UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MASTER

DEVELOPMENT OF A TRADING ANALYSIS AND OPERATION SYSTEM FOR AMERICAN STOCKS USING DEEP REINFORCEMENT LEARNING

VÍCTOR CLEMENTE TERUEL

2020

TRABAJO DE FIN DE MASTER

Título:	Desarrollo de sistema de trading para acciones americanas utilizando aprendizaje profundo y por refuerzo
Título (inglés):	Development of a Trading analysis and operation System for American stocks using Deep Reinforcement Learning
Autor:	Víctor Clemente Teruel
Tutor:	Carlos Ángel Iglesias Fernández
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

Development of a Trading analysis and operation System for American stocks using Deep Reinforcement Learning

2020

Resumen

El trading es una actividad en la que operadores especializados tienen la habilidad de conseguir beneficios gracias a sus decisiones en el mercado. Este perfil ha tenido que estudiar y especializarse bastante para llegar a un punto en el que tengan mayor fiabilidad para predecir acciones del mercado. Mucha gente tiene la creencia de que generar beneficios mediante trading es algo trivial, pero la realidad es que el 98% de la gente que entra en este mundo pierde toda la inversión económica realizada. El mayor problema del perfil de un trader es todo el tiempo que lleva en encontrar buenas compañías u oportunidades de inversión. Para encontrar estas oportunidades, ellos realizan un análisis de cada empresa tanto en el ámbito fundamental como en el ámbito técnico. Hay que añadir que en los tiempos actuales, se ha disparado la utilización de algoritmos inteligentes o "Machine Learning". Muchas de las implementaciones van destinadas a mejorar la vida de las personas y a hacer más fáciles ciertas tareas, automatizándolas con ordenadores que replican o mejoran la actividad humana.

Con todo ello, el objetivo principal de este proyecto es lograr la creación de un algoritmo inteligente que pueda tomar conocimientos específicos de tendencias de mercado, aprendiendo mediante el uso de diferentes datos de precios recogidos de acciones americanas, todo ello utilizando aprendizaje profundo unido al aprendizaje por refuerzo. Además, se va a crear una herramienta de análisis visual para poder detectar y comparar las distintas tendencias de las acciones empresariales en ciertos momentos del mercado, correlándolas con el propio índice principal del mercado americano (S&P500). Este proyecto va a incluir un estado del arte de cada tecnología implementada, una descripción de las técnicas de trading aprendidas por el programa, una explicación de las premisas principales para obtener datasets específicos y así sacar mejor partido al entrenamiento realizado en los algoritmos, y una comparación final de modelos en función de la tendencia testeada (en tendencia alcista, bajista o lateral).

Palabras clave: Trading, aprendizaje profundo, aprendizaje por refuerzo, acciones, mercado, America

Abstract

Trading is an activity in which specialized people have the ability to create profit from their actions. They are people that study a lot to reach a point in which they can predict or suspect what is going to happen at a specific point in the stock market. Most of the people think that take profit from stock markets is trivial, but the reality is that 98% of people who try to reach the same point lost all their money. The biggest problem that traders have is that they need to spend hours and hours in front of a computer to search which companies are going to have a big tendency to change, in a short or in a long period of time, depending on which type of trading they operate, and then join this position before the tendency change appears. To find these opportunities, they can analyze the fundamental characteristics (cash flow, balance sheets...) and/or the technical chart (tendency direction, statistics...) of a company. Nowadays, there is a growth in machine learning developments and utilities, because of the improvement of this technology. Most of the machine learning applications are implemented to make our day-to-day life easier or to automatize some activities that computers can prepare better than humans.

The main objective of this project is to create a program that takes some of the knowledge that a specialized person can learn from courses, books or speeches, and train this program with different stocks datasets, to finally operate correctly in the American stocks market, all of it using deep learning and reinforcement learning. In addition, an analytic tool will be created to detect which tendency is the different stocks in a specific moment, comparing each other and with the main US index (S&P 500), to extract the datasets that will be provided to the program. This project is going to include a state of the art of each technology implemented, an description of which trading techniques are learned by the program, an explanation of which are the main premises to obtain train datasets, and the final results of how this intelligence operates in different market situations (bullish, bear or lateral trends).

Keywords: Trading, Deep Learning, Reinforcement Learning, Stocks, Market, America

Agradecimientos

Estos 6 años en esta escuela han marcado un antes y un después en mi vida. Entré con la moral bajo mínimos, cansado y sin apenas ganas de esforzarme por cualquier cosa, y en estos momentos siento que lo único que me apetece es comerme el mundo a bocados. Es por ello que, si tuviese posibilidad, agradecería a todas y cada una de las personas de las que he recibido un mínimo de ayuda durante estos años. En esta sección, voy a resaltar a esas personas más importantes durante este periodo.

Antes que nada, quiero agradecer a mi tutor Carlos Ángel por todo el seguimiento y ayuda recibida, por todas las correciones y consejos que me ha ido aportando durante estos meses, porque gracias a él puedo decir que he mejorado drásticamente en muchos aspectos básicos, y no tan básicos, y creo que puedo decir que he conseguido encontrar una materia en la que me siento cómodo, que me apasiona y me anima a desarrollarla más en profundidad para poder mejorar la vida de los demás con ella. Quiero meter también en este apartado a mi compañero y amigo Alejandro López, el cual me aconsejó a Carlos como tutor y fue el que me descubrió este mundo por primera vez, algo que fue un punto de inflexión de cara a este proyecto.

Cómo no voy a mencionar a mis padres y familiares en este proyecto, ese apoyo más cercano que se tiene cuando más lo necesitas y que nunca van a faltarte. Ellos han pasado por todos mis sufrimientos durante estos 6 años, viéndome estudiar y realizar proyectos hasta la saciedad, llorar por desesperación, sentir mis agbios en primera persona... Papá, mamá, Sergio (mi hermano): todo esto no hubiese llegado a buen puerto sin vosotros. Abuela, abuelo: se que me queda poco tiempo para disfrutar con vosotros. Por eso, os quiero dedicar esta memoria, y que os quede el recuerdo más bonito que se os pueda dejar. Gracias también a mis tios y primos por toda la ayuda que me han prestado durante este tiempo. Voy a estar ahí simepre para todo lo que necesitéis.

Estos 6 años me han hecho disfrutar de muchisima gente dentro de esta escuela. Quiero que sepáis que este párrafo va dedicado a todos vosotros. Por esas horas de estudio, por esos trabajos interminables, y por exámenes y evaluaciones inalcanzables, que como todo en esta

vida, se puede lograr. Os mencionaría a todos, pero creo que leyendo esto os vais a sentir identificados cada uno de vosotros que ha compartido un pedazito de vida conmigo, y que espero que sigamos compartiendo momentos más allá de estas paredes. Mención especial dentro de este mensaje para Mario, Carlos y David, que han sido con los que más tiempo he pasado realizando actividades dentro y fuera de la escuela, que desde que os ví de lejos el primer día sentía que podíais ser grandes personas para mi vida. Una pena no haberme cruzando antes con vosotros. A Carmen, Sandra, Irene, Jhoana, Andrés y Lourdes, que espero que acabe todo el problema actual del virus simplemente para poder disfrutar de vosotros en cualquier parte de Madrid, o del mundo. Se os quiere. Mencionar también a todo el grupo restante de chicos que formamos parte de la promoción del 2014, que tenemos esa gran piña que cada vez hemos podido disfrutar menos por tantas obligaciones. Aquí os espero para vivir días y experiencias inolvidables como las del inicio de la etapa universitaria.

No puedo olvidar a esa gente de fuera de la propia universidad que también me han marcado y ayudado. Esos amigos de toda la vida que han estado ahí para tomarme unas cañas con ellos, o simplemente para hablar de experiencias pasadas y futuras. Gracias Pablo, Iván, Manuel Artigas, Alejandro, Rubén, Alberto y Manuel Velázquez. También a esos amigos nuevos que se conocen sin esperártelo, y que al final se quedan como gran familia. Gracias Jorge, Fernando, Guillermo y Diego. Japón y el resto de mundo nos espera.

Y cómo me iba a olvidar de ti. Tú eres la única que sabes lo que me pasó el primer día de universidad que me hizo caer en un hoyo del que tuve poco tiempo para recuperarme. Tu sabes la adrenalina que me metiste dentro del cuerpo para reactivarme en tiempo récord, y la que me ha ido dando empujoncitos cada vez que me paraba en el camino. Tú me volviste a cambiar la vida justo otra vez en el momento más caótico de mi vida, y tu eres una de las personas que me ha seguido dando fuerzas para acabar lo mejor posible esta formación. Lo mejor de todo es que lo vas a seguir haciendo, y cada vez más fuerte. Por todo esto y muchas cosas más, gracias Soraya. Te tengo mucho que devolver.

A todo el resto de personas que hayan llegado hasta aquí leyendo esto, sólo espero que este proyecto sea de vuestro agrado y que os aporte conocimientos y visiones nuevas. Busco con ello despertar vuestro interés por este bonito mundo de la economía y la inversión, unido a ciertos algoritmos que pueden facilitarnos la vida un poco más. Nos vemos en este trepidante camino llamado vida. Un saludo a todos,

Víctor Clemente Teruel

Contents

Resumen **VII**

Abstract **IX**

Agradecimientos **XI**

Contents **XIII**

List of Figures **XVII**

List of Tables **XIX**

1 Introduction **1**

1.1 Context introduction 2
1.2 Problem specification 3
1.3 Project goals 4
1.4 Structure of this document 4

2 State of Art **7**

2.1 Trading Differentiation 8
2.2 Loss Recovery 9
2.3 Market Selection 10
2.4 Other Market Projects 12

3 Enabling Technologies **15**

3.1 Machine Learning Technologies 16
3.1.1 Machine Learning Types 16
3.1.2 Deep Learning Architectures 17
3.1.3 Reinforcement Learning 20
3.2 Machine Learning Libraries 22
3.2.1 Python 22
3.2.2 Numpy 23
3.2.3 Pandas 24

3.2.4 Plotly . . . 24
3.2.5 Chainer . . . 26
3.2.6 Kaggle . . . 26
3.2.7 Anaconda Navigator . . . 27
3.2.8 Jupyter . . . 28
3.3 Trading Tools . . . 29
3.3.1 Tradingview . . . 29
3.3.2 TA-Lib . . . 30

4 Deep-Learning Based Reinforcement Learning Model 33
4.1 Introduction . . . 34
4.2 Deep Neural Network Structure . . . 35
4.3 Environment class . . . 37
4.4 Training Process . . . 39
4.5 Testing Process . . . 41

5 Experimentation and Evaluation 43
5.1 Introduction . . . 44
5.2 Dataset Analysis . . . 44
5.2.1 Data Structure . . . 44
5.2.2 Price Selection . . . 45
5.3 Market Stocks Selection . . . 47
5.3.1 Theoretical Description & Dataset Selection . . . 47
5.3.2 Technical Company Analysis . . . 54
5.3.3 Data Preprocessing . . . 59
5.4 Neural Structure Optimization . . . 60
5.4.1 Model 0: Basis Structure . . . 61
5.4.2 Model 1: 1 Hidden Layer Optimization . . . 61
5.4.3 Model 2: 2 Hidden Layers Optimization . . . 64
5.4.4 Model 3: Optimization of M1 . . . 68
5.4.5 Model 4: Optimization of M2 . . . 73
5.4.6 Model Comparison . . . 75
5.5 Testing Analysis . . . 76
5.5.1 Bullish Trend . . . 77
5.5.2 Bearish Trend . . . 80
5.5.3 Lateral Trend . . . 83

6 Conclusions and Future Work **87**
6.1 Achieved Goals 88
6.2 Conclusions 89
6.3 Future lines of work 90

A Project Impact **91**
A.1 Introduction 92
A.2 Social Impact 92
A.3 Economic Impact 92
A.4 Environmental Impact 93
A.5 Ethical Impact 93

B Project Budget **95**
B.1 Physical Resources 96
B.2 Human Resources 96
B.3 Indirect Costs 97
B.4 Taxes 97

Bibliography **99**

List of Figures

2.1 General representation of trader activity [6] 8
2.2 Percentage needed to reach the breakeven point [8] 9
2.3 Comparison between most important indexes during last 25 years 11

3.1 CNN functionality example, analyzing an image [54] 18
3.2 RNN architecture example: Elman network [54] 19
3.3 DBN architecture example [54] 20
3.4 A NumPy structure [27] 23
3.5 Plotly has different chart types 25
3.6 Structure of the candlestick [46] 25
3.7 Kaggle model creation process [3] 27
3.8 A Jupyter notebook 28
3.9 An example of the Tradingview website [44] 30
3.10 Example of some analytics (MACD,RSI,STOCH...) that could be obtained mathematically from TA-Lib 31

4.1 Project Architecture 35
4.2 *Unified Modeling Language* (UML) Diagram 35
4.3 *Deep Neural Network* (DNN) Architecture 37
4.4 Environment structure 38
4.5 Reward process 39
4.6 Training process 40
4.7 Testing process 42

5.1 Example of an internal dataset structure 45
5.2 Apple stock prices comparison 46
5.3 Correlation table between Apple and S&P 500 prices 47
5.4 Apple test & train charts 48
5.5 Microsoft test & train charts 49
5.6 Coca-Cola test & train charts 49
5.7 Walmart test & train charts 50

5.8 JPMorgan test & train charts 51
5.9 Amazon test & train charts 51
5.10 Facebook test & train charts 52
5.11 AMD test & train charts 53
5.12 Procter & Gamble test & train charts 53
5.13 Chart of different stock prices, compared with the main country index (S&P500) 55
5.14 Correlation table of different shares with the S&P500 index 55
5.15 Correlation matrix of different shares with the S&P500 index 56
5.16 AMD trend analysis 57
5.17 Analysis of share trends 59
5.18 Reward Results of 1 Hidden Layer Models 63
5.19 Loss Results of 1 Hidden Layer Models 64
5.20 Reward Results of 2 Hidden Layers Models 66
5.21 Loss Results of 2 Hidden Layers Models 67
5.22 Reward Results of 2 Hidden Layers Models with different neuron sizes . . . 68
5.23 Loss Result obtained from 50&25-Neuron Hidden Layers Model 68
5.24 Sigmoid Hyperparameters 69
5.25 Linear Unit Function Hyperparameters 69
5.26 Softplus Hyperparameter 70
5.27 Tanh Hyperparameter 70
5.28 Softmax formula 70
5.29 Model 1 Hyperparater Optimization - Reward Charts 71
5.30 Model 1 Hyperparater Optimization - Loss Charts 72
5.31 Model 2 Hyperparater Optimization - Reward Charts 74
5.32 Model 2 Hyperparater Optimization - Reward Charts 75
5.33 General representation of a bulish trend [35] 77
5.34 Testing with Apple datasets 78
5.35 Testing with Facebook datasets 79
5.36 General representation of a bearish trend [31] 81
5.37 Bearish trend with Microsoft datasets 82
5.38 Bearish trend with Coca-Cola dataset 83
5.39 General representation of a lateral trend [11] 84
5.40 Lateral trend with *Advanced Micro Devices* (AMD) dataset 85

List of Tables

5.1 Dataset statistics 45
5.2 Samples Quantities 60
5.3 Market tests comparison *(R=reward, P=profit)* 76
5.4 Bullish testing comparison 80
5.5 Bullish testing comparison 83
5.6 Bullish testing comparison 84

B.1 Resource prices 98

CHAPTER 1

Introduction

This chapter describes the main objective of this final project, explaining the beginning points of how the idea was growing, and summarizing the structure of the research and development that involves it.

1.1 Context introduction

During all the mandatory teaching period in Spain, which covers 12 years until secondary school title is reached, economy topics are slightly learnt. Different subjects explain basic concepts of it, but also they do not go into detail about more complex components or knowledge. Furthermore, economy subjects are not imparted to all students: they are only taught by students who has select them during last year of secondary school and high school, and that students choose them with the goal of being enrolled in economic disciplines in their future.

Most people who had won millions of dollars during their life, that are also known as the most influential tycoons (for example Warren Buffet or Bill Gates) reach their fortune because of their economy knowledge [25], which is a basic feature that people must know to obtain better profits in their life. Market operation, money creation or interest types are an example of concepts that many people do not know. A famous consequence of how the economy works is, for example, when people get so much loans simultaneously without thinking what result will it have to their life, meanwhile they buy different objects that they can not afford, and probably this behavior could commit in financial problems in the future.

The stock market is frequently an unknown topic for most people. Each day, a lot of market news are visualized in television, but only a few people know its meaning. A stock market is a place which can be a real or a virtual one, and it allows to use money with the goal of obtaining shares of a company, maintaining them during a specific time period, to finally sell them in a better price and obtain a profit thanks to the operation. Markets are regularized automatically because of the supply and demand of the different stocks available. A market can be in a bullish, a bearish or a lateral trend, which depends on the price movement and the maximum and minimum reached.

There are two profile types that operate in stock markets: traders and investors. Their reach is to obtain profit from different operations, independently of what tendency the market has. They can operate alone, in a specific fund, or for an economy enterprise. Between traders and investors, the first group are the main example to this project, with the objective of simulate different trades (like them) and obtain profits from it.

1.2 Problem specification

Market operators follow a complex process which involves two primary analysis, in which they determine if a company stock is a competent candidate to invest:

- **Fundamental analysis:** it consists on an exhaustive analysis of the economic situation of the company. This analysis also covers all news that are related with the enterprise. To make a good fundamental analysis, it is necessary to underline three documents that each company makes public each quarter, which are used by the analysts to establish that economy situation. The documents are slightly described below:

  - *Income Statement:* it resumes all incomes and expenses of the company, including revenues, cost of goods, selling, general and administrative expenses. It also includes the earnings per share result, which reflects the profitability that a company has made during a quarter.
  - *Balance Sheet:* it explains which inventory form the assets and liabilities. It contains cash and short-term investments, accounts receivable, inventories, net properties and equipment, short-term and long-term debt, liabilities and taxes.
  - *Cash Flow Statement:* it consists in a detail resume of where is the money spent. There are the net operating cash flow, net investing cash flow, net financing cash flow and the free cash flow.

  It is important to specify that the publication of these documents is mandatory if a company is into a stock market. In addition, at the end of the year a final-year document is made, with a summary of all year results.

- **Technical analysis:** it covers all related to graph analysis, formulas, indicators like "supports" and "resistances", tendency channels, etc. The main goal is to predict the market behavior, trusting in a pattern replay by the chart, to discover entry points and exit points in a share. It depends on price and offer and demand volume.

Related with last explanations, underline that technical analysis keeps in mind, by share price, the economy situation of the company, all the news related with the company and any other valuation made by famous analysts, following the famous Dow theory [56] sentence which says: "The market discounts everything". The project objective is to take advantage of fundamental and technical analysis, reflected both of them in the stock price, to operate with different American stocks, making economy profit thanks to a deep learning intelligence which learns from a reinforcement learning environment.

1.3 Project goals

This project automatizes the market analysis using different machine learning techniques. All algorithm has a technical analysis basis, which is mandatory to carry out a development that reaches different goals during the process. The main objectives of this final project are described below:

- Develop a **deep learning intelligence** that operates in the American stock market and takes profit in a specific time period. To reach it, the architecture defines a **reinforcement learning** technique to evaluate the behavior of the algorithm, and then improve it during its learning. During the progress, it is necessary to define and apply different graphical analysis tools, in which stocks can be compared with other ones or analyzed individually, to finally determine which shares are selected from the dataset. An example of that tools are: charts with stock price and indicators, or a correlation matrix to compare stocks trends with country indexes.
- Carry out a **performance evaluation** using different shares from popular companies. This goal is reached after dataset selection and algorithm optimization. These two steps are realized by the study of different stock companies and hyperparameters utilized, choosing them from the dataset and development libraries respectively. After finishing these two investigation tasks, it is realized a behavior comparison between American shares with different market trends, adding their corresponding causes and conclusions about all the results obtained during the process.

