of the predictions, as well as the model's confusion matrix.

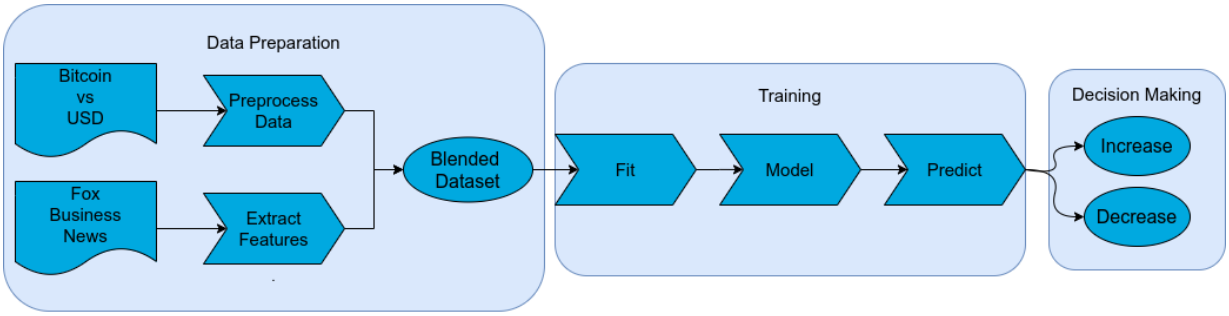


Figure 6.4: Include News Pipeline

As we said before this prediction will be done to check the possibility of predicting Bitcoin's price using counts of the number of appearances of specific words from headlines of business news, to see if it could be interesting, and worth the effort, to include it as a new input parameter for our RL agent, if we are able to get a relationship between the price increase / decrease of Bitcoins and the business news Fox publishes.

6.3 Get Bitcoin Data and Preprocess

First, by creating a free account in OpenBlender [48] we called their API on our selected Bitcoin's price dataset [49] that can be seen in Figure 6.1.

Then, since what we are interested in, is knowing whether the price will fall or rise, we created a new column with a binary value. Based on the value on the column "change" we create a new column called "change_over_0" with value "1" if "change" is greater than zero, and value "0" otherwise. We can see the commands executed and the outcome in Figure 6.5.

```
df2['change_over_0'] = df2['change']
df2.loc[~df2['change_over_0'].str.contains('-', na=False), 'change_over_0'] = 1
df2.loc[df2['change_over_0'].str.contains('-', na=False), 'change_over_0'] = 0
df2.head()
```

	volume	timestamp	price	high	low	open	change	change_over_0
0	31.08K	1584723600	6205.8	6897.0	5687.6	6173.5	0.52%	1
1	21.13K	1584550800	5416.2	5452.7	5033.9	5339.0	1.33%	1
2	41.78K	1584378000	5058.5	5381.9	4468.0	5381.9	-6.03%	0
3	18.64K	1584205200	5218.2	5673.2	5099.0	5621.8	-7.24%	0
4	116.68K	1584032400	4927.0	8001.7	4612.1	7975.3	-38.18%	0

Figure 6.5: News Add Change Over 0 Attribute

Sice we can't use information from a given day to predict the value of the same day, we need to add a 1-day lag to our "change_over_0" column. That way we will have information on whether the price rose or fell the day before. We achieve this by shifting our column by one position. And since we don't want incomplete rows, we removed the first and last row. This way, as it can be seen in Figure 6.6, we have our dataset with a new row containing "1" if the value rose the previous day or "0" if it fell.

```
df2.change_over_0 = df2.change_over_0.shift(1)
df2.drop(df2.tail(1).index,inplace=True)
df2.drop(df2.head(1).index,inplace=True)
df2 = df2.dropna()
df2
```

	volume	timestamp	price	high	low	open	change	change_over_0
1	21.13K	1584550800	5416.2	5452.7	5033.9	5339.0	1.33%	1.0
2	41.78K	1584378000	5058.5	5381.9	4468.0	5381.9	-6.03%	1.0
3	18.64K	1584205200	5218.2	5673.2	5099.0	5621.8	-7.24%	0.0
4	116.68K	1584032400	4927.0	8001.7	4612.1	7975.3	-38.18%	0.0

Figure 6.6: News Change Over 0 Shifted

6.4 Include Fox News

One useful feature offered by OpenBlender is that you can Blend two different data-sets. As we stated in Section 6.1 we decided to use a dataset that consists of the Business News from Fox [50], and blend it with our Bitcoin-USD dataset [49].

Since we thought, adding some Natural Languaje Processing (NLP) would be to complicate things too much, we wanted to turn Fox's Business News' headlines into a numerical feature. We did this by creating n-grams (with n=1 and n=2) and count the number of times certain words appeared in them.

For this task, OpenBlender also offers the chance of creating text vectorizers. Since their API call to create a text vectorizer wasn't working for some reason, we created it manually from their website, but it basically consisted of the features seen in Figure 6.7.

```
vectorizer_parameters = {
    'token' : 'MyOpenBlenderToken',
    'name' : 'FoxBusinessTextVectorizer',
    'sources':[{'id_dataset' : 'BusinessFoxNewsDatasetId',
                'features' : ['headline', 'title']}],
    'ngram_range' : {'min' : 1, 'max' : 2},
    'language' : 'en',
    'remove_stop_words' : 'on',
    'min_count_limit' : 2
}
```