1.4 Structure of this document

In this section, a brief overview of the chapters included in this document is provided. The structure is made as follows:

Chapter 2: State Of Art This chapter contains all the theoretical concepts utilized in this thesis.

Chapter 3: Enabling Technologies This point defines all the tools, programs, libraries and all related information needed during the development.

Chapter 4: Deep-Learning Based Reinforcement Learning Model This point is in charge of explaining all the the algorithm architecture and its behavior.

Chapter 5: Algorithm Training This chapter contains all the training carried out by the deep learning intelligence, analyzing the neural structure and the hyperparameters used.

Chapter 6: Deep Learning Testing This chapter analyzes the testing task with its final results and comparison between different market trends.

Chapter 7: Conclusions and Future Work This chapter exposes the final words about what is relevant in the project and which points are able to be improved.

CHAPTER 2

State of Art

This chapter covers all the theoretical explanations that are necessary to build the basis of this project, including trading concepts, market trend analysis and external approaches.

2.1 Trading Differentiation

Trading and investment ambit is growing during last years as other professional jobs [39]. This growth is caused by the facility to operate in markets (using only a computer or a smartphone), the high demand of online courses [45], and the renown taken by traders thanks to the social networks [13]. The main problem is that is common to confuse a trader with an investor, and they are different professional profiles. This project emphasizes better with the trader methodology, but it is important to differentiate them [32].

Figure 2.1: General representation of trader activity [6]

Principally, a trader is any people who exchanges (or trades) different stocks amounts. Their objective is to operate daily the market and make profit from it. It is important to emphasize in the idea of "any people", because it not only includes enterprise traders. Traders usually combine technical and fundamental analysis features to obtain a privileged point of view, even if they focus on the advantages of technical analysis, because they think that all info in contained into the charts. It follows the next sentence, contained into the Dow theory [56]: "The market discounts everything".

On the other hand, an investor is any person who buy shares of different companies, and make profit trusting on the large potential of them. Usually, this analysis used by them is mainly based in a future growth of the company, after studied the fundamental analysis of the company by their economic documents, and they do not focus on the technical analysis.

They expect to finish their positions in months, or in the next years.

Traders have the advantage of staying current in the companies that they invested, but with the main disadvantage of spending a lot of time looking different stocks, comparing it with others, and analyzing their technical situation. Instead of investors, they have more maneuverability of cutting losses than investors. The project algorithm is more related with trading methodology, making short-term decisions and following the market too.

2.2 Loss Recovery

As previously discussed, one of the most important goals of traders is avoiding high losses and keeping an eye in bad decisions. The effort needed to reach the losses breakeven point, that means the moment when the total losses are recovered, increases as the same time as the losses grow. The next image reflects this situation, showing the effort needed to recover losses generated:

Percent Loss	Percent Gain	Percent Loss	Percent Gain
5%	5.3%	55%	122.2%
10%	11.1%	60%	150.0%
15%	17.6%	65%	185.7%
20%	25.0%	70%	233.3%
25%	33.3%	75%	300.0%
30%	42.9%	80%	400.0%
35%	53.8%	85%	566.7%
40%	66.7%	90%	900.0%
45%	81.8%	95%	1900.0%
50%	100.0%		

Figure 2.2: Percentage needed to reach the breakeven point [8]

This project is focused on developing an accurate deep learning intelligence, minimizing the loss percentage as much as possible. This point is considered in future optimization analysis, specially in the neural and hyperparameter optimization cases, when different possibilities are exposed. In conclusion, this project puts accurate ahead of precision, making loss value more important than other parameters.

2.3 Market Selection

One of the most important steps taken in the project is the selection of the most appropriate index or country in which the deep learning intelligence will operate. It is possible to find and download any index dataset. For example, Yahoo Finance has the option of searching any index and download their values in CSV format [21] [22].

During this section, different country indexes are compared. Probably, they are into the group of the most important indexes in the world:

- *DAX:* main Germany index.
- *FTSE 100:* main United Kingdom index, weighted by the 100 most important enterprises of this country.
- *CAC 40:* main France index, weighted by the 100 most important stocks of France.
- *IBEX 35:* main Spain index, which is weighted by the most important 35 s of Spain.
- *S&P 500:* main United States index, contains the 500 most important enterprises.
- *CSI 300:* main China index.
- *Nikkei 225:* main Japan index.

The main point to keep in mind during this section is that the project is in search of a market with not a high volatility. It is possible to train the intelligence in a market with high volatility, but it implies a very difficult process to optimize the algorithm. That is because lateral markets, which are the ones in which the volatility has higher values, produce more false signals to buy or sale shares [12], and the algorithm would have more probability of mistaking because of it.

Considering last theory, the next task is to analyze the tendency and volatility of every index emphasized previously, and compare each other to determine which is the best country to take. To carry out this task, all the index charts are shown in the next page [19], one next to the other, to facilitate the comparison process:

(a) CAC 40 (France)

(b) CSI 300 (China)

(c) DAX (Germany)

(d) FTSE 100 (United Kingdom)

(e) IBEX 35 (Spain)

(f) Nikkei

(g) S&P 500 (United States)

Figure 2.3: Comparison between most important indexes during last 25 years

As it can be seen in the last in last graph summary, there are found three indexes which have a lateral trend. These indexes are *CAC40*, *CSI 300* and *IBEX 35*, automatically rejected for this project. The other five indexes have a rising trend, but not during all time interval. *Nikkei* and *FTSE 100* have a bullish trend from 2012 and 2010 respectively, but *DAX* and *S&P 500* have a cleaner trend, with less corrections that the other ones. It has a strong impact in the task of decide the market in which the learning algorithm will learnt.

Between *DAX* and *S&P 500*, the American one was selected to collaborate as a dataset in this project with their company shares. That is because the rising trend that it has was more continuous than the *DAX* index, and it would cause a better learning results than using the European one. Either way, all countries could be used to apply the algorithm into them. But with the American market, the final results would be more positive than the other ones.

It should be noted that all indexes have a strong decline at the final of its tendency timeline, because this is the period of time in which COVID-19 (famous named as "coronavirus") brokes all markets trends. This issue was really painful in most of the stock markets in the world, but in this project this time period is not covered.

2.4 Other Market Projects

To finish the chapter, this memory covers different project examples that were found on internet and are related with trading ambit and machine learning techniques. These developments are created by individual traders, and are described below:

- The first example [14] is related with Forex, which is a type of market where different currencies are traded [58]. This project implements a deep learning algorithm that operates trading Great Britain Pounds to United State Dollars, and vice versa, creating a strategy that is finally profitable for real-time trading.

- Other development case [43] consists in beating the S&P500 index profit for 10 years. This goal is reached using Support Vector Machine, which is not a deep learning algorithm but it is an example of a profitable development. This example tries to have a stable revenue percentage, against the strategy of the deep learning intelligence developed and explained in this master final project.

- Finally, last example [4] implements machine learning algorithms like linear regression and decision trees with market datasets. The main difference from this example is

that it operates with one minute interval, instead of one day interval like this master project. This case utilizes several indicators like bid volume, ask volume and VWAP to reach money profit.

Last cases exemplifies that is possible to create economic profit if the strategy is correct. This project wants to follow these ideas, trying to maximize earnings thanks to deep learning and reinforcement learning techniques.

CHAPTER 3

Enabling Technologies

This chapter offers different definitions and explanations of machine learning techniques, main technologies utilized, libraries and tools that have made possible this development.

3.1 Machine Learning Technologies

3.1.1 Machine Learning Types

Firstly, it is important to specify a definition of what machine learning is [17]:

"Machine Learning (or automatic learning) is a scientific class that is part of Artificial Intelligence (also known as AI), and is capable to create systems that learn by itself."

These types of systems do not need a typical programming that is carried out by most of projects. They need to explain in detail the algorithm implemented and a dataset in which the intelligence program bases on to learn on its own. The main characteristic that machine leaning has is that it can build a mathematical model based on a training data, which is a fragment of the original dataset used [63]. Inside this machine learning world, there are different types of learning process that distinguish each machine:

- **Supervised learning** [1] is a technique that utilizes labeled datasets, which consists in a pair or vector objects that have a component with input data, and other component with the desired solution. The term appears from the idea of learning from a training dataset, following the same procedure as students with teachers in the school.

- **Unsupervised learning** [15] is a type that looks for previously undetected patterns in a dataset which has no pre-existing labels or a minimum of human supervision. This methodology has the main advantage of not to label datasets, which usually takes so much time.

- The last of these learning techniques is in the middle of supervised and unsupervised learning, and it is called **semi-supervised learning** [74]. It combines a small amount of labeled data with a large amount of unlabeled data during the training process. Intuitively, several problems with labeled data are like sample problems that the teacher solves for the class as an aid in solving another set of problems that will be unlabeled.

Inside unsupervised learning, there is another learning type that is the cornerstone of this project: **Deep Learning** [53]. It is the basis of the algorithm developed to this project case. In addition, it is combined with reinforcement learning to improve the algorithm behavior. Next section explains in depth these concepts.

3.1.2 Deep Learning Architectures

Deep learning is based on artificial neural networks with representation learning. It has been applied to different fields with satisfactory results, which can be compared, in some cases, with human expert performance. Some of the fields that are included in deep learning experiments are: Computer Vision, Speech Recognition, Audio Recognition, Natural Language Processing, Social Network Filtering, Machine Translation, Bioinformatics, Drugs Analysis and Design, Medical Image Analysis and Board Game Programs.

Artificial Neural Networks [50] (ANN) are the basis of deep learning architectures. They were inspired by processing and communication methodology that occur in biological systems. They differ from biological brains in that Artificial Neural Networks tend to be static, and biological brains that organisms have are more dynamic.

An ANN is based on a collection of connected artificial neurons, which loosely model the neurons in a biological brain. Each connection or edge transmits a signal to other neurons, imitating real synapses in a biological brain. An artificial neuron process a signal received and it can signal neurons connected to it. All these signals are real numbers. Then, the output of each neuron is computed by some non-linear function of the sum of its inputs. Neurons and edges typically have weights that are adjusted during the learning process, and weight increases or decreases the strength of the signal at a connection. Furthermore, neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold.

An ANN case is the one called **Deep Neural Network** [55] (DNN), which is the deep learning architecture utilized in this project. It is an artificial neural network with multiple layers between the input and output layers. Deep Neural Network finds the correct mathematical adaptation to turn the input into the output, and it can be a linear relationship or a non-linear relationship. This network moves through the layers calculating the probability of each output. Each mathematical manipulation is considered a layer, and complex DNN have many layers, hence the ”deep” reference. In addition, DNN can model complex non-linear relationships. DNN generates compositional models where the object is expressed as a layered composition of primitives. The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network.

Convolutional Neural Network [54] (CNN) is a multilayer neural network architec-

ture that was biologically inspired by the animal visual cortex. It is based in regularized versions of multilayer perceptrons which uses the convolutional operation, that is a specialized kind of mathematical linear operation. First layers recognize features, and the following layers recombine features into higher-level attributes of the output. CNN is made up of several layers that implement feature extraction, and then classification. It is used for different Computer Vision tasks, such as image and video recognition, and in various tasks of natural language processing.

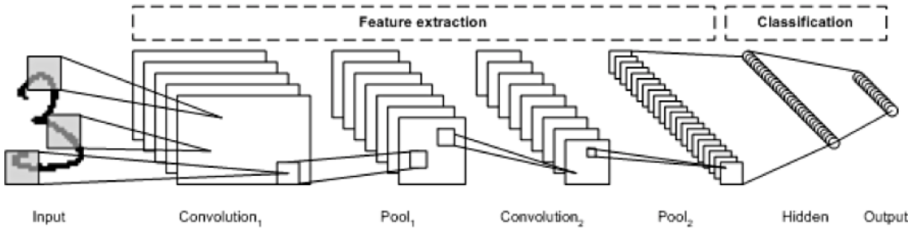


Figure 3.1: CNN functionality example, analyzing an image [54]

Recurrent Neural Network [29] (RNN) is also a deep learning architecture where connections between nodes form a directed graph along a temporal sequence, and also it can have connections that feed back into the same layer. It permits temporal dynamic behavior using its internal memory, where every calculated information is captured, stored and utilized to calculate the final outcome. Into RNN, there are two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that can not be unrolled. Some applicable RNN examples are related with tasks such as handwriting recognition, speech recognition and machine translation.

A RNN architecture example is the Elman network [29], which consists in a three-layer network with the addition of a set of context units. The hidden layer is connected to these units fixed with a weight of one. At each time step, input is fed forward and a learning rule is applied. The fixed back-connections save a copy of the previous values of the hidden units in the context units. With it, the network can maintain a sort of state.

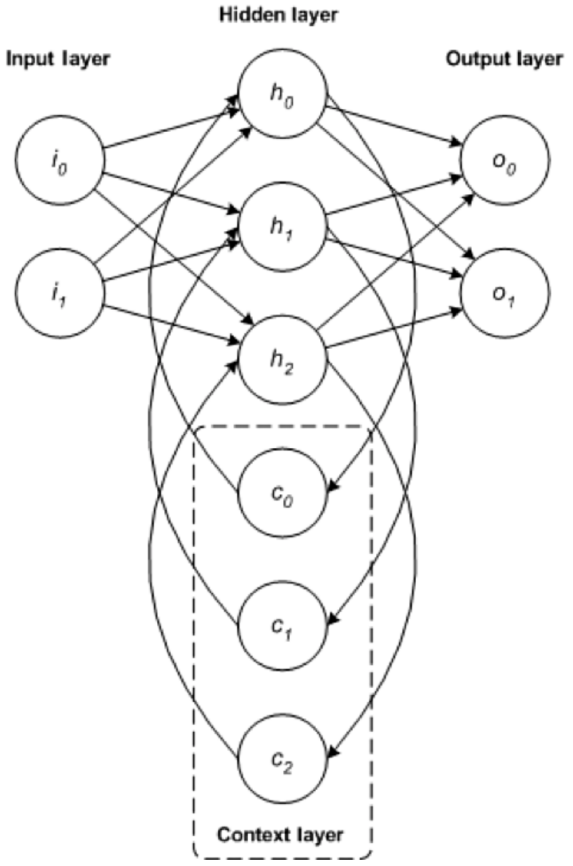


Figure 3.2: RNN architecture example: Elman network [54]

Other type is the **Deep Belief Network** [52] (DBN), that consists in a multilayer network in which each pair of connected layers are a restricted Boltzmann machine (RBM) [73]. A RBM is is an undirected and generative energy-based model with input and hidden layers connected between them, but not inside each layer. When a Deep Belief Network is trained with a set of examples without supervision, it can learn to probabilistically reconstruct its inputs, and its layers act as feature detectors. After this learning step, a DBN can be further trained with supervision to perform classification. There are several ingenious implementations of DBNs in real-life scenarios, like electroencephalography and drug discovery).

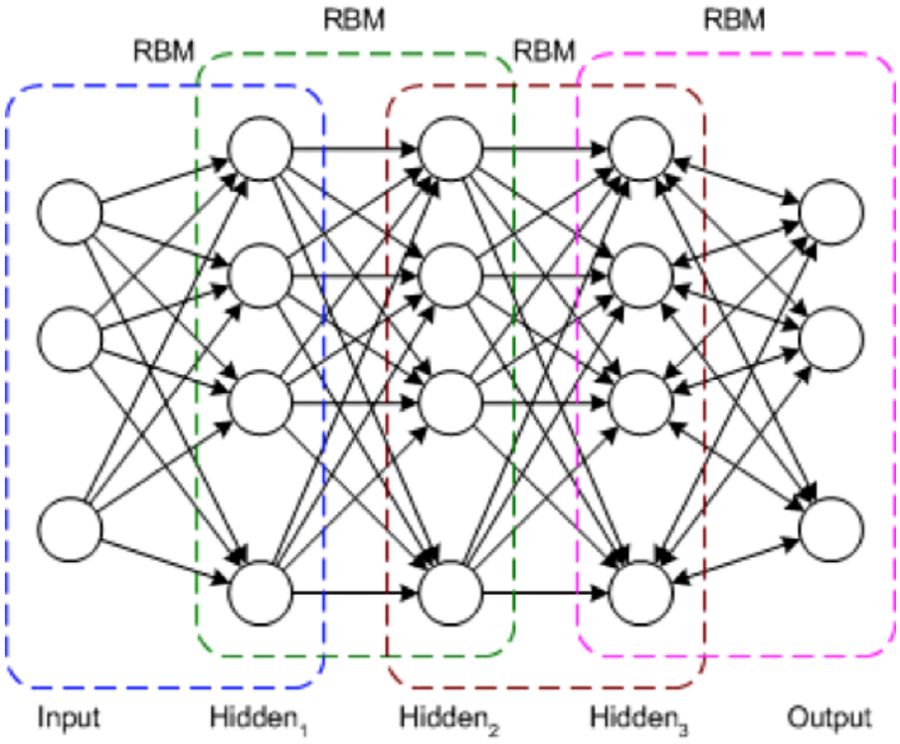


Figure 3.3: DBN architecture example [54]

3.1.3 Reinforcement Learning

One of the most important characteristics of this project is the learning technique implemented. During the development, reinforcement learning is the one selected to improve the deep learning model created. This section contains a description of its operation, and what is the reason of the environment creation.

Reinforcement learning [42, 72] is a technique concerned with how agents ought to take actions in an environment with the goal of to maximize the notion of cumulative reward. It implements a system of reward and punishment, and differs from supervised learning in not needing labelled input/output pairs or sub-optimal actions to be explicitly corrected. The main goal is to find a balance between exploration and understanding the dataset correctly, and exploitation of knowledge by using past data. Problems of interest in reinforcement learning have also been studied in the theory of optimal control, concerned with optimal solutions characterization and algorithms for their exact computation. This theory is less concerned with learning or approximation, particularly in the absence of a mathematical model of the environment, and in economic ambit, reinforcement learning is used to explain how equilibrium may arise under bounded rationality.

Environment is initialized in a Markov decision process [64] (MDP), because reinforcement learning algorithms utilize dynamic programming techniques. The main difference between the classical dynamic programming and reinforcement learning is that the latter does not assume knowledge of MDP model, and its target is larger than classical models. It is studied in many disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. Reinforcement is modeled with this Markov decision process, and the deep learning intelligence interacts with the environment in discrete time steps. For each time step, in the Markov decision process are different elements that affect to the learning:

- A set of environment and agent states.
- A set of actions made by the agent.
- The probability of a transition, at time t, to change from state s to state s' under action a.
- The corresponding reward when the state changes.

During the growth of reinforcement learning, different applications of this technique in stock markets have been done. An article of the economic group of the Erlangen-Nuremberg University [23] collect different publications and articles, and classify them into three principal groups: critic-only approach, actor-only approach, and actor-critic approach.

Critic-only approach is the most frequent application of reinforcement learning in financial markets. Its first introduction was made by Neuneier in 1996 [23]. The idea of this approach is to learn a value function in which the agent can compare the expected outcomes of ”going long” or “going short”. During the decision making process, the agent senses the current state of the environment and selects the action with the best outcome according to the value function.

The **actor-only approach** is the second most common approach. In it, the agent senses the state of the environment and acts directly, without computing and the expected outcomes of different actions. The agent produces a a policy from its actions, having in mind the previous state. The key advantage of this approach is the continuous action space of the agent with its faster convergence of the learning process.

Finally, the **actor-critic approach** combines the advantages of actor-only and critic-only approaches. The main idea is to simultaneously use an actor and a critic. The actor gives the current environment state to the agent, and the agent makes an action. The critic judges if the selected action was the best or not. In a few words: the actor learns to choose the best action considered by the critic, and the critic learns from the actions taken.

Comparing with these three versions, this master project has the main objective of simulate the trading process that occurs when a person operates manually in the market: if a person buys a share that will fall down the next days, there is a loss obtained because of this operative, and vice versa. This person needs to learn form its errors made, and this functionality is the desired to the project. This market simulation resembles to the actor-critic approach, working the environment as a critic, and the deep learning intelligence as an actor which learns to decide.

This methodology is achieved thanks to a specific environment created for this purpose, that analyzes daily the share price and all its movements, and all that info is passed to the deep learning algorithm, which learns from it. Furthermore, the conclusion reached is that a feedback is obtained periodically, and rewards are won according to the decision taken during last days or weeks. This environment based in reinforcement learning allows deep learning algorithm to improve its operation and decisions, adapting in each neural layer of the Deep Neural Network all the mathematical operations that are responsible of taking decisions.

3.2 Machine Learning Libraries

3.2.1 Python

Python [71] is a high-level programming language that is focused on general-purpose. It was created by Guido van Rossum with its first released in 1991. Python's design philosophy emphasizes code readability, and also it has constructs and object structures, aim to help programmers write in a clear way. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

In this project, Python will be used because it is the most important language to develop or create Machine Learning projects, although there are other languages less important, like C++, Java or Javascript [5]. The Python importance resides in all its libraries created by

social communities like numpy, scipy or pandas, which supports mathematical operations for multidimensional data, managing datasets, etc... Then, the most important development libraries implemented in this project are described.

3.2.2 Numpy

NumPy [37] is one of the most famous Python software library, which is used to apply scientific computing with Python. It adds support for large, multi-dimensional arrays, along with a large collection of high-level mathematical functions to operate on its.

The ancestor of NumPy, called Numeric, was originally created by Jim Hugunin. In 2005, Travis Oliphant created NumPy, incorporating features of the competing Numarray into Numeric, with extensive modifications.

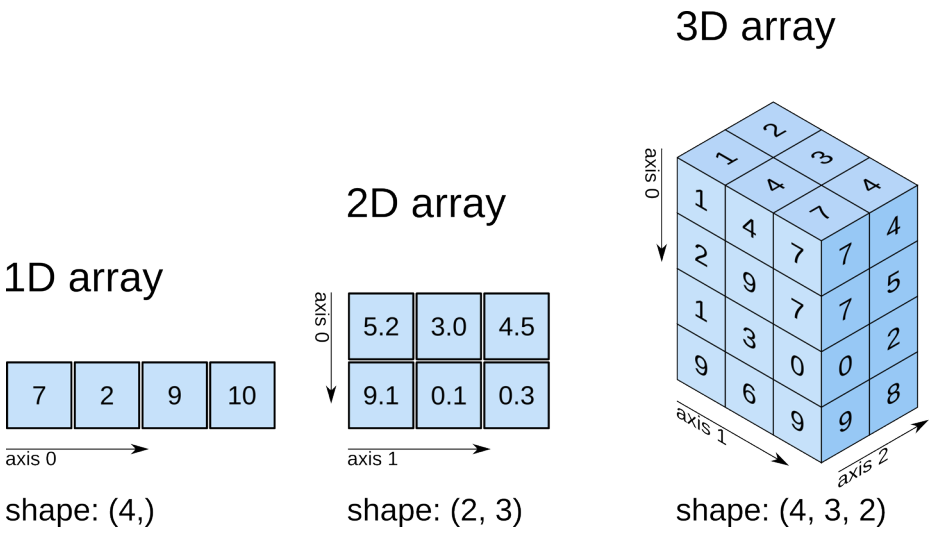


Figure 3.4: A NumPy structure [27]

Among all useful objects of this library, it is important to highlight:

- A powerful N-dimensional array object.
- Broadcasting functions.
- Tools for integrating C/C++ and Fortran code.
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

3.2.3 Pandas

Pandas [67] is another famous Python software library, used for data manipulation and analysis. It offers data structures and operations for manipulating numerical tables and time series. This library is very optimized for performance, thanks to code parts that are written in C language. It has some important features which are exposed:

- DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between different data structures.
- Reshaping and pivoting of data sets.
- Label-based slicing and fancy indexing.
- Data structure column insertion and deletion.
- "Group by" engine, allowing split/apply/combine operations on data sets.
- Merging and joining.
- Hierarchical axis indexing.

In this project, Pandas is mainly used to create dataframes. Pandas allow importing data of different file formats. During this implementation, CSV is the format utilized to collect data. Pandas also allows various data manipulation operations such as "groupby", "join", "merge" or "melt", and data cleaning features like "fill", "replace" or imputing null values, and these utilities are essential in this development to prepare a dataset correctly.

3.2.4 Plotly

This Python library [28] is an interactive and open-source plotting library that supports over 40 unique chart types with a large range of statistical, scientific, and 3-dimensional use-cases. Built on top of the Plotly JavaScript library, this library enables Python users to create beautiful interactive web-based visualizations that can be displayed in Jupyter notebooks, saved in HTML files, or served as part of a Python application. Plotly library makes high quality graphs, and it can create graphs like line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, multiple subplots, polar charts, and bubble charts.

Figure 3.5: Plotly has different chart types

An element that will be printed in future analysis and is related with graph prices is the **candlestick**. A candlestick is an attribute that can represent price variation during the period of tine that it covers. It is one of the elements that can create an **open-high-low-close chart** [66]. To explain it in more detail, here is its representation:

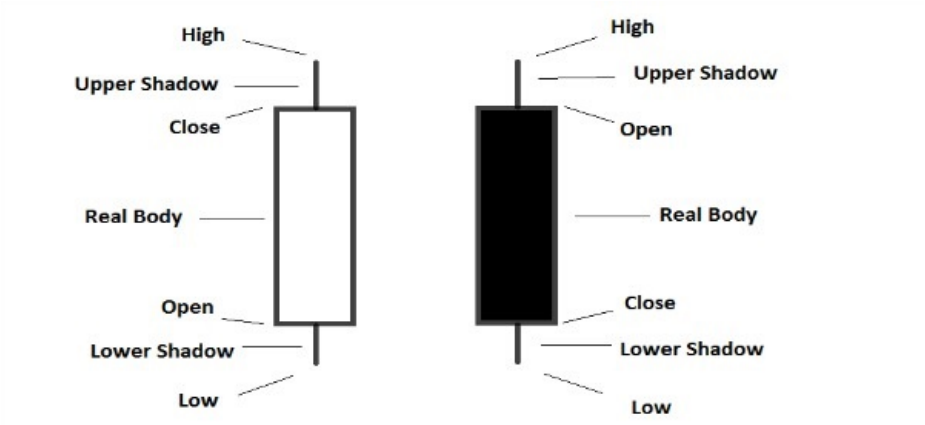


Figure 3.6: Structure of the candlestick [46]

And its parameters are:

- **Candlestick body color:** it depends if the day price fluctuation is positive or negative. It is classified if the close price is higher than the open price (in that case, the candlestick is white or green inside) or vice versa (in that case, the candlestick is black or red inside it).

- **Upper shadow:** it represents the difference between the highest price of the season, and the close or open price of the season (if it is a bullish or bearish day, respectively).
- **Lower shadow:** the opposite of the upper shadow, it represents the difference between the lowest price of the season, and the close or open price of the season (if it is a bearish or bullish day, respectively).

3.2.5 Chainer

Chainer [10] is a powerful, flexible, intuitive and open source deep learning framework, written purely in Python on top of Numpy and CuPy Python libraries. This development is led by Preferred Networks, which is in partnership with global enterprises that are leader in its sectors like IBM, Intel, Microsoft, and Nvidia. One of Chainer's most famous feature is its early adoption of "define-by-run" scheme, as well as its performance on large scale systems. The first version was released in June 2015, and it improves its popularity in Japan since then. Since 2017, it is listed in top 10 open source machine learning Python projects.

Chainer supports CUDA computation (Compute Unified Device Architecture) [51], which is a parallel computation platform. It includes a compiler and some tools, which are required to create algorithms for Nvidia GPUs. This example demonstrates the potential of this framework. In addition, Chainer supports different network architectures, including feed-forward, recurrent and recursive nets. It also supports per-batch architectures. Forward computation can include any control flow statements of Python without taking off the ability of back propagation. All these features make code intuitive, readable and easy to debug.

3.2.6 Kaggle

Kaggle [62] is an online community of data scientists and machine learning developers. This platform allows users to find, publish and explore different datasets, and also build or create models in a web-based environment, working with other data scientists and machine learning engineers. There are tournaments to solve data science challenges, which stimulate participation of all users, which is an important point to improve and maximize community.

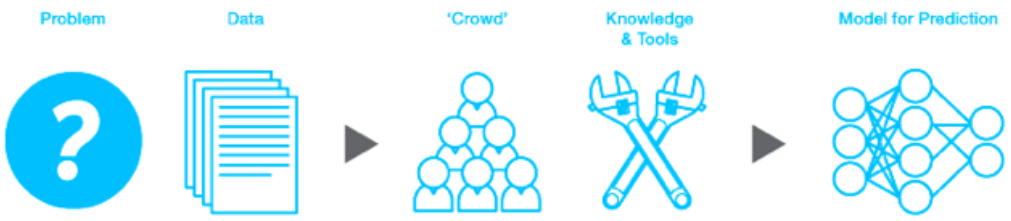


Figure 3.7: Kaggle model creation process [3]

Just as last mention, Kaggle has its own environment to create or develop different kind of programs. It was a possibility to develop on this platform, but the web page does not work really good, so it would not be used to create the intelligence. Instead of Kaggle, Jupiter will be used, because it works better and it does not need network connection.

The utility of Kaggle is that there are code examples that could be implemented or used as a template, and some share price datasets. A good dataset was found, which contains a lot of documents with different parameters that could be used for the development. All last parameters are useful to create candlestick charts, and also allow the possibility of using different analysis techniques.

3.2.7 Anaconda Navigator

Anaconda Navigator [2] is a desktop graphical user interface (GUI) included in Anaconda distribution. It allows to launch applications, and easily manage Conda packages, environments, and channels, all of that without using command-line interfaces (CLI) or commands. Anaconda Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

When a Python program is being created, it is common to have some Python dependencies problems, because many packages depend on specific versions of other packages. Usually, multiple versions of packages are used, and it is a great idea to have multiple environments, to reach that these different projects have versions that they need. With command-line program Conda, that is both a package manager and an environment manager, it is easier to help developers, and to be ensure that each version of each package has all the dependencies it requires. In that situation, Navigator is an easy way to work with packages without needing to type conda commands in a terminal window. It is useful to find packages, install them in an environment, run the packages, and also update them.

This graphical user interface is an easy way to program the artificial intelligence that is needed in the project.

3.2.8 Jupyter

Project Jupyter [70] is a nonprofit organization, which was created to develop open-source software, open-standards, and services for interactive computing. Jupyter supports execution environments in several languages. Its name is a reference to the three core programming languages supported by Jupyter: Julia, Python and R, and also a homage to Galileo's notebooks, reminding the Jupiter's moons discovery.
The philosophy of Project Jupyter is to support interactive data science and scientific computing across all programming languages, through the development of open-source software. Project Jupyter has developed the following interactive computing products:

- Jupyter Notebook (product that is going to be used on this project).
- JypyterHub: a multi-user server for Jupyter Notebooks.
- JupyterLab: the next-generation user interface for Project Jupyter.

Jupyter Notebook [61] is a web-based interactive and computational environment for creating Jupyter notebook documents. A Jupyter Notebook document is a JSON document which follows a specific schema, and contains an ordered list of cells which can contain code, text, plots and rich media. Normally, these kind of notebooks has an unique ending file extension, which is the ".ipynb" extension.

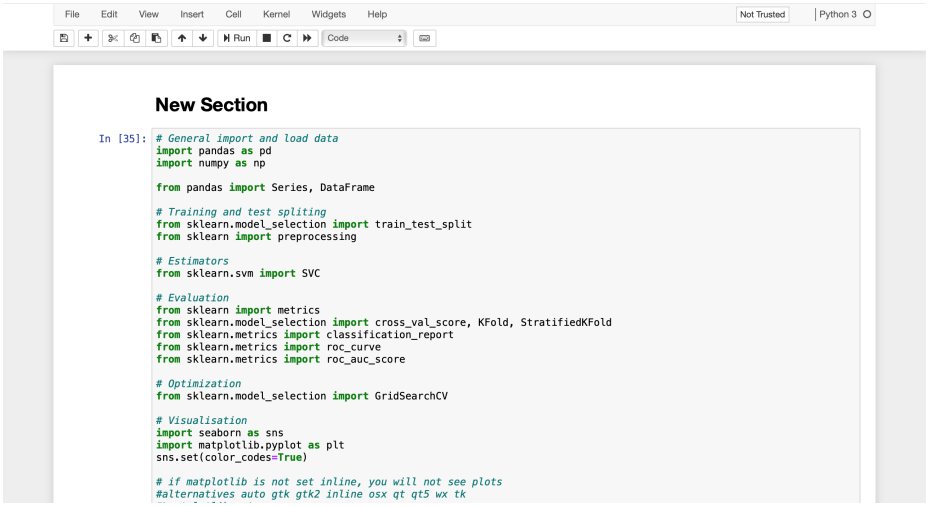


Figure 3.8: A Jupyter notebook

A Jupyter Notebook can be converted to a lot of open standard formats like HTML, LaTeX, PDF or Python, using a "Download As" utility in the web interface, thanks to the nbconvert library, or "Jupyter nbconvert" command line interface in shell. To simplify visualisation of Jupyter notebooks, the nbconvert library is provided as a service through NbViewer. NbViewer is capable of load a notebook in a web browser using a URL to any publicly available notebook document. The notebook is converted into HTML on the fly and display it to the user.

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default, Jupyter Notebook ships with the IPython kernel. In this project, the kernel used is based in Python3 (programming language used to deploy this application).

3.3 Trading Tools

Apart from libraries and programming code, there are different tools that are needed to carry through the project. Into this section, web tools and chart tools utilized during the project are explained in depth.

3.3.1 Tradingview

Tradingview [44] is an online platform used for analyzing stock market. It provides different kind of graphs like line graphs and candlesticks, and it has a free-licence. There are some monthly subscriptions, but there are only when real-time is needed. In this project, Tradingview is going to be used to analyze, in a manually way, stock markets. This process will be done before implementing any new dataset inside the project. It is very useful to detect market tendencies.

Last point is essential during the development, because if an incorrect train dataset is introduced, it will ruin all the learning of the intelligence. For example, if a bearish trend stock is introduced in a bullish intelligence, that intelligence will learn the incorrect situations to go into the market, and to go out when a bear trend is going to start.

Figure 3.9: An example of the Tradingview website [44]

Above this paragraph, there is an example of this web application. The photo is about an American stock graph (Carnival), and it is very easy to draw different lines to analyze the trend, or to see the SMA or Simple Moving Averages.

3.3.2 TA-Lib

TA-Lib [26] is a technical analytic library, which is widely used by trading software developers. It is required to perform technical analysis of financial market data. Its main characteristics are:

- Availability of more than 200 indicators (such as MACD, RSI, Stochastic, Bollinger Bands, etc).
- Candlestick pattern recognition.
- A Python wrapper based on Cython, which implements an API to calculate all types of technical analytics that the library includes.

Figure 3.10: Example of some analytics (MACD,RSI,STOCH...) that could be obtained mathematically from TA-Lib

TA-lib contains different types of indicators, classified in specific categories:

- Overlap Studies
- Momentum Indicators
- Volume Indicators
- Volatility Indicators
- Price Transform
- Cycle Indicators
- Pattern Recognition

In this project, TA-Lib could be a good implementation to obtain some statistics from a dataset implemented, because this API helps in the process of calculating these indicators. Important to underline that most of indicators have large formulas that are easier to implement with this library.

CHAPTER 4

Deep-Learning Based Reinforcement Learning Model

This chapter presents the main architecture of the project, including the dataset structure and the deep learning schema. That deep learning structure has different sections: from its neural structure, till the procedure to be trained itself, or to analyze and play with a dataset fragment.

4.1 Introduction

Architecture needs to be created in basis of the main goal of the project. This goal is to take advantage of price patterns with a neural network, and operate with American stocks to make economic profit. Following it, the architecture selected to develop this tool consists in deep learning instead of any other machine learning technique, because [33] deep learning tends to solve the problem end to end where as machine learning techniques need the problem statements to break down to different parts to be solved first, and then their results to be combine at final stage. Furthermore [40], it does not need to label data, and is efficient at delivering high-quality results. On the other hand, deep learning needs a lot of data to be trained, but it is solved because share datasets obtained are very big.

Between all the deep learning architectures available, *Deep Neural Network* (DNN) was selected instead of any other one. It fits with the idea of performing the mathematical composition of the structure to improve outputs [55]. In addition, there was other neural architecture choices, but they were not selected because [20]:

- *Convolutional Neural Network* (CNN) is very focus and specialized in computer vision problems.
- *Recurrent Neural Network* (RNN) is designed for language and forecast models. It has a short memory inside the cells, but it was considered that this property could not be useful.
- *Deep Belief Network* (DBN) has a different topology than DNN: not all its layers are connected each other, and for this project this property was not desired.

Following, the main structure of the project architecture is similar to Fig. 4.1. This architecture is focused on the idea of having a neural structure and a reinforcement class as the cornerstones of the architecture. In this structure, a dataset which could be categorized in train or test dataframes, is introduced into the preparation phase. During this step, they are prepared before being introduced into the deep learning intelligence. In this architecture, the DNN algorithm, colored in blue, is connected to an environment class, painted in purple. Environment element is used to simulate the trading activity, and it returns the corresponding feedback and the actual state to the deep learning algorithm. At the end of the process, the DNN returns two information arrays: a total reward array and a total losses array, used to value the behavior of the algorithm. These arrays are explained later with full detail.

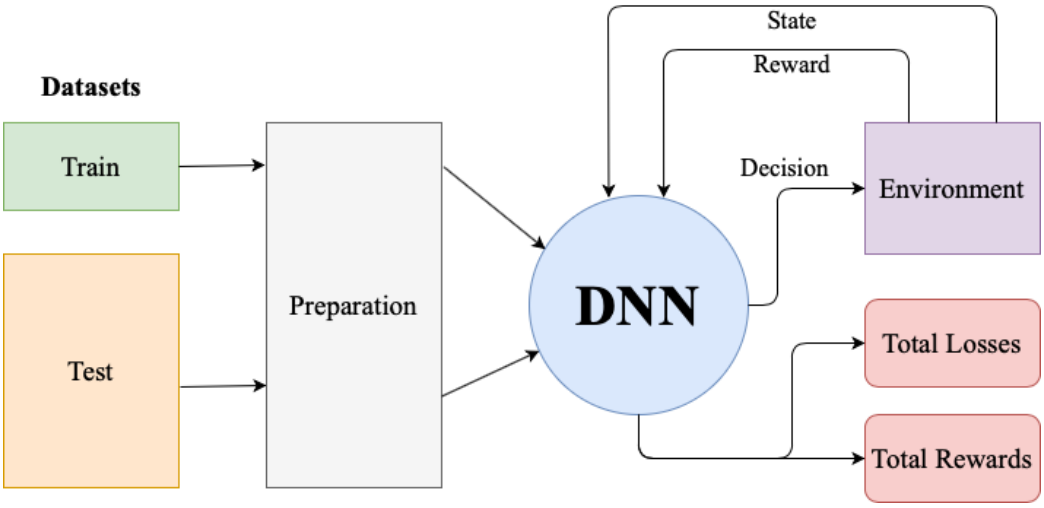


Figure 4.1: Project Architecture

During this chapter, the DNN and the environment class are explained in sections 4.2 and 4.3, respectively. In addition, in the final part of the chapter, the training and the testing process are explained with full detail. An *Unified Modeling Language* (UML) diagram is in Fig. 4.2, which is compound by the two classes that define this project: the environment class and the DNN class. They are connected and they use functions from the other class, but all this operation is explained during this chapter. Underline this picture because it is the basis of the algorithm behavior.