Figure 6.7: News Text Vectorizer

As we see in Figure 6.8, we used the generated Text Vectorizer to retrieve the News-Bitcoin blended data with the n-grams that we will use to predict the price. We set the restriction as "Predictive" because we don't want to use any future information. We also selected a time interval of 12h, this way it will try to predict taking into account the news from the past 12 hours. Last but not least, we selected as a start date the first date in the Fox news dataset (08-20-2019), and, as the end date, the date when we tried this (03-21-2020).

```
parametersNews = {
   'token':'MyOpenBlenderToken',
   'id_dataset':'BitcoinDatasetId',
   'blends':[{'id_blend':'TextVectorizerId',
              'blend_type' : 'text_ts',
              'restriction' : 'predictive',
              'blend_class' : 'closest_observation',
              'specifications':{'time_interval_size' : 3600*12 }}],
   'date_filter':{'start_date':'2019-08-20T16:59:35.825Z',
                  'end_date':'2020-03-21T17:59:35.825Z'},
   'drop_non_numeric' : 1
}
```

Figure 6.8: News and Bitcoin Blended Data Parameters

6.5 Preprocess Blended data and Train model

Once we have all the information we need. We need to clean our dataset and do the preprocessing we saw in Section 6.3. First we decided to change the type of the column "Volume" from string to float. Then, we add our new column with the binary value on whether the price rose or fell. And last, we removed the two columns that were text from our dataset.

```
df["volume"]= df["volume"].str.replace("K", "", case = False)
df["volume"] = df.volume.astype(float)
df["volume"] = df["volume"] * 1000
df['change_over_0'] = df['change']
df.loc[~df['change_over_0'].str.contains('-', na=False), 'change_over_0'] = 1
df.loc[df['change_over_0'].str.contains('-', na=False), 'change_over_0'] = 0
df.change_over_0 = df.change_over_0.shift(1)
df.drop(df.tail(1).index,inplace=True)
df.drop(df.head(1).index,inplace=True)
df = df.dropna()
del df['change']
del df['5e7821d395162926d7d2565d_source']
df.columns = [str(col).replace("5e7821d395162926d7d2565d_", "") for col in df.columns]
df.head()
```

perty	business grady	unexpected	tesla	radio host	agree	...	material	idea	correspondent	political	propaganda	police officer	resources	oil	incredible	change_over_0
0	0	1	2	0	0	...	0	0	1	0	0	0	0	7	0	1.0
0	0	0	0	0	0	...	0	0	4	0	0	0	0	0	0	1.0
0	0	0	1	0	1	...	0	0	1	0	0	0	0	0	0	0.0
0	0	0	0	0	0	...	0	0	0	0	0	0	0	6	0	0.0
0	3	0	2	0	0	...	0	0	1	4	0	0	0	0	0	0.0

Figure 6.9: News Final Dataset

After having our final dataset, we decided to separate our data into "train" and "test". As we can see in Figure 6.10, we have 77 observations to train and 31 to test with.

```
X = df.loc[:, df.columns != 'change_over_0'].values
y = df.loc[:,['change_over_0']].values
div = int(round(len(X) * 0.29))
# We take the first observations as test and the last as train because the dataset is ordered by timestamp descendi
X_test = X[:div]
y_test = y[:div]
print(X_test.shape)
print(y_test.shape)
X_train = X[div:]
y_train = y[div:]
print(X_train.shape)
print(y_train.shape)
```

```
(31, 4994)
(31, 1)
(77, 4994)
(77, 1)
```

Figure 6.10: News Train and Test Sets

We decided to use "RandomForestRegressor" and "DecisionTreeClassifier" to predict the price. After fitting our RandomForest and DecisionTree, and calling the predict function, our ML model predicts a number between 0 and 1. Since our expected prediction is a binary number we rounded our predictions assuming that the price increases if it's higher than 0.5 and decreases if it's lower than 0.5.

Lastly, for a better understanding of our results, we printed the Area Under The Curve (AUC), the confusion matrix and the accuracy score. All of this can be seen in Figures 6.11 and 6.12.

```
rf = RandomForestRegressor(n_estimators = 1000)
rf.fit(X_train, y_train.ravel())
y_pred = rf.predict(X_test)

threshold = 0.5
preds = [1 if val > threshold else 0 for val in df_res['y_pred']]

print(roc_auc_score(preds, df_res['y_test']))
print(metrics.confusion_matrix(preds, df_res['y_test']))
print(accuracy_score(preds, df_res['y_test']))
```

```
0.4212962962962963
[[16 11]
 [ 3  1]]
0.5483870967741935
```

Figure 6.11: News Results Random Forest

```
max_depth=3
random_state=1

# Create decision tree model
model = tree.DecisionTreeClassifier(max_depth=max_depth, random_state=random_state)

# Train the model using the training sets
model.fit(X_train, y_train.ravel())
y_pred = rf.predict(X_test)

threshold = 0.5
preds = [1 if val > threshold else 0 for val in df_res['y_pred']]

print(roc_auc_score(preds, df_res['y_test']))
print(metrics.confusion_matrix(preds, df_res['y_test']))
print(accuracy_score(preds, df_res['y_test']))
```

```
0.2931034482758621
[[17 12]
 [ 2  0]]
0.5483870967741935
```

Figure 6.12: News Results Decision Tree

6.6 Results and Conclusions

Results

The AUC represents the degree or measure of separability. It tells how much, a model is capable of distinguishing between two values. The higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

As we can see in Figures 6.11 and 6.12 the obtained ROC AUC value is not very high. We obtained a value of 0.42 for when using Random Forest algorithm and 0.29 when using Decision Tree algorithm. This is the first flag telling us that our ML model is not very capable of distinguishing between whether the price will rise or fall.

There are four different possible outcomes when making a binary prediction such as the one we are trying to make (If the price change will rise or fall):

- If the outcome is that the price rose and we predicted that it would, we consider it a **True Positive (TP)**, since our prediction was true and it was a positive outcome.
- If the outcome is that the price fell and we predicted that it would rise, we consider it as a **False Positive (FP)**, since our prediction was false but it was a positive outcome.
- If the outcome is that the price rose, but our prediction was that it would fall, we consider it as a **False Negative (FN)**, since our prediction was wrong and it was a negative outcome.
- If the outcome is that the price fell, and indeed, our prediction was that it would do so, we consider it a **True Negative (TN)**, since we were right and it was a negative outcome.