Figure 4.2: UML Diagram

4.2 Deep Neural Network Structure

The main characteristic of the DNN is that layers are interconnected through neurons and they have mathematical optimization during the training, which is a main point in the algorithm improvement. To program a DNN, it is important to import a library that supports these type of learning. In this case, "Chainer" library [9] is the library used.

There are two main elements inside the library that are necessary to elaborate the neural structure:

- **Link:** it is an object that can saves parameters. It also establish the relationship between neurons of different layers, and in the Fig. 4.3, they are represented by arrows. An important point is that they behave like regular functions while replacing some arguments by their parameters.

  The most frequent use of links is the Linear link. It represents a mathematical function $f(x)=Wx+b$, where the matrix "W" and the vector "b" are parameters. This type of link is invoked as the following expression:

  $$f = L.Linear(i, j)$$

  The last formula declares the Linear link and its input and output dimensions. In the last expression, "i" and "j" are the dimensions of input layer and layer 1 respectively, that are connected by linear links. For this project, all tests of this parameter will be done with a few number of hidden layers, and mixing different quantities of neurons.

- **Function:** it is the option that determines the hyperparameter utilized into each layer of the intelligence. There are several functions available to use in the Chainer library, and during the next chapter, a selection of functions to be tested will be done.

A Chainer tool that is used during the training process is the Zerograds function [9]. It is capable to restart or initializes all gradient arrays by zero. It simulates that a reset button is pressed. Gradient arrays are the most important variable when a neural structure is created. They are the main element responsible of the different behaviours of the intelligence. When an intelligence is trained using deep learning, each gradient matrix is modified, suffering changes to alter reply of the intelligence.

With all last parts explained, it is possible to display the neural network structure. The Fig. 4.3 contains a graph representation of the DNN implemented. It is created with an input and output layers, and it also has the possibility of having one or two hidden layers, named L1 and L2. Hidden layer quantity depends on what structure configuration is being trained. Furthermore, each layer could have an specific number of neurons, represented by i, j, k or l. But during the project, input and output layers will have a size of 91 and 3 respectively. Then, hidden layers will have different neuron layer sizes, to finally find what size is the best to be used in the project.

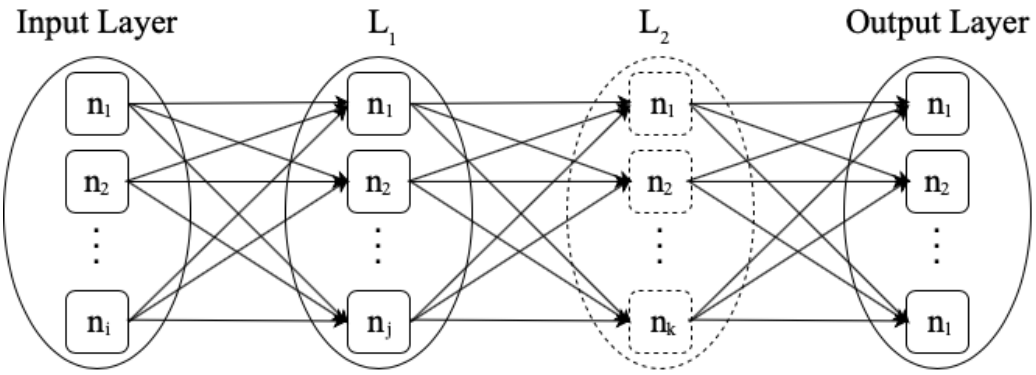


Figure 4.3: DNN Architecture

4.3 Environment class

During the analysis of the Erlangen-Nuremberg University article [23], the conclusion of using a actor-critic approach was made. In this approach, it is necessary to create the critic element. In this project, this element is the environment class. It is the one that gives the current state and results obtained to the actor, which is the DNN.

This class simulates the share market, and it is created to give a feedback to the deep learning algorithm according to decisions taken. It evaluates every step made by the intelligence, and then all the results reported by the environment are collected. These results can be used to train the DNN or to evaluate the effectiveness of the algorithm.

Environment provides two main functions which can be utilized by the deep learning intelligence to operate in the stock market. These two functions are the described below:

- *Reset:* it initializes the local variables to its default value. It is normally used when a process starts, to clean all last environment used.
- *Step:* also called "decision", it provides different possibilities to the DNN, and then the environment class is in charge of the analysis of the step taken by the intelligence. During every day "t" passed, also called turn, the algorithm can choose between three different resolutions which will imply different consequences:
  - "Stay": selected when the intelligence does not want to take action during that day.
  - "Buy": when this option is selected, the intelligence buy one share of the company operated. That share is saved, having the possibility of selling it in the future.

– "Sell": used when the intelligence wants to indicate that all the shares that are in its possession must be sold during that turn.

The process made by the environment can be represented in the following schema. In the picture, there are different time periods that corresponds with each day. The deep learning algorithm sends a "decision" to the environment class, corresponding to "stay", "buy" or "sell" actions. These actions are represented in the picture with 0, 1 and 2 respectively. When a decision is taken, the environment receives it and calculates rewards and losses reached in the next day. These parameters are joined with the observations array, and they are sent to the DNN in order to improve its behavior.

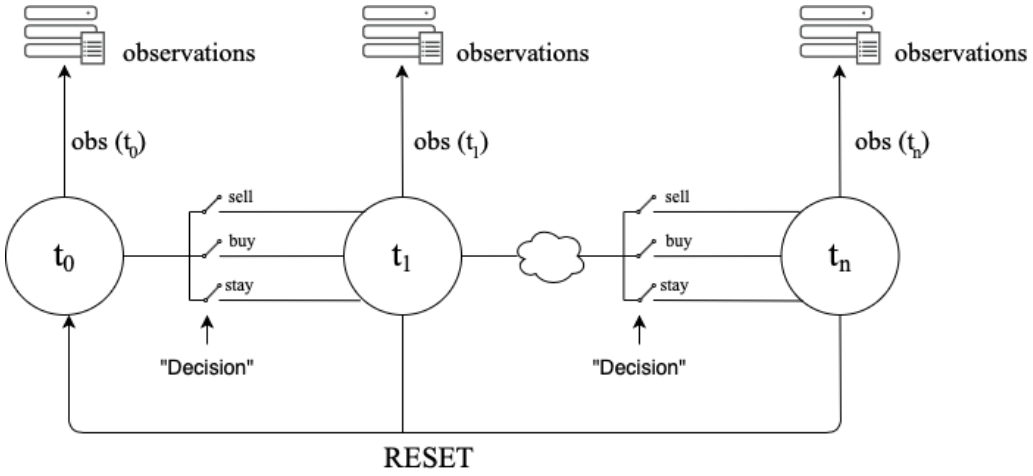


Figure 4.4: Environment structure

On each daily step, the environment class is in charge of recalculating: the share pocket that the intelligence has, the positions closed during that turn, and the reward obtained. Each turn concludes with a positive or negative reward, which feeds the reinforcement learning method explained in last points. The recompense obtained depends on the situation:

- If the algorithm tries to sell its position without any share in the pocket, it implies a negative recompense that turn.
- If the algorithm sells its position and the selling balance is negative, it implies a negative recompense that turn.
- If the algorithm sells its position and the selling balance is positive, which implies positive recompense that turn.

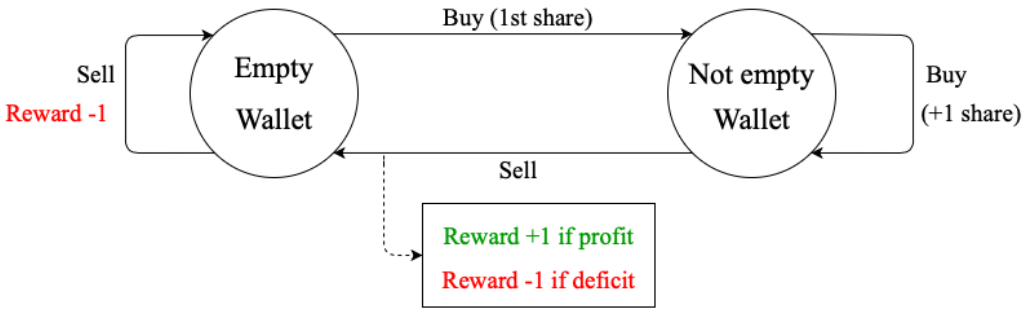


Figure 4.5: Reward process

4.4 Training Process

There are different main points that must be considered in detail during the intelligence learning process. Among all them, there are different arrays that save rewards and losses obtained during the process, and they are underlined in the UML diagram mentioned in the introduction of this chapter. Next, the process about how the deep intelligence is trained is explained below, accompanied by a training process picture. The principal variables utilized during the training process are the following ones:

- The dataset utilized.
- The deep learning intelligence "Q" and the environment class.
- A memory array that saves all the results that the intelligence is having during the process. It saves in every row, for each step done, the past observation "p_obs", the action done "p_act", the reward and the newest observation "obs".
- Losses and rewards accumulated during the process.

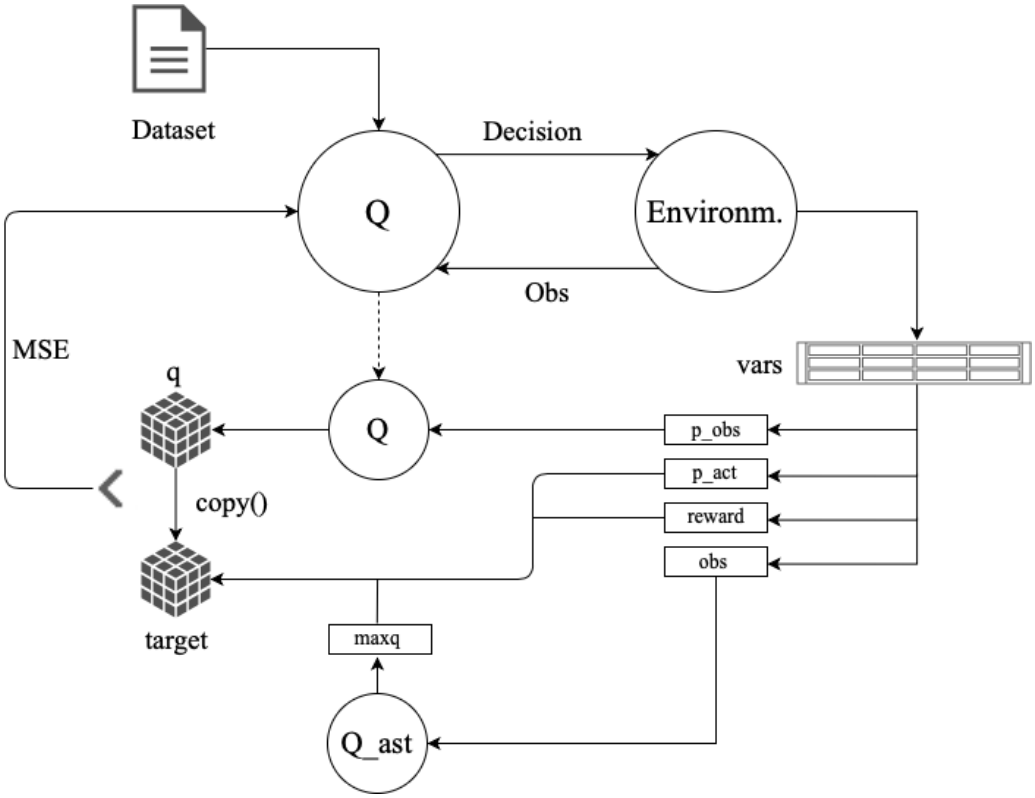


Figure 4.6: Training process

There are more constraints specified inside it, which are related to batch and memory size, epoch cycles and train frequency. The learning process has different steps, that are the following ones:

1. A train dataset is introduced into the deep learning intelligence "Q"
2. The deep learning intelligence is trained using that dataset and the environment class.
3. There is a memory array which is continuously saving variables of the algorithm decisions processed in the environment class. Variables saved are "p_obs", "p_act", reward and "obs", all of them explained in the beginning of the section.
4. When the memory array reaches its maximum length, defined by a constant value, the learning process begins.
5. The different parameters contained in memory are separated in unique arrays: past observations, past actions, rewards and present observations.
6. Past observations are passed to the deep learning algorithm to obtain its decision matrix, that is in charge of taking decisions. This matrix is called "q".

7. Observations are passed to "Q_ast". It is a copy of the deep learning "Q", and it is updated every "update_q_frequency" cycles. Then, The maximum argument of each row of the matrix is obtained and saved in "maxq" parameter.

8. Target matrix is created. It is a copy of "q" matrix. Then, with the following formula, some cells of the target matrix are replaced, with the goal of improving this structure.

$$target[x, b_pact[x]] = reward[x] + 0,97 * maxq[x] \tag{4.1}$$

9. The Mean Square Error between the "q" and target matrix is calculated. The error is given to the algorithm, with the main objective of recalculating its gradients and improve future predictions.

10. Finally, the "total_losses" and "total_rewards" arrays are updated with all losses and rewards obtained during the training process. In next chapter, it will see that these arrays are passed to a graph function that prints the final result of each parameter in different charts. With this charts, it is easier to compare the different behaviors of each intelligence configuration.

4.5 Testing Process

That process is similar to the training process, with the main difference that testing do not modify the deep learning intelligence. Apart from that point, the procedure has parts in common with training process. The testing procedure has the following steps:

1. A stock dataset is chosen from the data folder.

2. This dataset file can be separated into two parts: the train part used during the past training process, and the rest of the data that are available to proof the accuracy of the intelligence, which is the test dataset.

3. Both datasets are charged into the environment to initialize the process.

4. Two copies of the DNN are created to the testing process. They are in charge of sending decisions called "act" during each turn, and then they receive the observations "obs" obtained by the environment to process the next actions.

5. At the same time, profits and rewards generated from both intelligence process are saved to finally print the results in a chart.

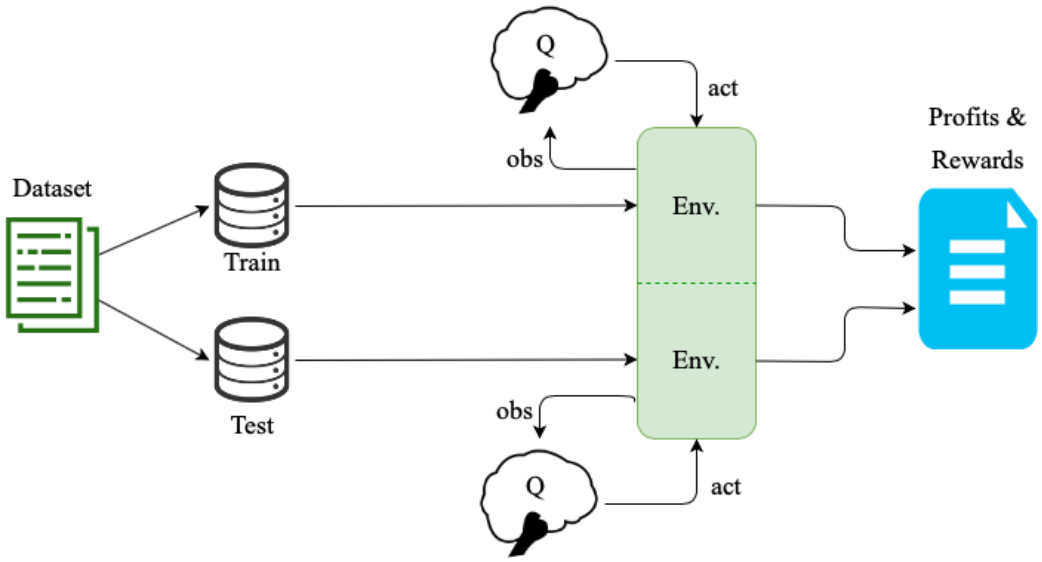


Figure 4.7: Testing process

After having all the most important project processes explained, it is time to continue with the next point, and apply all of them into a real case. It will be seen that every process has its importance in every step, making all of them a meticulous behavior to obtain the best results.

CHAPTER 5

Experimentation and Evaluation

This chapter includes a neural network optimization, modifying neural structures and hyper-parameters and ejecting different tests, comparing all the results to finally obtained the best parameters combination. In addition, all datasets utilized during the process are explained.

5.1 Introduction

As explained in the previous chapter, all the development process can be described into three principal steps. The first one corresponds to the data preparation, in which different company stock data prices, divided into train and test datasets, are selected and analyzed. Then, a *Deep Neural Network* (DNN) with reinforcement learning is established as the "brain" of the project, learning from the train datasets prepared. Finally, this intelligence can be trained with different test datasets to collect different conclusions from this process. This chapter describes these different steps with full detail to give the reader a general process defined.

5.2 Dataset Analysis

In this section, the main dataset utilized is explained with full detail. The principal sections to have in mind are the structure that the dataset has, the main data parameters to be used during the training and testing process, and the preprocessing of the proper dataset.

5.2.1 Data Structure

Getting a stock price dataset is not an easy task, because most of web pages contribute with datasets that are accessible only if you pay to obtain it. Also, it is important to use the newest dataset available on internet websites, to cover the maximum time possible. For this project, the dataset used was downloaded from Kaggle [34]. Even though it was released some years ago, it is sufficiently complete to analyze how the intelligence acts. There are other price datasets available like Yahoo finance [79], which has more recent values, but it has some cases in which different daily price values are not completed correctly. To facilitate the task and save time comparing them, Kaggle dataset was selected for the development.

The Kaggle dataset downloaded is formed by different files and folders. It has two main folders: one with *Exchange-Traded Fund* (ETF) values, and the other one with individual share datasets, in which all of them have the same internal structure. Kaggle dataset structure characteristics are explained in table 5.1:

Dataset			
Elements	Number of items	Description	Dataset dates
Stocks	7195	Daily prices of American companies	Length variable End date: 10/11/2017
ETFs	1344	Price values of Exchange-Traded Funds	

Table 5.1: Dataset statistics

Related with individual share datasets, the table 5.1 describes the internal structure of each share file. All stock datasets are saved in *Comma-Separated Values* (CSV) format. In addition, every file has different attributes that are described below:

```
Date,Open,High,Low,Close,Volume,OpenInt
1984-09-07,0.42388,0.42902,0.41874,0.42388,23220030,0
1984-09-10,0.42388,0.42516,0.41366,0.42134,18022532,0
1984-09-11,0.42516,0.43668,0.42516,0.42902,42498199,0
```

Figure 5.1: Example of an internal dataset structure

- **Date** of each row.
- **Open** price of the day.
- **Highest** price of the day.
- **Lowest** price of the day.
- **Close** price of the day.
- **Volume:** number of operations that the market has in that day.
- **OpenInt:** this data is at zero in every row, so it is ignored.