The confusion matrix shows all the TPs, TNs, FPs and FNs as it can be seen in Table 6.1

		Predicted Values	
		Positive	Negative
True Values	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Table 6.1: Confusion Matrix

And of course, the last value we can use to measure our ML model's success or failure is the accuracy of its predictions, this is, the percentage of times it made a correct prediction.

Random Forest

As we saw in Figure 6.11, for the RandomForest approach, we got **54.83%** of the predictions correct with a **0.42** AUC.

		Predicted Values	
		Positive	Negative
True Values	Positive	16	11
	Negative	3	1

Table 6.2: Random Forest Confusion Matrix

From its confusion matrix shown in Table 6.2 we can see that we got the following results:

- 16 times we predicted a decrease and it decreased.
- 11 times we predicted a decrease and it increased.
- 3 times we predicted an increase and it decreased.
- 1 time we predicted an increase and it increased.

This results are obviously not good enough since it is just a little bit better than using a coin toss to predict.

Decision Tree

On the other hand, as we saw in Figure 6.12, for the Decision Tree approach, we got an accuracy of **54.84%**, meaning that only 54% of the predictions made were correct. We also obtained a **0.29** value for the AUC.

		Predicted Values	
		Positive	Negative
True Values	Positive	17	12
	Negative	2	0

Table 6.3: Decision Tree Confusion Matrix

From the confusion matrix shown in Table 6.3 we can see that we got the following results:

- 17 times we predicted a decrease and it decreased.
- 12 times we predicted a decrease and it increased.
- 2 times we predicted an increase and it decreased.
- 0 time we predicted an increase and it increased.

This obtained results are also not good enough to be considered when trying to trade Bitcoins.

Conclusion

We can conclude that, even though some level of accuracy was accomplished, since it is higher than 50%, this results are not good enough for us to use as it is right now. We decided that it was not worth the effort to try to include the information, as we have it now, into our RL agent.

This doesn't mean that the work done in this chapter is useless, far from it, this has the potential to become something that would improve our RL agent's performance. There are always ways to improve the results obtained in this chapter. We could tune it a little bit to be able to use it successfully in our project. We will see in Section 7.4 some things we might be able to do to make this kind of data potentially more useful to us.

CHAPTER 7

Conclusions and Future Work

This chapter will describe the achieved goals done by the master thesis following some the key points developed in the project.

We will extract some conclusions, and analyze the problems that we faced during the development of this thesis.

We will also go over the goals that were set at the beginning of this project and that were achieved by the end of it. As well as the ones that could not be accomplished, if any.

Last but not least, we will talk about related future work. What else could be done, taking this project as a base, and what could be achieved.

7.1 Conclusion

This project has fulfilled the proposed objectives. This thesis's main goal was to create an Algorithmic Trading System for Cryptocurrencies using Reinforcement Learning (RL) and Deep Learning (DL).

In addition to this main goal, we didn't want to settle with just an agent capable of making trading decisions but to have an agent that make good decisions. furthermore, we wanted an agent that made better decisions than other well-known trading strategies. So we compared our RL agent's rewards and results with the performance of two other well-known strategies: A Momentum strategy, and a Buy-and-Hold strategy.

We studied related work to see how other people tried to accomplish tasks that were similar to ours, from just predicting the price the cryptocurrency will have, to also creating RL agents.

We selected the dataset that we though fit better our requirements, we cleaned it, preprocessed it, and trained and tested our model with it.

Finally, as we have seen in Chapter 5 we got the desired results with our RL agent obtaining not only some rewards, but more than the two other strategies we compared it with.

7.2 Achieved Goals

During the development of this project we achieved various features and goals. The most important ones are described bellow.

Build a pipeline for cryptocurrency prediction . All our data needed to be well preprocessed in order to train and test our model correctly. We created a pipeline to accomplish this task, which cleaned our dataset from unnecessary data and collected all the key features, and, later, fed our model.

Vanquish all our targets. Our objectives were fully achieved. We created a RL agent capable of trading with cryptocurrencies on its own. And as we saw in Chapter 5 our RL agent surpassed the other two strategies both in rewards obtained by trading and in the MDD.

Exceed our projects initial scope . We added a last chapter (Chapter 6) where we tried to predict Bitcoin's price variation using business news to add information to our agent to add value to its performance.

7.3 Problems Faced

During the development of this project we faced some problems. Next we will note and give a brief description to the most significant ones.

Learning About ML, RL and Trading. Maybe the most obvious one would also be the hardest one. Or, at least, the most time-consuming one. In order to correctly develop this Master's Thesis, first I had to learn about the technologies needed, as well as about the trading market, trading strategies, etc. Of which I knew very little, or nothing about, and of which I have never heard anything from in class.

Overfitting. In statistics, overfitting is "The production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". [96] After doing some research and including a lot of parameters, we discovered that the obtained results were not as good as expected. Later we realized that, in order for us to avoid having an overfitted model we had to remove some parameters, so our model didn't have more parameters than can be justified by the data.

Time of Computation. During the development of this project, the only hardware available was a pretty old laptop. Due to this, the time that it took for our agent to train and test was really high. This delayed our projects time schedule.

Library versions. For the development of this project, I scoped various previous projects, most of which were developed several years ago and the code they provided only worked with previous library versions. For which I had to install several past versions of the different libraries I wanted to try out, and most of them had several compatibility issues.

Bitcoin's price drop. Up until 2018 bitcoin's price went periodically up, and after 2018, it experience a dramatically big drop. As I stated before, most of the related work was done before said date and, therefore, the predicted results got an easier better score than my initial ones.

Time-Sensitive. One of this project's main concerns was to make sure that it never used future data to predict past data, since it would defeat the purpose of having to predict nothing to begin with. If you already know what the bitcoin's price will be in the future you don't need to predict anything. So I had to be very careful not to allow the agent to know any future data (like when using k-fold for training purposes).

7.4 Future Work

In this section we will explain the possible improvements or new features that could be added to the project.

Use News Feeds. As we saw in Chapter 6, it is possible to predict variations in Bitcoin's price. We tried doing this with a dataset containing Business News from Fox. And we only used n-grams (with n=1 and n=2) from it. The issue with this dataset was that it only contained news from october 2019. And since then, Bitcoin's price has mostly fallen, thus, having our RandomForest and DecisionTree classifiers predict almost all the time that the value would decrease.

We could try to find any other dataset with news form previous times. As well as try using not just n-grams but add NLP to analyze full sentences and let our RL agent decide which news are relevant and which are not.

Use stock value from different markets and products. Another piece of information that might be useful, would be to include data of the price of other products as well as value of different companies or markets. Just because we don't think Bitcoin's value is related to any of it, it doesn't mean that our RL agent wouldn't be able to come up with some kind of relationship between them.

Create an application. Another thing we could do to both, see our results better, and interact with our agent is to create an application, it could either be a web application or one for Android or iOS.

Depending on the amount of work that we want to put into it, we could use the application to show how the RL agent is doing in a real-time, day by day basis. Add any specific news we see fit. And even see what the agent recommends doing and either let it do what it wants, or even force the agent to take any other specific action we might want it to.

Compare it to other trading strategies. We could compare our RL agent with other known trading strategies that are commonly used and that have proven to obtain good results. It would of course have to be trading strategies that require little to no human interaction at all. Trading strategies based on a set of preset rules that the agent would have to follow.

Add decisions made from other tragind strategies into our RL agent. Lastly, we could even try to use the outcome obtained from any other trading strategies such as the Buy and Hold or Momentum strategies, (or any other ones from the previous bullet-point) as another input for our RL agent. It may be able to figure out the best prediction among all of them and even figure out when and why does each strategy succeed or fail.

APPENDIX A

Impact of this project

Our project has developed an automatic system capable of trading bitcoins on its own and gain money with it.

In this appendix we will reflect, in a qualitatively way, about the possible impact that this project may have and its repercussions. Weather they are positive or negative, direct or indirect, actual or future.

We will divide it into four sub-chapters and discuss them from a social, economic, environmental and ethical stand point.

We will also discuss the responsibilities that come along with the development of this Master's Thesis, the possible ethical and professional implications of it.

A.1 Social Impact

Due to the nature of this project, it is bound to have a strong social impact. As we have explained in Chapter 7, the ultimate goal of this project would be to create an application (Web or Mobile) that would contain this thesis' project, and that would allow the users to view or even interact with the systems' decisions. In particular, the goal is to improve the trading decisions of the users or even to generate profit on the background without needing any human interaction whatsoever.

As we have explained throughout this thesis, algorithmic trading is a reality nowadays and its being used for trading in many different fields by many hedge funds. This project proved the viability of using it also with cryptocurrencies in general and Bitcoin in particular.

Trading is usually seen as something inaccessible by most people, due to the fact that unless you have a very good understanding of how the market in which you want to invest, you are risking loosing your money by making bad transactions or decisions. This tool, while not necessarily used as a 100% accurate predicting tool (an oracle of some sort), might actually help people know in which direction Bitcoin's price is more likely to go, thus making it easier and more accessible for everyone to enter the trading market.

Even for more experience stock brokers or even hedge funds, that are more used to trade and have more knowledge of Bitcoin's price changing patterns, this might be a useful tool. Nowadays, a lot of money is made in "fast trading" or "high speed trading", which consists of ultrafast trading, by using computers with algorithms to execute trades within milliseconds of market changes. Our system, since it is based in RL is a hands-off system that doesn't need human interaction and can therefore make transactions in such little time. So, if we manage to improve even further this system, by implementing the ideas seen in Chapters 6 and 7, traders, brokers or hedge funds could give it real money to trade and gain profit competing with other "fast trading" strategies.

All of this would have a strong impact in today's society and the trading system as we now know it, for both accessibility and speed.

Last, but not least, we should consider the case where many people use our system. If enough people use our system to make trading decisions, based on what our system predicts the price of Bitcoin will be, it could even turn the tables and itself become the source and reason for Bitcoin price changes. This meaning, that if a lot of people use our system, and it suddenly predicts a drop of Bitcoin's price, many people will suddenly want to sell all their Bitcoins, resulting in a price drop of Bitcoins, therefore having our system actually been the cause of this drop itself.

A.2 Economic Impact

Certainly, given this project's nature and goals, it also has a huge economic impact, since the main objective and the project's usefulness and purpose is to make money from trading Bitcoins in real time.

The use of this system could result in a pretty big improvement in the benefits obtained from trading, thus having a big impact in our users' economies.

Also, if enough people use our system, as we have seen in the previous section, our system might even create a big impact in the price of Bitcoin itself and even in today's economy. Therefore, we should keep a close eye on its performance, number of users, and predictions over time.

Last, but not least, obviously we could monetize our system by selling it or charging users to use it, and have an economic input for ourselves.

A.3 Environmental Impact

To successfully develop this project, we need the hardware specified in Chapter 7. We will need a computer and/or server to deploy our system to make it accessible for our users.

Computers and other Information Technology (IT) infrastructures consume significant amounts of electricity. To this consumption, we must also add the energy spent in the cooling systems.