5.2.2 Price Selection

An essential decision to the learning process is to choose, from all parameters available in each dataset file, what principal price is better to be used during the development. This study is needed because not all options available are optimal to use. To decide it, a comparison between attributes is created using different analysis tools. The different elements that will be part of this comparison are:

- **Daily close price of the S&P 500 index:** close price is the main parameter chosen from different websites when they represent an index or a share [79, 30]. It implies that close price needs to be the main parameter to be compared with all price parameters contained in the dataset files.
- **Apple dataset:** for this comparison, it is necessary to pick an stable share which follows the index trend, to compare different behaviors between price parameters and index tendency. Apple is one of the most important companies of the S&P 500 index [41], so it helps for the price analysis.

Figure 5.2 corresponds with the dataset prices contained in Apple stock file. It includes daily values of 2017, from February to November. The image contrast the daily price volatility that the share has between open, high, low and close prices.

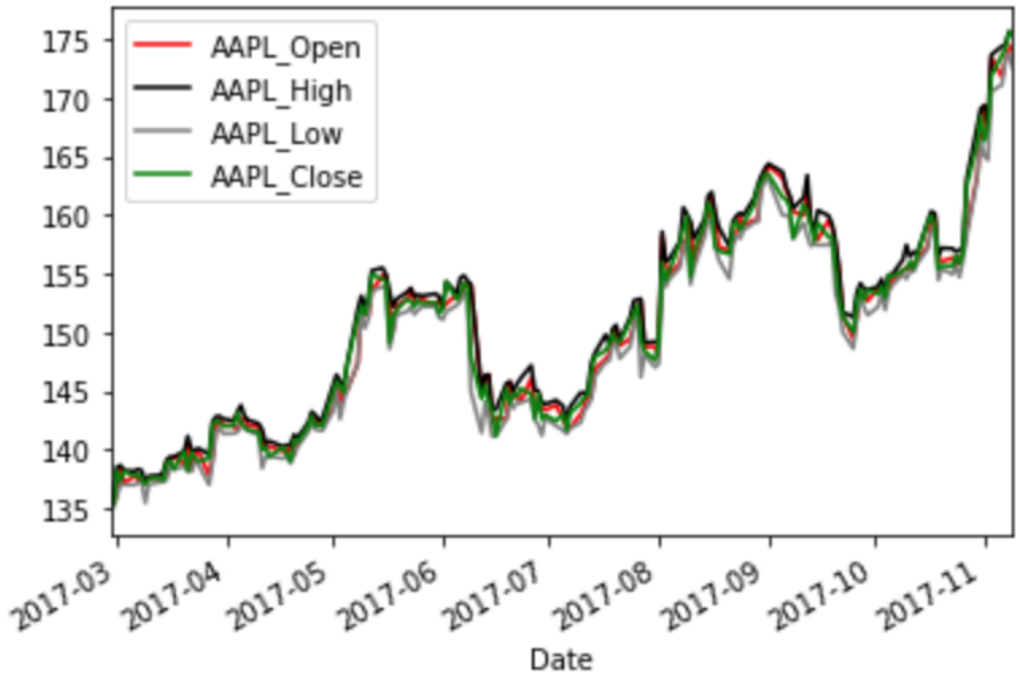


Figure 5.2: Apple stock prices comparison

Next, the correlation table 5.3 displays the different relationship between S&P 500 and Apple stock prices. It is useful to detect which price from Apple stock resembles the S&P price. Prices of Apple have a similar correlation because it comes from the same share.

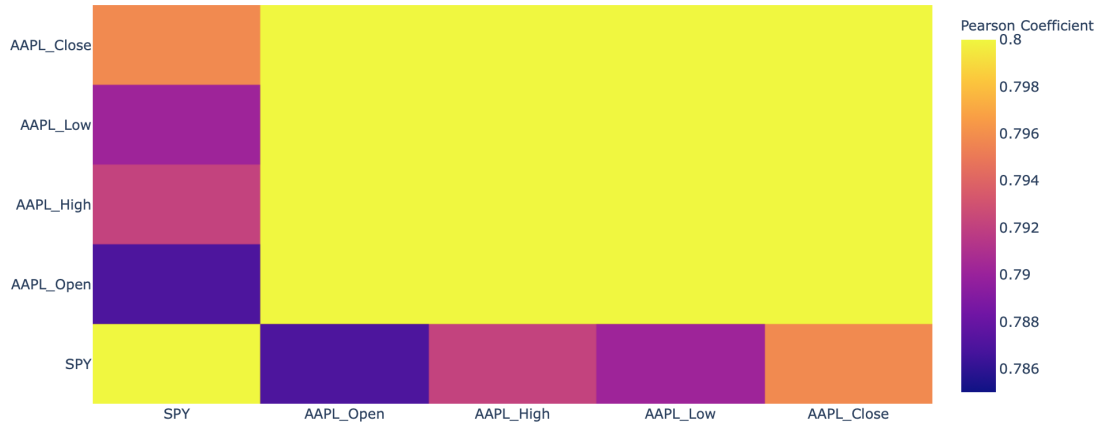


Figure 5.3: Correlation table between Apple and S&P 500 prices

The conclusion obtained is that Apple close price is the most similar to the index, detected in the hottest color seen in the correlation table. This theory is also related with the S&P close price, and it reflects that daily price fluctuation is less important than the final price, which indicates the final result of the day.

5.3 Market Stocks Selection

An important factor in this project is to recognise the different stocks utilized during this project. It is important to underline the main point of not having stocks with a high volatility to avoid an erroneous training from the algorithms. In addition, different market trends are extracted from every dataset extracted, to train and test the algorithm in every possible case. This section contains a description of each company utilized, and different parameter comparisons between them, with the main objective of analyze their situations.

5.3.1 Theoretical Description & Dataset Selection

Firstly, to select and separate these datasets, it is essential to select different companies that are models of their sectors, economically solid and without a high volatility caused by news or other causes. Following there is a list with all the companies selected to this process:

- **Apple Inc. (Market ticket: "AAPL"):** [49] it is an American multinational technology company headquartered in Cupertino (California). This company designs, develops, and sells consumer electronics, computer software, and online services. It is considered one of the Big Four technology companies, alongside Amazon, Google, and Microsoft. Apple is famous because of its main creation: the iPhone: the first

multi-gesture touch screen phone of that company. But this company has other products and services that are a market reference, and are normally copied by its rivals.

The test and train Apple datasets, with 319 and 598 values respectively, are used during the development and testing, and they can be extracted from the Figure 5.4. Both datasets are contained into a bullish trend, with some price corrections that can train the algorithm to not get used to a constant trend.

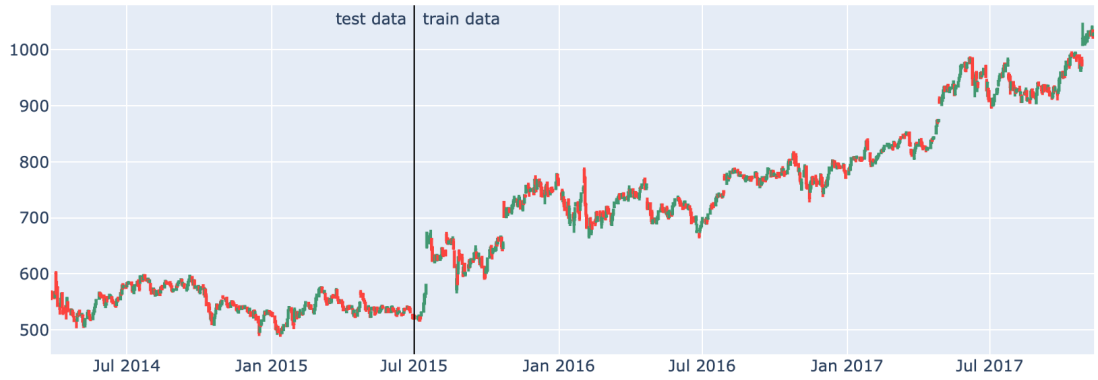


Figure 5.4: Apple test & train charts

- **Microsoft Corporation (Market ticket: "MSFT"):** [65] is an American multinational technology company headquartered in Redmond (Washington). It develops, manufactures, licenses, supports, and sells computer software, consumer electronics, personal computers, and related services. Its best known software products are the Microsoft Windows operating systems, the Microsoft Office suite, the Xbox video game consoles and the Microsoft Surface touchscreen computers. The word "Microsoft" is a portmanteau of "microcomputer" and "software".

  The train dataset, with 2212 values, is similar to the Apple one: it covers a bullish tendency with some corrections. On the other hand, the test dataset is contained into a lateral market, which will have a great utility during the testing section to prove the intelligence behavior in this case.

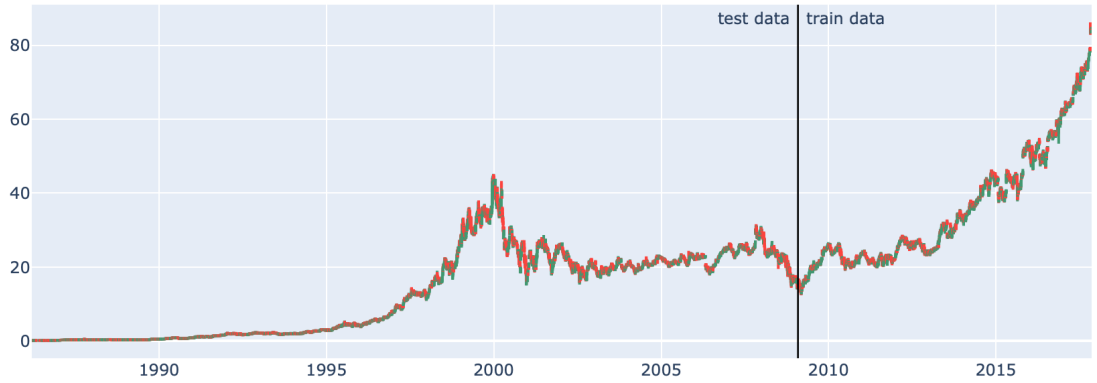


Figure 5.5: Microsoft test & train charts

- **The Coca-Cola Company (Market ticket: "KO"):** [77] is an American multinational corporation, and manufacturer, retailer, and marketer of nonalcoholic beverage concentrates and syrups. The company is headquartered in Atlanta (Georgia), but incorporated in Delaware. The most famous product of the company is the Coca-Cola drink. The company takes its name because of the product).

  The train dataset (2171 samples) has the same characteristics as the Microsoft and Apple datasets. It is also contained into a bullish trend, that will learn the algorithm to correctly operate in this cases. A correct implementation is to select different bullish trend datasets to not train the algorithm with the same data. Test dataset (9905 values) will be used in the same way as the Microsoft test data, taking the lateral trend to analyze the algorithm behavior in this tendency.

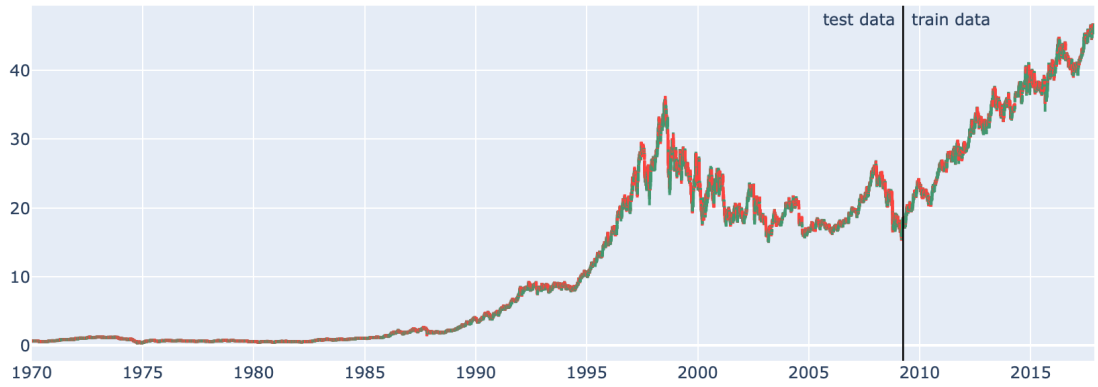


Figure 5.6: Coca-Cola test & train charts

- **Walmart (Market ticket: "WMT"):** [78] is an American multinational retail corporation that operates a chain of hypermarkets, discount department stores, and

grocery stores. It is headquartered in Bentonville (Arkansas). Walmart has 11,503 stores and clubs in 27 countries, operating under 56 different names. The company operates under the name "Walmart" in the United States and Canada, as "Walmart de México y Centroamérica" in Mexico and Central America, as "Asda" in the United Kingdom, as the "Seiyu Group" in Japan, and as "Best Price" in India.

Highlight the train dataset trend with around 2200 values, which has a large and abrupt correction. It is completely different to the previous ones, and this element helps the intelligence to have unexpected situations and learn from them. In the testing dataset, the lateral tendency from year 2000 stands out among other older data, and it will be useful in future sections.

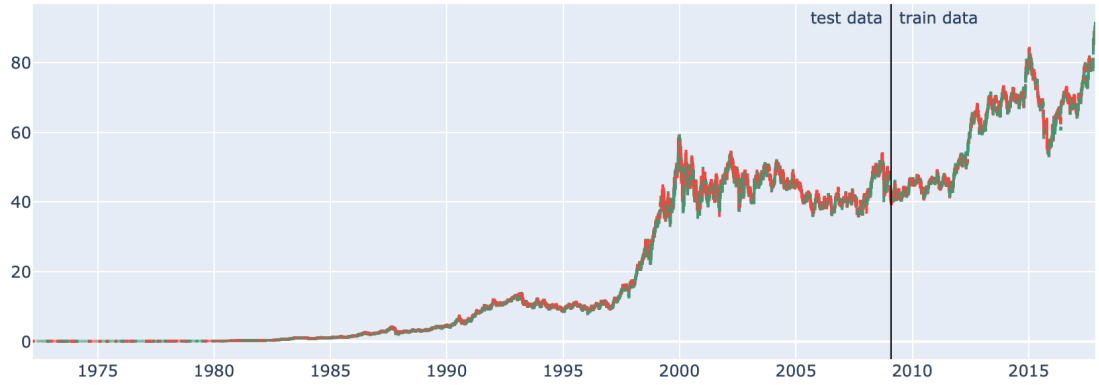


Figure 5.7: Walmart test & train charts

- **JPMorgan Chase & Co. (Market ticket: "JPM"):** [60] is an American multinational investment bank and financial services holding company headquartered in New York City. JPMorgan Chase is ranked by S&P Global as the largest bank in the United States and the sixth largest bank in the world by total assets, with total assets of US $2.687 trillion. It is also the world's most valuable bank by market capitalization.

  JPMorgan price graph corresponds with the last bullish trend dataset extracted for training. It is very similar to the Coca-Cola chart, so they have identical tendencies: a bull trend contained into the train dataset, and a lateral and bearish trend in the test part. It implies that they have similar influences in the algorithm training and testing process.

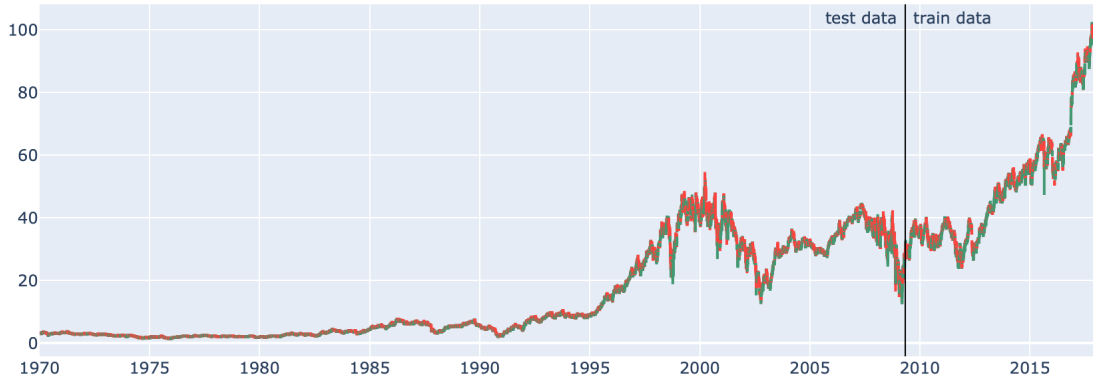


Figure 5.8: JPMorgan test & train charts

- **Amazon (Market ticket: "AMZN"):** [48] is an American multinational technology company based in Seattle, with 750,000 employees. It focuses on e-commerce, cloud computing, digital streaming, and artificial intelligence. Amazon is considered one of the Big Four tech companies, (along with Google, Apple, and Microsoft). It is the world's largest online marketplace, AI assistant provider, and cloud computing platform (measured by revenue and market capitalization). In addition, Amazon is the largest Internet company by revenue in the world, and the second largest private employer in the United States.

  Amazon is a stable share, which has few volatility, but during last years it increases its volatility because of a company revaluation by the investors. It seems that they put their confidence on the company. An interesting training dataset is the one contained between 2010 and 2014. It is a growing lateral tendency, interesting to be trained by the algorithm. Testing dataset after year 2015 could be interesting to study.

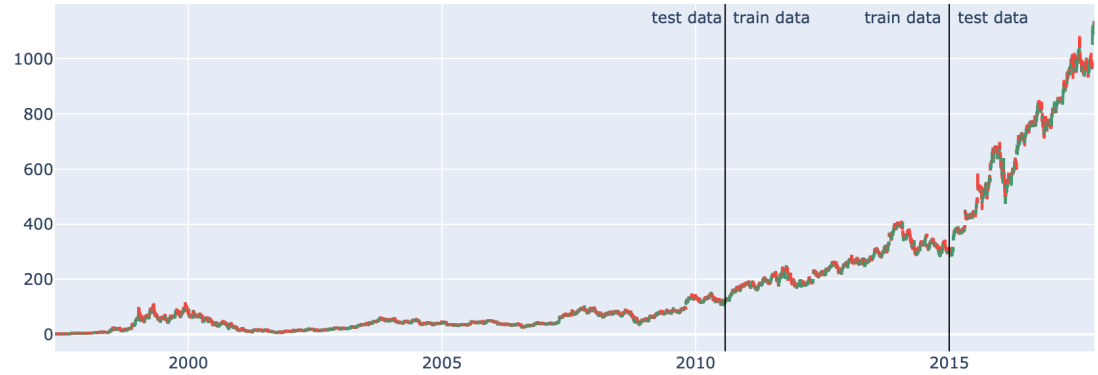


Figure 5.9: Amazon test & train charts

- **Facebook (Market ticket: "FB"):** [57] is an American social media and technology company based in Menlo Park (California). This company was founded by Mark Zuckerberg. Facebook is one of the world's most valuable companies, and it is considered one of the Big Five technology companies along with Microsoft, Amazon, Apple, and Google. Facebook offers different products and services, like Instagram, WhatsApp, and Oculus.

  Facebook has the same situation of Amazon: it is involved in a bullish trend, but until 2015 it has a false growth, reflected in low peaks during 2014. It can be take as a train dataset to prepare the algorithm in this situation.