However, this environmental impact can be lowered if we use some sort of cloud computing to store and run our system. Since they are held in facilities that are already properly prepared to optimize all of this consumption. Some of which also use renewable energies and/or reuse the energy for other means (Such as using the heat produce by the computing processes in all these computers to heat some other facilities or buildings).

A.4 Ethical and Professional Implications

The first ethical problem, would be that this systems can fail, it can predict that the price will do a certain thing, and then be wrong. If of people trusted this system's prediction and it is wrong, many people could loose a lot money. That's why it is important to clarify that this system is not perfect and it shouldn't be taken as something 100% accurate.

Another ethical problem might be the fact that due to this type of systems, and the easy usage for people that want to trade, less people will require the services of traders, stock brokers and hedge funds, leaving this businesses with less customers and therefore, less money.

APPENDIX B

Budget for this Project

In this appendix we will reflect, and give a detailed description of the budget set for the development of this project, as well as the necessary budget to implement the ideas explained in Chapter 7.

This budget should cover all expenses such as material resources, licenses, taxes and human resources.

B.1 Technological Resources

The technological resources needed for a correct development of this project can be divided into two groups:

- **Software:** This part would include the software licenses needed for the development of our system. Nonetheless, we created our system using Open Source Software (OSS), therefore, there is no need to pay for any of the technologies used. The only one technology that has a "freemium" policy would be OpenBlender, used in Chapter 6.

- **Hardware:** This would include the hardware necessary for the development of our system. The hardware used has been mainly a laptop with the following characteristics:

  - Ram: 16GB
  - CPU: Intel Core i7, 2.5GHz x 4
  - Hard disk: 500GB

  The estimated value of a laptop with said characteristics might be around **$900**.

Amortization has not been taken into account for this calculations.

B.2 Licenses

As we have said in the previous section, since our software is OSS, there's no need to pay any licenses.

B.3 Human Resources

Considering the amount of work and hours put into this project, it would require one person working part-time during at least seven or eight months. This consideration takes into account also all the time spent in research and learning the necessary techniques, technologies and about the stock market and trading systems and strategies, as well as the state of the art and related projects previously done.

If we consider a part-time salary of a software developer in Spain of $800 per month, it would cost around **$6000**.

B.4 Taxes

As for right now, the system is not on sale, therefore, there's no need to pay taxes. However, we should take into account that if it is finally monetized in Spain it would be subject to a 15% tax on the product.

B.5 Computational Resources

As of today, the project is being stored and run in my own personal computer. If we decided to expand our business we would have to move it to a server, so we would have to consider the different prices that cloud computing platforms charge for their services.

Bibliography

[1] Open AI. Open ai. https://openai.com/about/. Online, Accessed 2020-Mar-16.

[2] Open AI. Open ai gym. https://gym.openai.com/. Online, Accessed 2020-Mar-16.

[3] @Bakkt. Tweet. https://twitter.com/Bakkt/status/1065241273009938432, 2018. Online; Accessed 2019-April-04.

[4] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[5] Richard Branson. Bitcoins in space. https://www.virgin.com/richard-branson/bitcoins-in-space, 2013. Online; Accessed 2019-February-13.

[6] Jason Brownlee. A gentle introduction to scikit-learn: A python machine learning library. https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/. Accessed: 2018-01-03.

[7] Dani Burger. Rise of robots: Inside the world's fastest growing hedge funds. https://www.bloomberg.com/news/articles/2017-06-20/rise-of-robots-inside-the-world-s-fastest-growing-hedge-funds. Online; Accessed 2019-Sept-17.

[8] CheapAir. https://www.cheapair.com/, 2018. Web Page; Accessed 2019-February-13.

[9] Coindesk. Bitcoin price. https://www.coindesk.com/price/bitcoin. Online, Accessed 2020-Mar-15.

[10] CoinMap. Map of bitcoin accepting venues. https://coinmap.org. Online; Accessed 2019-April-04.

[11] Lester Coleman. Bank of england reopens the possibility of bank-run cryptocurrency. https://www.ccn.com/bank-of-england-reopens-the-possibility-of-bank-run-cryptocurrency/, 2018. Online; Accessed 2019-April-04.

[12] Wikipedia contributors. Blockchain. https://en.wikipedia.org/w/index.php?title=Blockchain&oldid=933511862. Online, Accessed 2020-Jan-17.

[13] Wikipedia contributors. Unix time. https://en.wikipedia.org/w/index.php?title=Unix_time&oldid=898002806, 2019. Online; accessed 2019-June-5.

[14] Ignacio Corcuera-Platas. Development of a Deep Learning Based Sentiment Analysis and Evaluation Service. Master thesis, ETSI Telecomunicación, Madrid, January 2018.

[15] CryptoCompare. The ultimate api solution. https://min-api.cryptocompare.com/#. Online; Accessed 2019-May-27.

[16] CUDA. Cuda. https://developer.nvidia.com/cuda-zone. Repository: https://github.com/numpy/numpy. Online; Accesed 2019-April-04.

[17] DanB. Rectified linear units (relu) in deep learning. https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning. Online; Accessed 2019-May-20.

[18] Michael del Castillo. The 10 largest companies in the world are now exploring blockchain. https://www.forbes.com/sites/michaeldelcastillo/2018/06/06/the-10-largest-companies-exploring-blockchain/#729a20301343, 2018. Online; Accessed 2019-February-13.

[19] Michael del Castillo. Nasdaq is now working with 7 cryptocurrency exchanges. *Forbes, recovered from https://www.forbes.com/sites/michaeldelcastillo/2019/01/30/nasdaq-is-now-working-with-7-cryptocurrency-exchanges/#2aa62e5b2472*, 2019. Online; Accessed: 2019-February-13.

[20] Destinia. https://destinia.com/, 2018. Web Page; Accessed 2019-February-13.