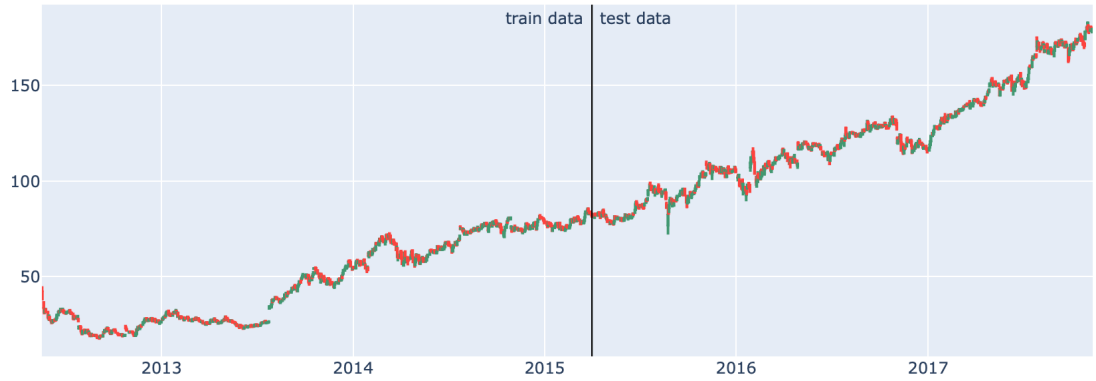


Figure 5.10: Facebook test & train charts

- **Advanced Micro Devices (Market ticket: "AMD"):** [47] is a multinational semiconductor company based in Santa Clara County (California). This company develops computer processors and related technologies for business and consumer markets. AMD's main products include microprocessors, motherboard chipsets, embedded processors and graphics processors for servers, workstations, personal computers and embedded system applications, etc...

  Advanced Micro Devices has an irregular trend in this temporal chart. Probably, it could be tag as a lateral trend situation. The best point of this chart is that it is easy to take a bearish period in which make some tests: from 2006 to 2012, the share price was decreasing from 400 to less that 10 dollars per share. This dataset is the one to be used in this bearish point.

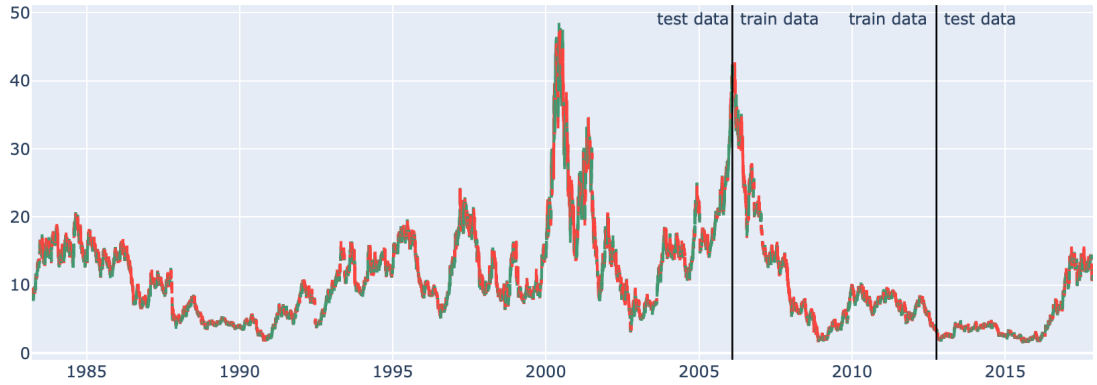


Figure 5.11: AMD test & train charts

- **Procter & Gamble (Market ticket: "PG"):** [69] is an American multinational consumer goods corporation headquartered in Cincinnati, (Ohio). It was founded in 1837 by William Procter and James Gamble. It specializes in a wide range of personal health/consumer health, and personal care and hygiene products; these products are organized into several segments including Beauty; Grooming; Health Care; Fabric & Home Care; and Baby, Feminine, Family Care.

  In this company, the situation is a bit different from AMD. Procter & Gamble chart has two different trends in the chart time period: there is a bearish trend during first months of 2015, and when it finishes, a bullish trend was started. But the period between 2000 and 2010 covers two recessions that are interesting for the training task.

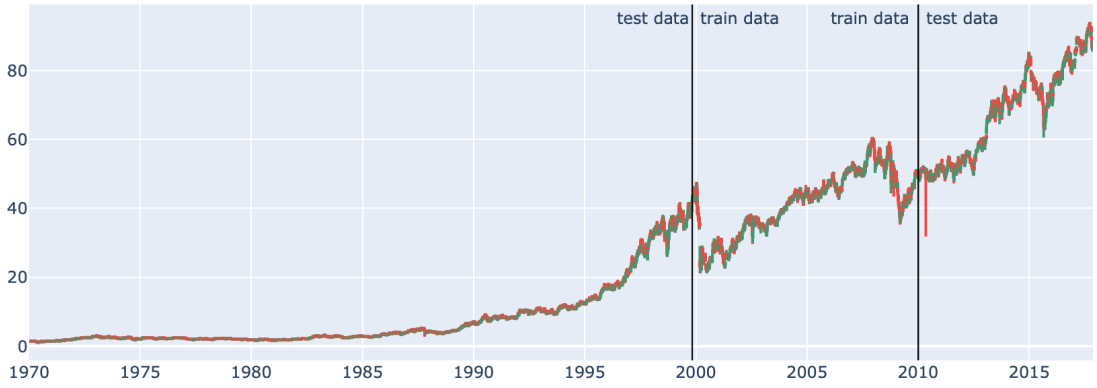


Figure 5.12: Procter & Gamble test & train charts

From share companies exposed before, all the training dataframes obtained have an attribute in common: they have a defined trend with large or short corrections. In their majority, the market is contained into a bullish trend, which is the one that is in an in-

creasing way, having minimum and maximum price values that are higher just as the graph time is continuing. That is because the American market suffers an inflation period during the last decade, since the last recession leaves the economy. It implies that most of the companies have been accumulate a big overvaluation in their prices.

Train and test datasets extraction is made by the creation of a tool that is able to cut the dataset into different sections. This tool has the ability of, selecting the start and end dates of the train dataset, save different data prices into dataframe structures. The remaining dataset is contained into a test dataframe for its usage. The final result obtained is a test and a train data dataframes, with the graphical representation, that helps in the work of verifying if the intersection is well done. It can be seen in the last pictures, because all of them are obtained from this tool.

The dataset evaluation and separation from the main stock dataset is evaluated manually: from the main chart printed, the training period dates are selected and put into the code. Then, its execution implies the desired separation. Code has the ability of entering one or two dates to split the dataset between this dates.

5.3.2 Technical Company Analysis

Another tool created for this project consists in a plotting method that takes all the shares utilized in the project and the American main index, all into a single chart. It is used to compare different trends and evaluate that all of them are good choices for the project dataframes. The result when the tool is executed is seen on the Figure 5.13.

Highlight that all the shares utilized confirms the tendency that the main index has (S&P 500), which is printed in blue color. It is a great indicative of the company selection made before. If a company was wrongly selected, this tool would show us a tendency contradiction before rectify it. In addition, Amazon stock price was divided into 1:3 factor with the objective of printing correctly all the shares.

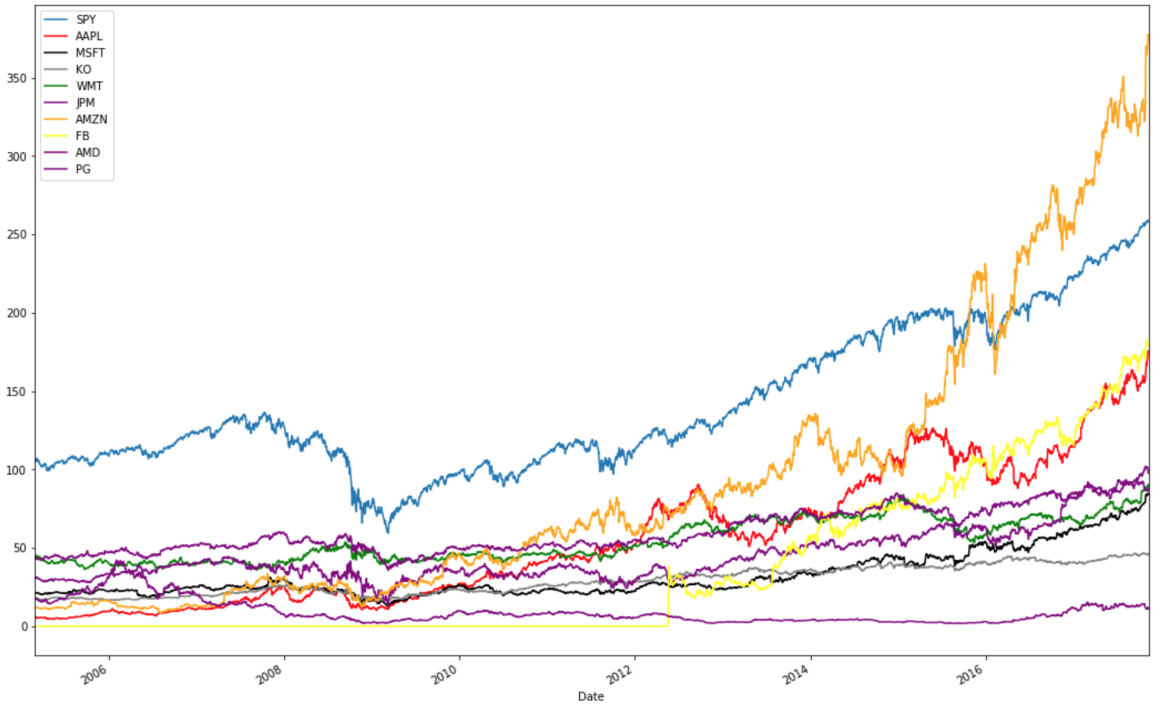


Figure 5.13: Chart of different stock prices, compared with the main country index (S&P500)

To complement the previous tool, a correlation table is implemented, which expresses the similarity of stocks introduced in it. It is based on the correlation table, which is also a Pandas function that return the correlation values between the different elements.

	SPY	AAPL	MSFT	KO	WMT	JPM	AMZN	FB	AMD	PG
SPY	1.000000	0.908707	0.954261	0.904231	0.849480	0.953522	0.911887	0.954436	-0.183600	0.961019
AAPL	0.908707	1.000000	0.892894	0.953426	0.902820	0.863617	0.917256	0.916594	-0.407120	0.913205
MSFT	0.954261	0.892894	1.000000	0.857968	0.778177	0.965024	0.958538	0.972307	-0.148269	0.918848
KO	0.904231	0.953426	0.857968	1.000000	0.920233	0.823619	0.893482	0.888525	-0.485972	0.940747
WMT	0.849480	0.902820	0.778177	0.920233	1.000000	0.771328	0.796222	0.834998	-0.469481	0.908770
JPM	0.953522	0.863617	0.965024	0.823619	0.771328	1.000000	0.923945	0.940752	-0.110271	0.908789
AMZN	0.911887	0.917256	0.958538	0.893482	0.796222	0.923945	1.000000	0.965609	-0.258982	0.898881
FB	0.954436	0.916594	0.972307	0.888525	0.834998	0.940752	0.965609	1.000000	-0.236093	0.930833
AMD	-0.183600	-0.407120	-0.148269	-0.485972	-0.469481	-0.110271	-0.258982	-0.236093	1.000000	-0.331071
PG	0.961019	0.913205	0.918848	0.940747	0.908770	0.908789	0.898881	0.930833	-0.331071	1.000000

Figure 5.14: Correlation table of different shares with the S&P500 index

To facilitate the understanding of the correlation table, a correlation matrix was developed. It is the visual representation of the Figure 5.3. It has different colours to determine each Pearson coefficient between a pair of values. Colours can be configured when the

Pearson coefficient range is defined. It is important to define that the Pearson Correlation coefficient is a linear indicator that measures the relationship scale between two quantitative and continuous variables [68]. Resuming, warming colors indicate that company prices are more related, and cold colors has the opposite meaning.

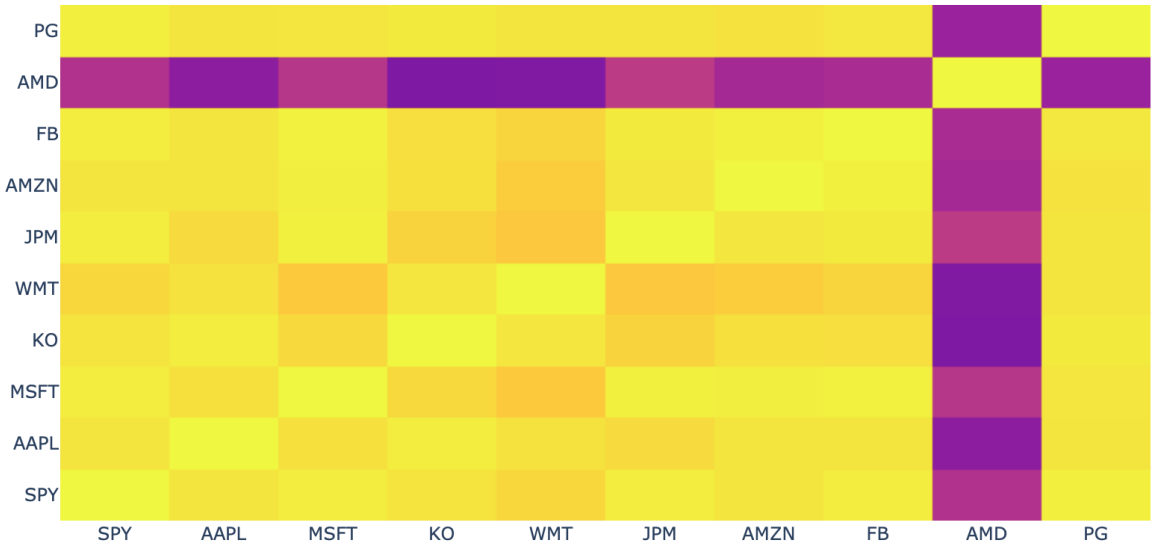


Figure 5.15: Correlation matrix of different shares with the S&P500 index

It is easy to see the similarity of the different company stocks used. All of them have warming colors, except the correlations with AMD stock, which have cold colors, indicating a disparity between share tendencies. This result could be probably caused by the lateral trend emphasized in this stock. To confirm this cause, a deeper analysis will be done.

The method to analyze the AMD trend in more profundity is to use TA-Lib library. This library has the ability of, in this case, printing the *Simple Moving Average* (SMA). They are use to study stock tendencies, interpreting different situations depending on the SMA position, if they are above or below the price, etc. Remember that

To clear up this study, the next list describes different possibilities that can occur to determine the tendency direction [36]:

- Stock price above SMA200 indicates bullish trend.
- Stock price below SMA200 indicates bearish trend.
- If the stock price cross above SMA200, a bullish trend is beginning.
- If the stock price cross below SMA200, a bearish trend is beginning.
- If the SMA50 cross above SMA200, a bullish trend is beginning.

- If the SMA50 cross below SMA200, a bearish trend is beginning.

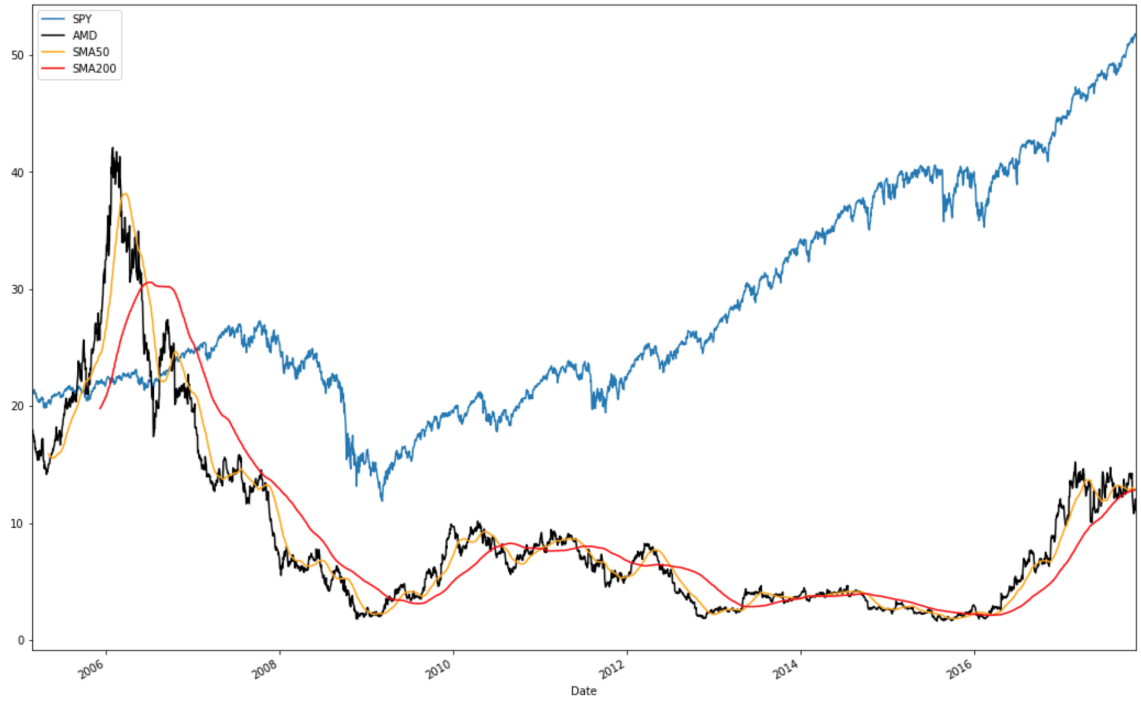
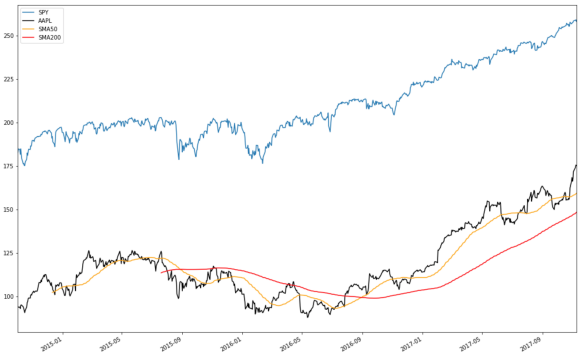


Figure 5.16: AMD trend analysis

In the Figure 5.16, it is easy to analyze the stock following last rules. Between 2005 and 2009 years, AMD trend can be identified as a bearish one: peaks are decreasing and SMA200 is above the stock price. From the end of this trend until 2016, the share is contained into a lateral trend: the stock price fluctuates between 4 and 12 dollars per share, and it only breaks the top price after 2016. All this behavior causes the incongruity seen in Figure 5.15.

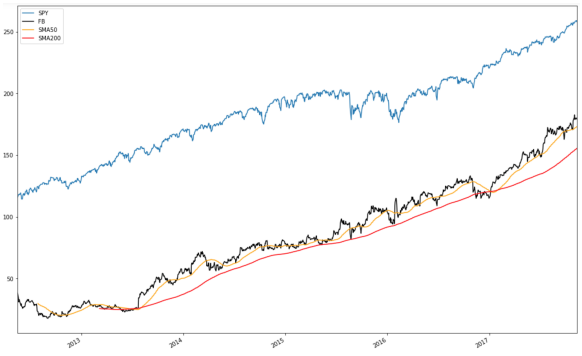
The other stocks used, which have high correlation with S&P 500 index, have the performance shown in Figure 5.17. They are also compare with their SMA50 and SMA200, to interpretate in a better way the different trends.