[21] Facebook. Facebook. https://facebook.com. Online; Accesed 2019-April-04.

[22] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–37, 1996.

[23] Forbes. State of ai and machine learning in 2019. https://www.forbes.com/sites/louiscolumbus/2019/09/08/state-of-ai-and-machine-learning-in-2019/#51e3f5641a8d. Online; Accessed 2019-Sept-17.

[24] Forums.fast.ai. 98.7https://forums.fast.ai/t/98-7-accuracy-on-bitcoin-price-prediction/21943. Online; Accessed 2019-June-2.

[25] Google. Google. https://google.com. Online; Accessed 2019-April-04.

[26] GSI-UPM. Machine learning. https://github.com/gsi-upm/sitc/blob/master/ml1/2_5_0_Machine_Learning.ipynb, 2017. Accessed: 2019-January-03. Commit: 23073b3431529798f7ff9380126b212c17f5ec16.

[27] La Información. Eurocoinpay, la app española que permite pagar con criptomonedas en las tiendas. https://www.lainformacion.com/economia-negocios-y-finanzas/eurocoinpay-o-como-puedes-pagar-en-criptomonedas-en-tu-restaurante-favorito/6494972/. Accessed: 2018-January-03.

[28] Business Insider. The world's 2 billion unbanked, in 6 charts. https://www.businessinsider.com/the-worlds-unbanked-population-in-6-charts-2017-8/?r=AU&IR=T%2F. Online; Accessed 2019-April-04.

[29] CB Insights. 'we don't hire mbas': The new hedge fund winners will crunch the better data sets. https://www.cbinsights.com/research/algorithmic-hedge-fund-trading-winners/. Online; Accessed 2019-Sept-17.

[30] Jagreet Kaur Gill, XENONSTACK. Automatic log analysis using deep learning and ai for microservices. https://www.xenonstack.com/blog/log-analytics-deep-machine-learning/, 2018. Online; accessed 2019-April-29.

[31] Stefan Jansen. *Hands-On Machine Learning for Algorithmic Trading*. Packt Publishing Ltd, 2018.

[32] The Wall Street Journal. The quants run wall street now. https://www.wsj.com/articles/the-quants-run-wall-street-now-1495389108. Online; Accessed 2019-Sept-17.

[33] Littman M.L. Kaelbling, L.P. and A.W. Moore. Reinforcement learning: A survey. *Library of Congress Web Archives Collection*, 4:237–285, 1996.

[34] Kaggle. Bitcoin historical data. https://www.kaggle.com/mczielinski/bitcoin-historical-data/data, Mar 2019. Online; Accessed 2019-May-5.

[35] Keras. Keras: The python deep learning library. https://keras.io/. Online; Accessed 2019-June-2.

[36] Nikhil Ketkar. Introduction to pytorch. In *Deep learning with python*, pages 195–208. Springer, 2017.

[37] Adam King. Adam king rewards graph. https://miro.medium.com/max/2582/1*SFNha2nSRaeE100dTCIXLQ.png. Online, Accessed 2020-Mar-15.

[38] Adam King. Creating bitcoin trading bots don't lose money. https://towardsdatascience.com/creating-bitcoin-trading-bots-that-dont-lose-money-2e716 Online, Accessed 2020-Jan-21.

[39] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[40] Saijel Kishan. Robots are eating money managers' lunch. https://www.bloomberg.com/news/articles/2017-06-20/robots-are-eating-money-managers-lunch. Online; Accessed 2019-Sept-17.

[41] Huang (Steeve) Kung-Hsiang. Introduction to various reinforcement learning algorithms. part i (q-learning, sarsa, dqn, ddpg). https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dq Online, Accessed 2019-Dec-04.

[42] Danqing Liu. A practical guide to relu. https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7. Online; Accessed 2019-May-20.

[43] De Louvois. Lamborghini aventador sv roadster lp750-4. https://delouvois.com/browse/all_categories/2017-lamborghini-aventador-sv-roadster-lp750-4/, 2018. Online; Accessed 2019-February-13.

[44] Isaac Madan, Shaurya Saluja, and Aojia Zhao. Automated bitcoin trading via machine learning algorithms. *URL: http://cs229. stanford. edu/proj2014/Isaac% 20Madan*, 20, 2015.

[45] Iron Man. Jarvis. https://i.redd.it/tr6fr1bhylo01.jpg. Online; Accessed 2019-April-30.

[46] Nasdaq. Nasdaq: Daily stock market overview, data updates, reports & news. https://www.nasdaq.com/. Online; Accessed 2019-June-2.

[47] Numpy. Numpy. https://www.numpy.org/. Online; Accesed 2019-April-04.

[48] OpenBlender. Openblender. https://www.openblender.io/#/welcome. Online, Accessed 2020-Mar-21.

[49] OpenBlender. Openblender bitcoin dataset. https://www.openblender.io/#/dataset/explore/5d4c3af79516290b01c83f51/or/21. Online, Accessed 2020-Mar-21.

[50] OpenBlender. Openblender bitcoin dataset. https://www.openblender.io/#/dataset/explore/5d571f9e9516293a12ad4f6d/or/21. Online, Accessed 2020-Mar-21.

[51] Mike Orcutt. The top 12 cryptocurrencies and what they are—and aren't—good for. Technical report, 2018. MIT Technology Review, recovered from https://www.technologyreview.com/s/610835/the-top-12-cryptocurrencies-and-what-they-are-and-arent-good-for/. Online; Accessed: 2019-February-13.

[52] Packt. Packtpub. https://www.packtpub.com. Online; Accessed 2019-May-10.

[53] Jaime José Palos Pereira. Design and development of a system for detecting cyberbullying in twitter based on machine learning techniques. Graduate thesis, ETSI Telecomunicación, Madrid, January 2018.

[54] Pandas. Pandas. https://pandas.pydata.org/. Repository: https://github.com/pandas-dev/pandas. Online; Accesed 2019-April-04.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[56] Anthony Pompliano. Tweet from anthony pompliano. https://twitter.com/apompliano/status/1065052950823911424?lang=es. Online; Accessed 2019-April-04.

[57] Vincent Poon. Trading bitcoin with reinforcement learning, Sep 2017. Online; Accessed 2019-May-5.

[58] PyTorch. Pytorch. https://pytorch.org/. Repository: https://github.com/pytorch/pytorch. Online; Accesed 2019-April-04.

[59] PyTorch. Torch.optim. https://pytorch.org/docs/stable/optim.html. Online, Accessed 2019-Dec-11.

[60] Quantivity. Why log returns. https://quantivity.wordpress.com/2011/02/21/why-log-returns/. Online, Accessed 2019-Dec-04.

[61] Quantopian. Alphalens. https://github.com/quantopian/alphalens. Online; Accessed 2019-Nov-13.

[62] Quantopian. Alphalens. https://www.quantopian.com/posts/alphalens-a-new-tool-for-analyzing-alpha-factors. Online; Accessed 2019-Nov-13.

[63] Quantopian. Quantopian. https://www.quantopian.com/. Online; Accessed 2019-Oct-17.

[64] Quantopian. Zipline. https://www.zipline.io/. Online; Accessed 2019-Oct-17.

[65] Willi Richert. *Building Machine Learning Systems with Python*. Packt Publishing Ltd, 2013.

[66] Federico Riveroll. Predicting bitcoin price with news using python. https://medium.com/swlh/predicting-bitcoin-price-with-business-news-python-f3bcf60f5818. Online, Accessed 2020-Mar-21.

[67] Kate Rooney. 84% of companies are dabbling in blockchain, new survey says. *CNBC, recovered from https://www.cnbc.com/2018/08/27/84percent-of-companies-are-dabbling--in-blockchain-new-survey-says-.html*, 2018. Online; Accessed: 2019-February-13.

[68] Abhinav Sagar. Cryptocurrency-price-prediction. https://github.com/abhinavsagar/Cryptocurrency-Price-Prediction. Online, Accessed 2020-Mar-16.

[69] Abhinav Sagar. Cryptocurrency price prediction using deep learning. https://towardsdatascience.com/cryptocurrency-price-prediction-using-deep-learning-70cfca50dd3a. Online, Accessed 2020-Mar-15.

[70] Dennis Sahlstrom. Elon musk: Bitcoin is brilliant and crypto is way better than fiat money. https://toshitimes.com/elon-musk-bitcoin-is-brilliant/. Online; Accessed 2019-April-04.

[71] Banco Santander. Santander lanza en cuatro países el primer servicio de transferencias internacionales con blockchain. https://www.santander.com/csgs/Satellite/CFWCSancomQP01/es_ES/pdf/Santander_lanza_en_4_pa%C3%ADses_el_primer_servicio_de_transferencias_internacionales_con_blockchain_2_12042018.pdf, 2018. Online; Accessed 2019-February-13.

[72] Banco Santander. Santander lanza en españa las primeras operaciones de we.trade, una plataforma blockchain que facilita la internacionalización de empresas. https://www.santander.com/csgs/Satellite/CFWCSancomQP01/es_ES/pdf/

Santander_lanza_en_Espana_las_primeras_operaciones_de_we_trade_una_plataforma_blockchain_que_facilita_la_internacionalizaci%C3%B3n_de_empresas_03072018.pdf, 2018. Online; Accessed 2019-February-13.

[73] Scala. Scala. https://www.scala-lang.org/. Online; Accessed 2019-May-10.

[74] Rafael Schultze-Kraft. Don't be fooled - deceptive cryptocurrency price predictions using deep learning. https://hackernoon.com/dont-be-fooled-deceptive-cryptocurrency-price-predictions-using-deep-learning-bf27e Online; Accessed 2019-June-1.

[75] Rafael Schultze-Kraft. lstm-bitcoin-prediction. https://github.com/neocortex/lstm-bitcoin-prediction. Online; Accessed 2019-June-2.

[76] Sunith Shetty. Predicting bitcoin price from historical and live data. https://hub.packtpub.com/predicting-bitcoin-price-from-historical-and-live-data/. Online; Accessed 2019-May-10.

[77] Lionel Shriver. Why cryptocurrency is the answer. https://www.spectator.co.uk/2018/01/why-cryptocurrencies-are-the-answer/, 2018. Online; Accessed 2019-April-04.

[78] Thomas Simonini. An introduction to policy gradients with cartpole and doom. https://medium.freecodecamp.org/an-introduction-to-policy-gradients-with-cartpole-and-doom-495b5ef2207f. Online; Accessed 2019-April-30.

[79] songrotek. Reinforcement learning classic algorithm combing 1: Policy and value iteration. https://blog.csdn.net/songrotek/article/details/51378582. Online, Accessed 2020-Jan-30.

[80] Randall Stephens. The future of cryptocurrency: Why e-commerce is the answer. https://medium.com/swlh/the-future-of-cryptocurrency-why-e-commerce-is-the-answer-822e62ba12f5. Online; Accessed 2019-April-04.

[81] Flood Sung. Dnq from entry to abandon 4 dynamic programming and q-learning. https://zhuanlan.zhihu.com/p/21378532?refer=intelligentunit. Online, Accessed 2019-Dec-04.

[82] Expert System. What is machine learning? a definition. https://www.expertsystem.com/machine-learning-definition/. Online; Accessed 2019-April-30.

[83] Ta-Lib. Ta-lib. https://www.ta-lib.org/. Online; Accesed 2019-April-04.