(a) Apple

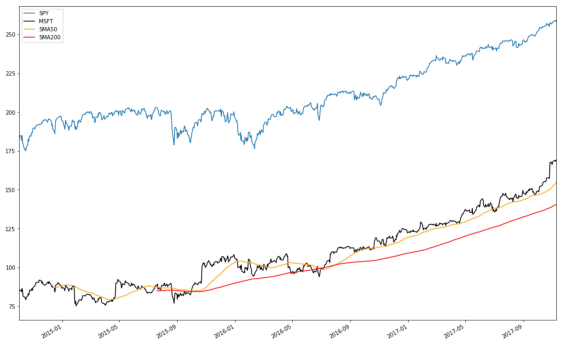
(b) Google

(c) Amazon



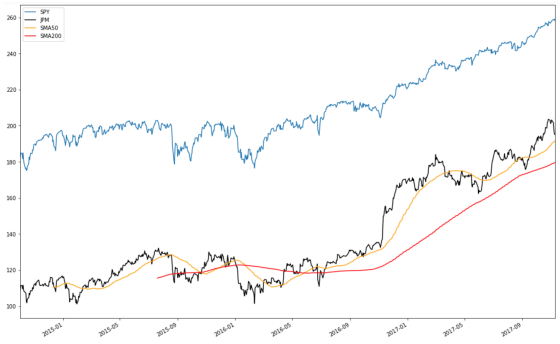
(d) Facebook

(e) Coca-Cola



(f) Microsoft

(g) Procter & Gamble



(h) JPMorgan

(i) Walmart

Figure 5.17: Analysis of share trends

5.3.3 Data Preprocessing

Having a description of all the companies utilized during the project, the next step is to prepare all the datasets before using them to obtain the different models. The first goal of this point is to select the data columns that are necessary to the project. Following the Apple price analysis done in the beginning of this chapter, the most optimum data chosen for the project is the close price of each company dataset. That price column is selected with the date column too. This is because date is necessary to print later different charts with the behavior of each model used.

After selecting the most important columns, the next point is to separate the training and testing datasets. On each share, this break point is chosen in different situations, but the main characteristic of this separation is that this point depends on when the trend starts. For this project, the different models are trained with a complete tendency to facilitate the training process for all the algorithms. Training is done for every model configuration with bullish, bearish and lateral markets, to have a global market knowledge.

The Table 5.2 reflects the sample quantity of training and testing datasets for each stock. Underline that the total training samples are around 25% of the global samples.

Share	Training Samples	Testing Samples
Apple	598	319
Microsoft	2212	5771
Coca-Cola	2171	9905
Walmart	2212	9231
JPMorgan	2150	9925
Amazon	1113	4040
Facebook	683	698
AMD	1857	6880
PG	2515	9560

Table 5.2: Samples Quantities

5.4 Neural Structure Optimization

It is important to find the equilibrium point in which the neural network has an optimum number of hidden layers and a correct number of neurons in each one. To optimize it, there is not a global theorem or law to determine the exact number of neurons and layers to put: each problem has its optimum configuration. The best option chosen to optimize it is with test and error method.

The process that this project will carry out is the following: different tests with different neurons and layers quantities are performed. The main objective is to get two different models: one with a single hidden layer, and other different model with two hidden layers. To reach it, the algorithm will be trained with different datasets, made up from stocks explained in last point, and different neurons quantity configuration: 3, 15, 25, 50, 100 or 200 neurons per each hidden layer. The best model will be chosen depending on the performance obtained. This performance is measured by two graphs: loss and reward graphs. They represent the money lost and the reward obtained during each step respectively. These results depends on the intelligence behavior, and they will represent which models are the best to be chosen. It is important to specify that these charts are the basis of the deep

reinforcement learning in this project, because they show if the neuron structure operates in the correct way, or if it makes several mistakes.

The next step will be the hyperparameter optimization. In Chainer library, there are different hyperparameters to be chosen, and each one is represent by a model function. The hidden layer tests are done with the hyperparameter selected by default, which is called "relu", but during the hyperparameter optimization 10 functions are tested to find the best model configuration. The evaluation step is the as in the hidden layer optimization: using loss and reward charts.

5.4.1 Model 0: Basis Structure

The basis model consists on a deep neural network structure with an input and output predefined layers, and with one or two hidden layers that are the ones modified to optimize the different models. The two models are called model "0a" and "0b", and they are shape with a one-neuron hidden layer and with double-one-neuron hidden layers respectively. They are the prototype for the future models and their optimizations, in which the hidden layer structures are modified. Specifically, their number of neurons and the hyperparameter utilized in them.

Highlight that fixed layers that are not changed during the project. That is because these layers have a specific size, and the input and output data are passed always with a fixed quantity. In this development, during each neural execution, it receives 100 samples on its input, and it exports three values that represent the decision probabilities for each action (buy, sell or hold). Conclude that only hidden layers are the only ones optimized during the model searching.

5.4.2 Model 1: 1 Hidden Layer Optimization

In this model case, the algorithm was tested with several neuron quantities in the hidden layer. The most optimum comparison process is firstly to visualize the reward charts of all the candidates for this model, seen in the Figure 5.18. From them, the next step is to take the best ones and analyze their looses, to finally select the best one-hidden-layer model. All the considerations about this first test round are the following.

With **3-neuron hidden layer** and **15-neuron hidden layer**, the algorithm reaches its value around 200-250, which is around the mean of all one hidden layer algorithms. It is seen that they also have a lot of fluctuation in the reinforcement obtained, and also

there is not a positive reward trend, probably caused by an underfitting training. It is probable that these few neuron quantities are not sufficient to operate with this market problem.

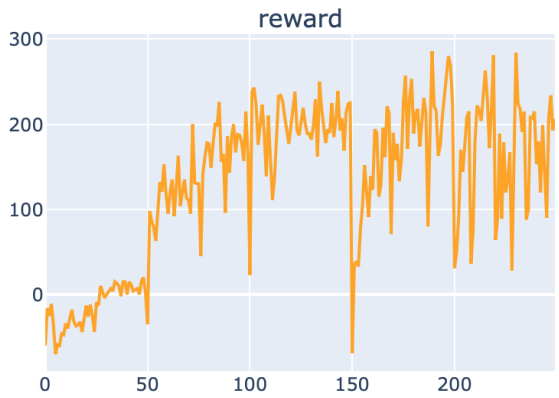
Also, even if their loss charts are not exposed because of their bad reward behavior, they have high losses obtained in different train period parts, reaching the 2000 value in several cases and with peaks higher than that value. It confirms the theory obtained from the previous analysis.

The **25, 50 and 100-neuron hidden layer** cases show that they have a light positive trend, because the minimum reward values of each one are growing up in the chart. On the other hand, although these cases can be selected to compare beyond the other one hidden layer models, each one has different characteristics that are underlined below:

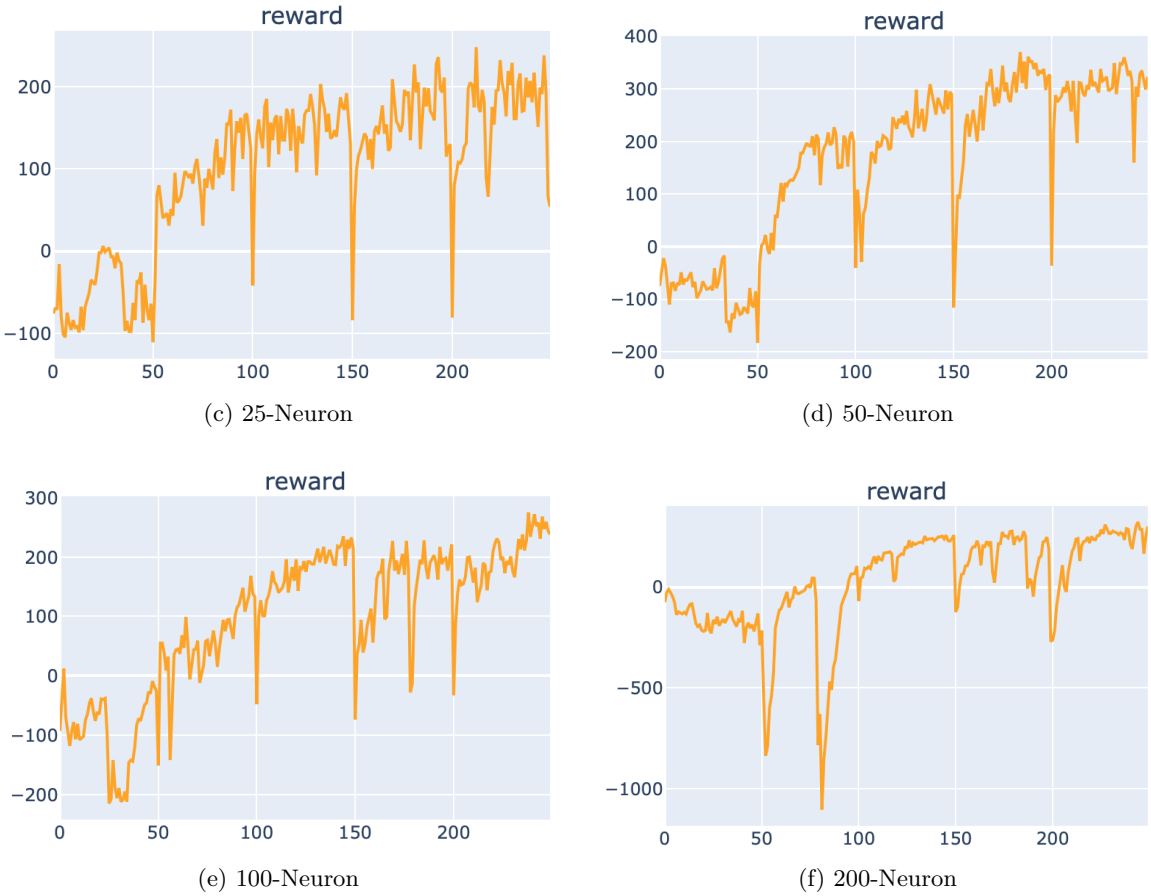
- The **25-neuron hidden layer** model has high reward variations during its training, complicating the task of predict its mean value.
- The **50-neuron hidden layer** shows a stable performance with low peaks quantity, exposing its facility to be adapted to the market environment.
- The **100-neuron hidden layer** has a similar behavior compared with the 50-neuron model, but reaching low values during the training process and having more low peaks than the other one.

Finally, the **200-neuron hidden layer** model is automatically rejected from this analysis, because it is not capable to reach the 100 reward value during the process. In addition, it is absorbed into the negative reward zone, which means that is closing loss actions continuously. Probably this model is a great example of an overfitting case because of its behavior.

(a) 3-Neuron



(b) 15-Neuron



(c) 25-Neuron

(d) 50-Neuron

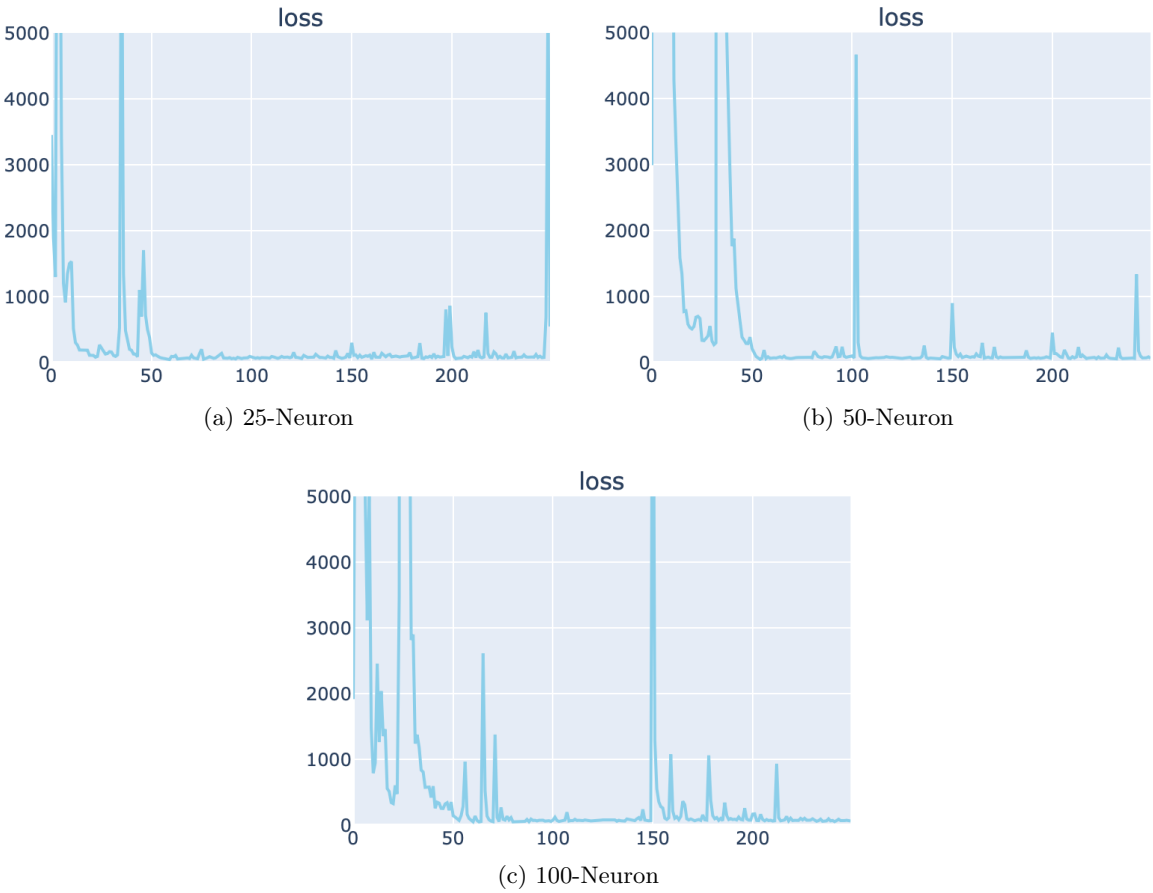
(e) 100-Neuron

(f) 200-Neuron

Figure 5.18: Reward Results of 1 Hidden Layer Models

The last step is to compare the best models of the reward analysis and dismiss the others. The best ones are the 25, 50 and 100-neuron hidden layer algorithms. Their losses are exposed in Figure 5.19, and different conclusions can be extracted from the graphs:

- 100-neuron hidden layer model is less prepared to be used in this problem than the other models chosen, because it has a loss value which is less predictive than the others.
- Between 25-neuron and 50-neuron hidden layer models, the second one incurs in more volatility because of the loss peaks that its chart contains, but also the 50-neuron hidden layer model is much more stable in the reward fact. Imposing reward difference above loss variation, **the 50-neuron algorithm seems better to do this task and is the one chosen to represent model number one.**



(a) 25-Neuron

(b) 50-Neuron

(c) 100-Neuron

Figure 5.19: Loss Results of 1 Hidden Layer Models

5.4.3 Model 2: 2 Hidden Layers Optimization

With this 2 hidden layers model search, the process is identical to the previous one: comparing reward charts to determine which models are the best ones, and then select the most appropriate one, getting different conclusions because of the analysis made. Reward charts can be seen in the Figure 5.20. The conclusions taken from these graphs are explained in the next paragraphs.

First of all, there are three models with a lateral trend in the reward chart: **double-3-neuron, double-15-neuron and double-25-neuron hidden layers**. A lateral trend is a situation in which the value fluctuates between two principal values. For this three cases, the value is contained between 100 and 300, but each structure has its own behavior:

- The double-3-neuron layers model is the worse one from this little group. It is continuously changing the response abruptly, and it complicates the task of obtaining a defined trend. Because of this reason, this model is discarded for the analysis.

- Double-15-neuron and double-25-neuron layers models have a similar performance. The graphs reflects that the value stabilizes in 200 during the last train periods. In addition, double-25-neuron layers model has less variance than the other model, but it is necessary to analyse their losses to take a final decision.

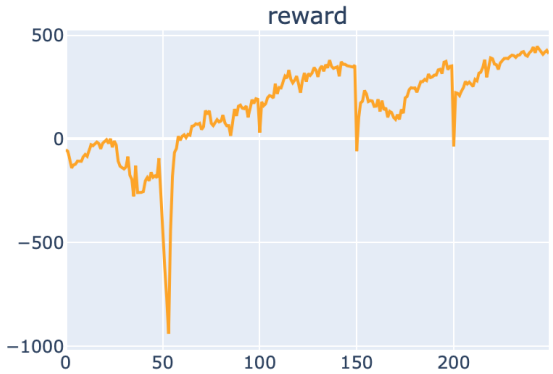
Double-50-neuron hidden layers model seems like the best model obtained with two hidden layers. Its fluctuation is minimum, and it also has a stable tendency in the rewards obtained during the training process. It can be a good candidate for the loss analysis.

Finally, the **double-100-neuron hidden layers** algorithm is the worse one, having more variance in the reward reached, and with several periods with negative reward. It can not be compared with other models selected in this comparison because of these reasons exposed.

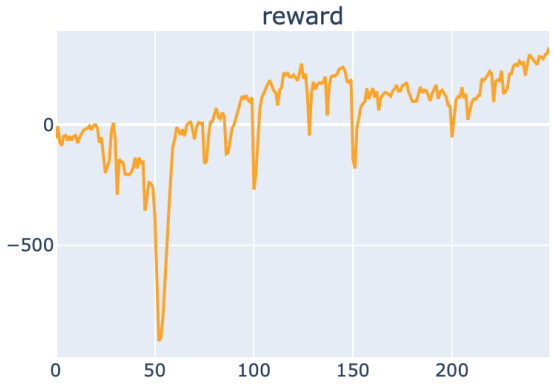
(a) Double 3-Neuron

(b) Double 15-Neuron

(c) Double 25-Neuron



(d) Double 50-Neuron



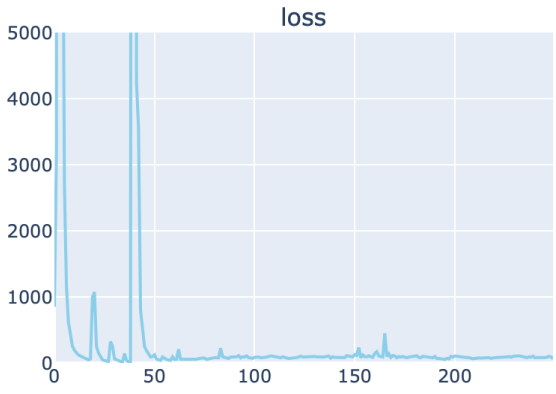
(e) Double 100-Neuron

Figure 5.20: Reward Results of 2 Hidden Layers Models

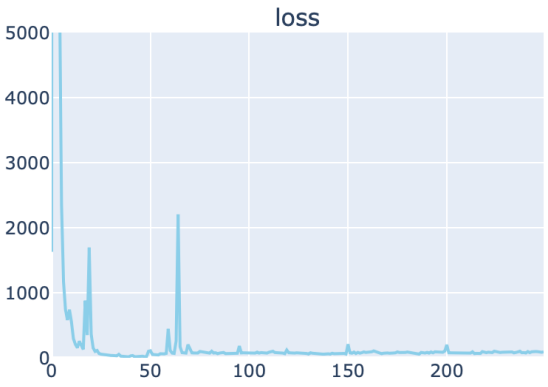
After obtaining the best 2 hidden layers models following the reward chart, it is time to analyse their loss graphs, to finally decide the model which is most optimized. The three loss charts are printed in Figure 5.21. From them, it is easy to conclude with different explanations.