[84] TensorFlow. Tensorflow. https://www.tensorflow.org/. Repository: https://github.com/tensorflow/tensorflow. Online; Accesed 2019-April-04.

[85] ThirstyScholar. Trading bitcoin with reinforcement learning. https://github.com/ThirstyScholar/trading-bitcoin-with-reinforcement-learning, 2018. Online; Accessed 2019-February-13.

[86] Financial Times. Blackrock bets on algorithms to beat the fund managers. https://www.ft.com/content/e689a67e-2911-11e8-b27e-cc62a39d57a0. Online; Accessed 2019-Sept-17.

[87] Financial Times. Computer-driven hedge funds join industry top performers. https://www.ft.com/content/9981c870-e79a-11e6-967b-c88452263daf. Online; Accessed 2019-Sept-17.

[88] Financial Times. Fintech: Search for a super-algo. https://www.ft.com/content/5eb91614-bee5-11e5-846f-79b0e3d20eaf. Online; Accessed 2019-Sept-17.

[89] Financial Times. Two sigma rapidly rises to top of quant hedge fund world. https://www.ft.com/content/dcf8077c-b823-11e7-9bfb-4a9c83ffa852. Online; Accessed 2019-Sept-17.

[90] Financial Times. When silicon valley came to wall street. https://www.ft.com/content/ba5dc7ca-b3ef-11e7-aa26-bb002965bce8. Online; Accessed 2019-Sept-17.

[91] Tweeter. Tweeter. https://twitter.com. Online; Accessed 2019-April-04.

[92] Wall-E. Wall-e and eve. https://images5.alphacoders.com/587/587128.jpg. Online; Accessed 2019-April-30.

[93] we.trade. https://we-trade.com/, 2018. Web Page; Accessed 2019-February-13.

[94] Wikipedia. Bitcoin. https://es.wikipedia.org/wiki/Bitcoin. Online, Accessed 2020-Mar-29.

[95] Wikipedia. Learning rate. https://en.wikipedia.org/wiki/Learning_rate. Online, Accessed 2019-Dec-11.

[96] Wikipedia. Overfitting. https://en.wikipedia.org/wiki/Overfitting. Online, Accessed 2020-Mar-19.

[97] Wikipedia. Q-learning algorithm. https://en.wikipedia.org/wiki/Q-learning. Online, Accessed 2019-Dec-04.

[98] Wikipedia. Stochastic gradient descent. https://en.wikipedia.org/wiki/Stochastic_gradient_descent. Online, Accessed 2019-Dec-11.

[99] Wikipedia contributors. Rectifier (neural networks). https://en.wikipedia.org/w/index.php?title=Rectifier_(neural_networks)&oldid=896458588. [Online; accessed 2019-May-23].

[100] Wikipedia contributors. Activation function. https://en.wikipedia.org/w/index.php?title=Activation_function&oldid=897708534, 2019. [Online; accessed 2019-May-23].

[101] Wikipedia contributors. Ai effect. https://en.wikipedia.org/w/index.php?title=AI_effect&oldid=894592033, 2019. Online; accessed 2019-April-29.

[102] Wikipedia contributors. Artificial intelligence. https://en.wikipedia.org/w/index.php?title=Artificial_intelligence&oldid=894539363, 2019. Online; accessed 2019-April-27.

[103] Wikipedia contributors. Backpropagation. https://en.wikipedia.org/w/index.php?title=Backpropagation&oldid=897855738, 2019. [Online; accessed 2019-May-23].

[104] Wikipedia contributors. Chain rule. https://en.wikipedia.org/w/index.php?title=Chain_rule&oldid=895640848, 2019. [Online; accessed 2019-May-23].

[105] Wikipedia contributors. Feedforward neural network. https://en.wikipedia.org/w/index.php?title=Feedforward_neural_network&oldid=891888163, 2019. [Online; accessed 2019-May-23].

[106] Wikipedia contributors. Gradient descent. https://en.wikipedia.org/w/index.php?title=Gradient_descent&oldid=897146259, 2019. [Online; accessed 2019-May-23].

[107] Wikipedia contributors. Greedy algorithm. https://en.wikipedia.org/w/index.php?title=Greedy_algorithm&oldid=889459823, 2019. [Online; accessed 2019-May-2].

[108] Wikipedia contributors. Long short-term memory. https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=898571612, 2019. [Online; accessed 2019-June-2].

[109] Wikipedia contributors. Multilayer perceptron. https://en.wikipedia.org/w/index.php?title=Multilayer_perceptron&oldid=895625879, 2019. [Online; accessed 2019-May-23].

[110] Wikipedia contributors. Numpy. https://en.wikipedia.org/w/index.php?title=NumPy&oldid=894973198, 2019. [Online; accessed 2019-May-5].

[111] Wikipedia contributors. Pytorch. https://en.wikipedia.org/w/index.php?title=PyTorch&oldid=895047672, 2019. [Online; accessed 2019-May-5].

[112] Wikipedia contributors. Reinforcement learning. https://en.wikipedia.org/w/index.php?title=Reinforcement_learning&oldid=893951726, 2019. [Online; accessed 2019-May-1].

[113] Wikipedia contributors. Technical analysis. https://en.wikipedia.org/w/index.php?title=Technical_analysis&oldid=886772262, 2019. Online; accessed 2019-April-28.

[114] Wikipedia contributors. Unsupervised learning — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Unsupervised_learning&oldid=891660794, 2019. [Online; accessed 2019-April-30].

[115] Math Works. What is deep learning? 3 things you need to know. https://www.mathworks.com/discovery/deep-learning.html. Online; Accessed 2019-April-30.

[116] Tristan Yates. Quants: The rocket scientists of wall street. https://www.investopedia.com/articles/financialcareers/08/quants-quantitative-analyst.asp. Online; Accessed 2019-Sept-17.