The first one is that both double-15-neuron and double-50-neuron have a very similar loss chart. It can be seen that the double-50-neuron case is worse than the first one, because it has more and bigger peaks than the other. The best loss graph corresponds to the double-25-neuron model, in which losses are very low.

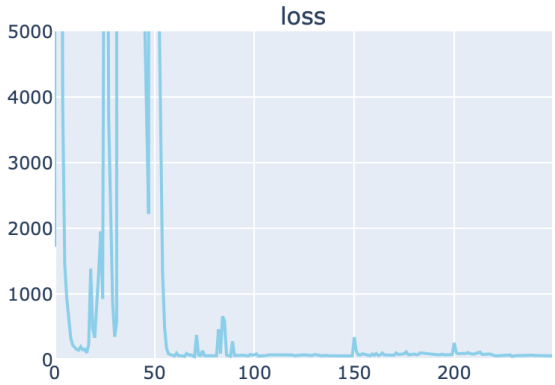
But it is not only the loss comparison. It is needed to mix this loss conclusion with the reward analysis. And the final observation is that the last model is more near to the 500 reward value than the others, and it is also maintaining it with less fluctuation. It is critical to decide that **the double-50-neuron layers model is the one chosen to be the model number two.**



(a) Double 15-Neuron



(b) Double 25-Neuron



(c) Double 50-Neuron

Figure 5.21: Loss Results of 2 Hidden Layers Models

Last point to underline in this section is that several tests were done with different neuron sizes in the hidden layers of the model structure. The main aspect to highlight is that all these tests do not conclude in any reasonable conclusion. There are not any structure better than the model number two selected during last analysis, formed with a double-50-neuron hidden layers. To expose this conclusion, the Figure 5.22 prints the reward charts obtained during the process.

(a) 25&3-Neuron

(b) 50&3-Neuron

(c) 50&25-Neuron

(d) 25&50-Neuron

Figure 5.22: Reward Results of 2 Hidden Layers Models with different neuron sizes

It is easy to see the bad adaptation of the model into the market problem, probably because to a non-fitting training between layers. It can be exclude from this analysis the 50&25-neuron layers model, which has a very great reward graph that highlights the adaptation of this model to reach the 400 value continuously. The Figure 5.23 shows the losses of this structure, that has a good behavior, similar to the best model chosen during this section. The problem that it has is that it does not improve the best result.

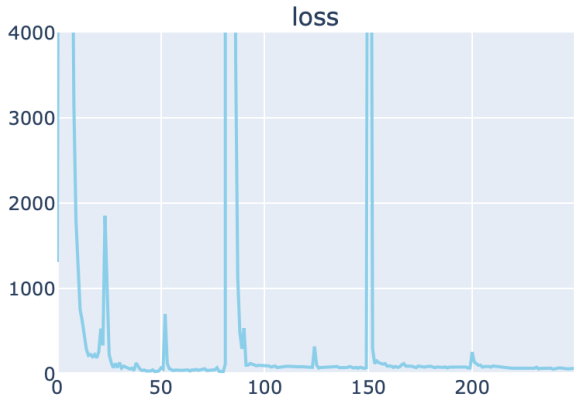


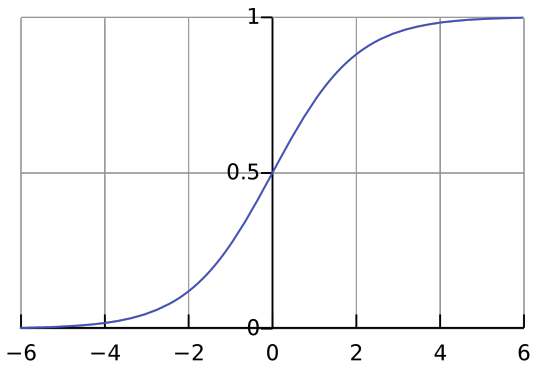
Figure 5.23: Loss Result obtained from 50&25-Neuron Hidden Layers Model

5.4.4 Model 3: Optimization of M1

This point consists on improving the one-hidden-layer neuron model obtained in last sections. This improvement task is made to obtain the best results possible. To reach it, the algorithm is trained with different hyperparameters available in the Chainer library. Different reward and loss charts are obtained from this step, to finally analyse it visually and extract conclusions from them.

There are several hyperparameters available in the library, but for this development the optimizing task is done with the hyperparameters seen below:

- **Sigmoids:** this function [75] has a characteristic "S"-shaped curve, also called sigmoid curve, and are also common in statistics as cumulative distribution functions. Inside this group, there are *Sigmoid* and *Hard Sigmoid* hyperparameters.



(a) Sigmoid Function

$$f(x) = (1 + \exp(-x))^{-1}$$

(b) Sigmoid Formula

$$f(x) = \begin{cases} 0, \text{ if } x \leq -2.5 \\ 0.2x + 0.5, \text{ if } -2.5 < x \leq 2.5 \\ 1, \text{ if } x > 2.5 \end{cases}$$

(c) Hard Sigmoid Formula

Figure 5.24: Sigmoid Hyperparameters

- **Linear Unit Functions:** [9] inside this group, there are five principal functions used, which differ not much from each other.

$$f(x) = \begin{cases} x, \text{ if } x \geq 0 \\ \alpha(exp(x) - 1), \text{if } x < 0 \end{cases}$$

(a) Elu

$$f(x) = \lambda \begin{cases} x, \text{ if } x \geq 0 \\ \alpha(exp(x) - 1), \text{if } x < 0 \end{cases}$$

(b) Selu

$$f(x) = max(0, x)$$

(c) Relu

$$f(x) = \begin{cases} x, \text{ if } x \geq 0 \\ ax, \text{if } x < 0 \end{cases}$$

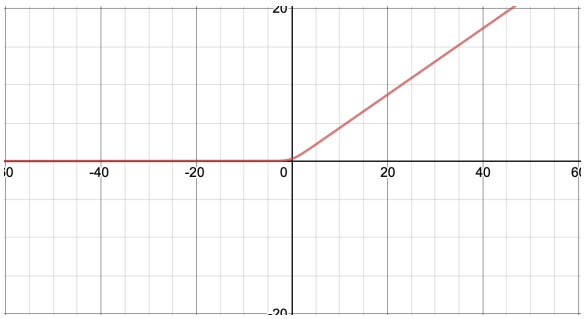
(d) Leaky Relu

$$ClippedReLU(x, z) = min(max(0, x), z)$$

(e) Clipped Relu

Figure 5.25: Linear Unit Function Hyperparameters

- **Softplus:** [9] this function is a smooth aproximation of relu function, and it has the formula seen in Figure 5.26:



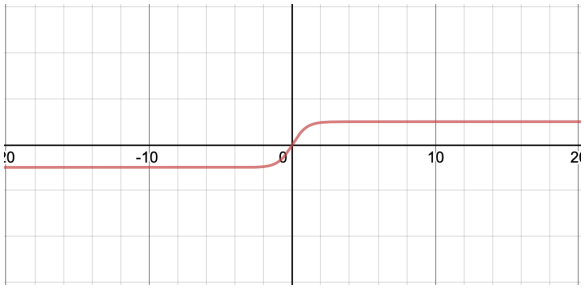
(a) Softplus Chart

$$f(x) = \frac{1}{\beta} log(1 + exp(\beta x))$$

(b) Softplus Formula

Figure 5.26: Softplus Hyperparameter

- **Tanh:** [59] is defined as the division between the hyperbolic sine and the hyperbolic cosine, seen in Figure 5.27:



(a) Tanh Chart

$$f(x) = tanh(x)$$

(b) Tanh Formula

Figure 5.27: Tanh Hyperparameter

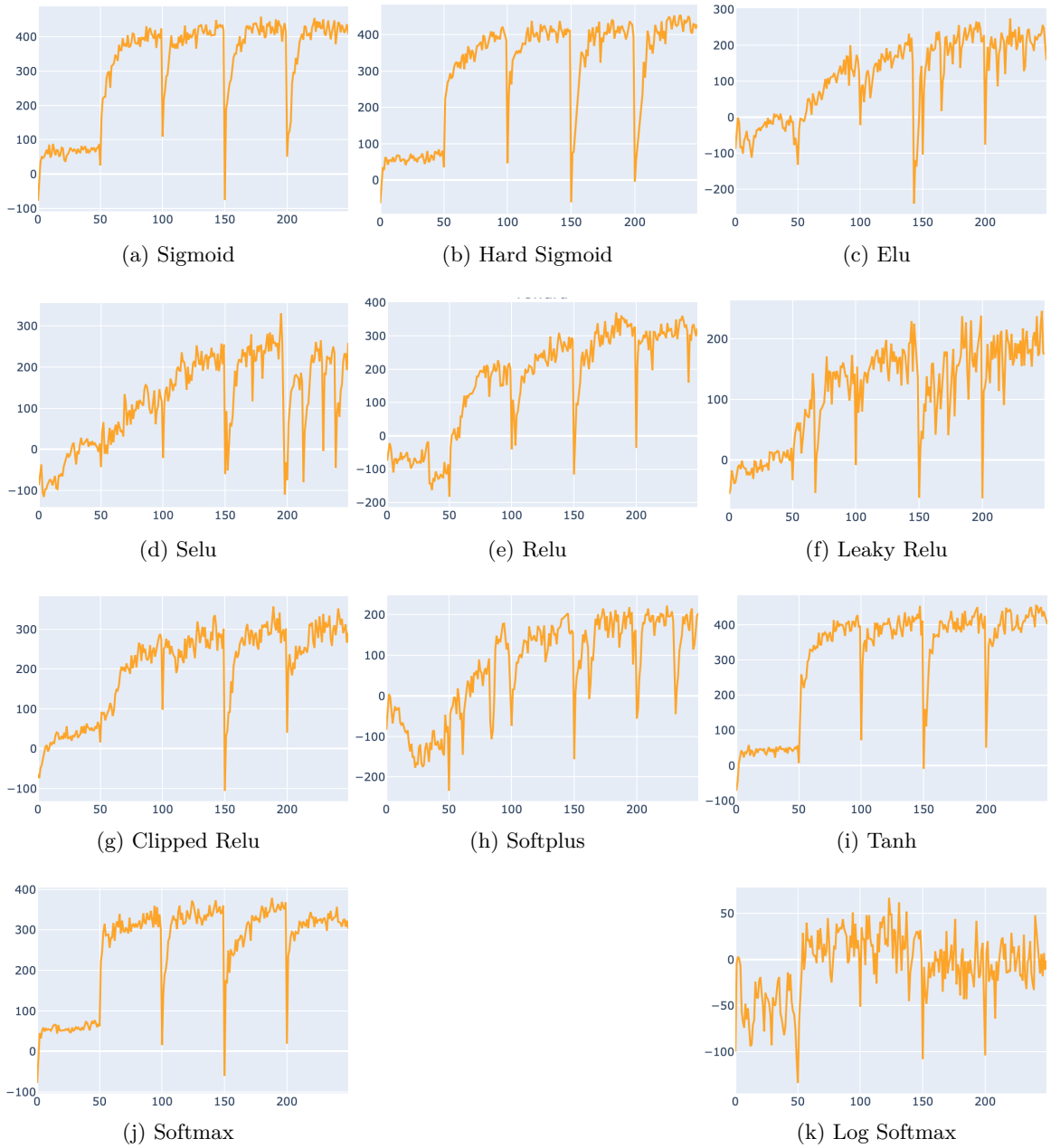
- **Softmax:** [76] also known as softargmax or normalized exponential function, is a function that applies the standard exponential function to each element of the input vector, and normalizes these values by dividing by the sum of all these exponentials. This normalization ensures that the sum of the components of the output vector is 1. Chainer library has a softmax function argument, and also a log softmax function, which is the softmax logarithm.

$$f(x) = \frac{exp(c)}{\sum_d exp(c_d)}$$

Figure 5.28: Softmax formula

After explaining all the hyerparameters used during this development process, the Figures 5.29 and 5.30 show a behavior comparison between hyperparameter configurations in

the best one-hidden-layer model obtained in last sections:



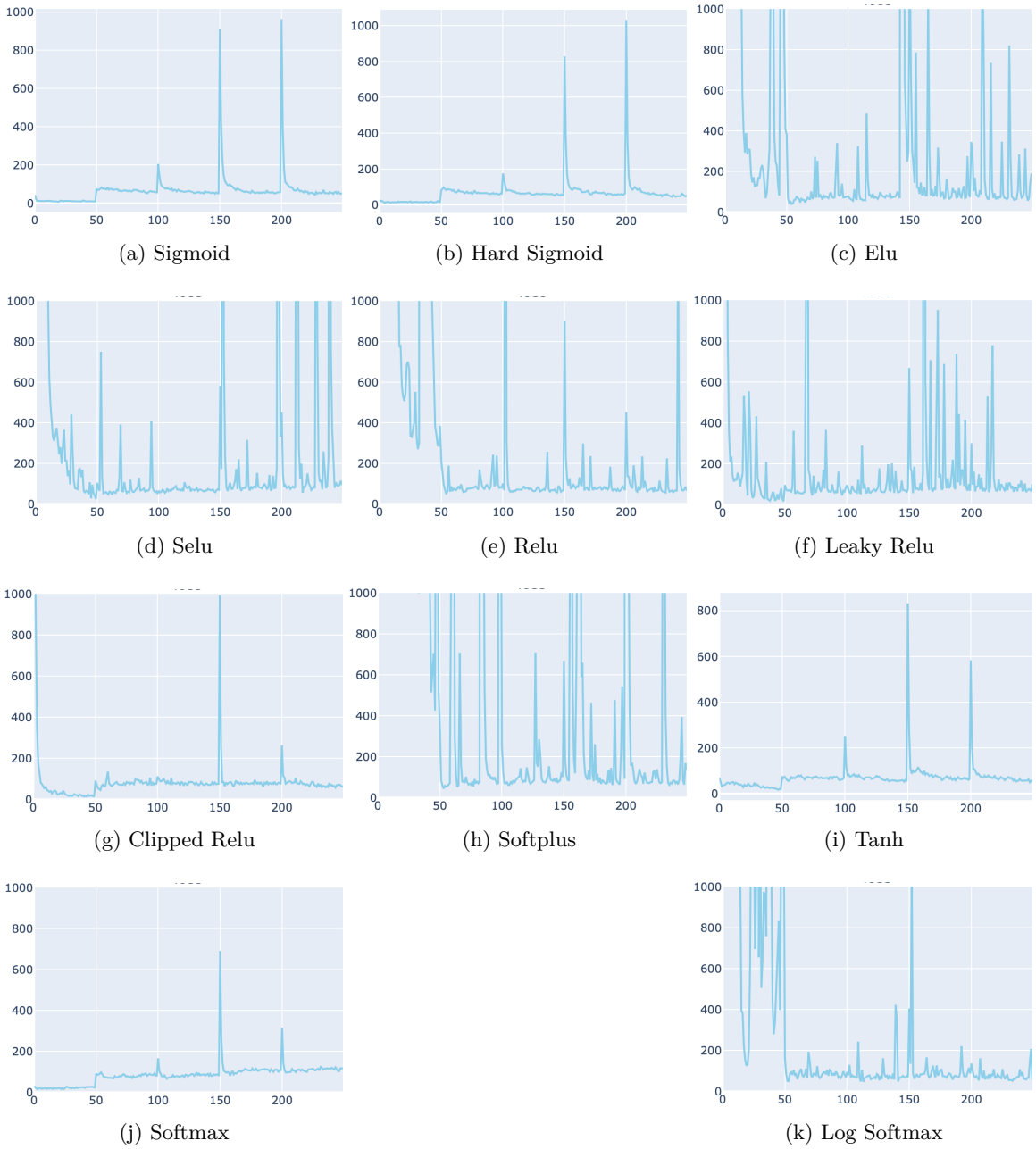
(a) Sigmoid (b) Hard Sigmoid (c) Elu

(d) Selu (e) Relu (f) Leaky Relu

(g) Clipped Relu (h) Softplus (i) Tanh

(j) Softmax (k) Log Softmax

Figure 5.29: Model 1 Hyperparater Optimization - Reward Charts



(a) Sigmoid (b) Hard Sigmoid (c) Elu

(d) Selu (e) Relu (f) Leaky Relu

(g) Clipped Relu (h) Softplus (i) Tanh

(j) Softmax (k) Log Softmax

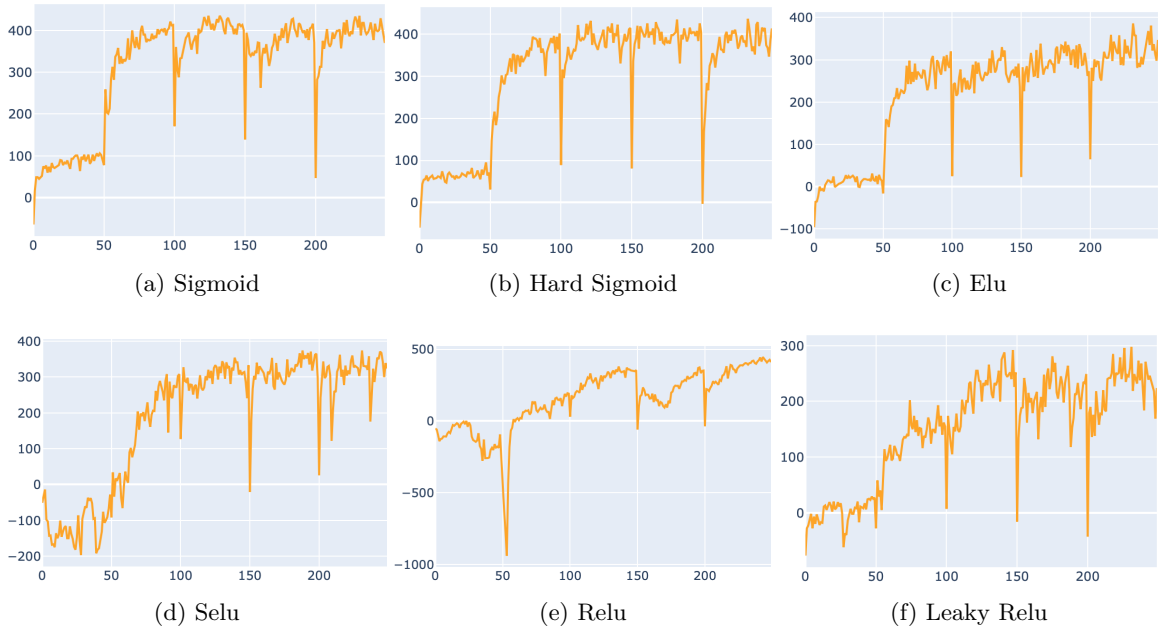
Figure 5.30: Model 1 Hyperparater Optimization - Loss Charts

From last comparisons, it is easy to extract that there are hyperparamerters which are not well adapted to the market problem, like selu, leaky relu, softplus and log softmax. These ones are discarded to be the best optimization model. Into the remaining hyperparameters, there are three that reach a 400 reward value, and also have a constant trend. The best ones are the sigmoid, hard sigmoid and tanh hyperparameters. For these three models, a loss comparison is done to obtain the best one.

In the loss point, the three options have a similar loss behavior, seen only a few peaks in the charts and having an stable value around 100. Going into details, tanh model has lower peaks than the sigmoid models, because tanh loss peaks are below 800 units, and in the sigmoid models these values are around 900 or 1000. Because of that conclusion, and the good response of the reward chart obtained, **tanh model is the best model chosen to do several test during the next sections.**

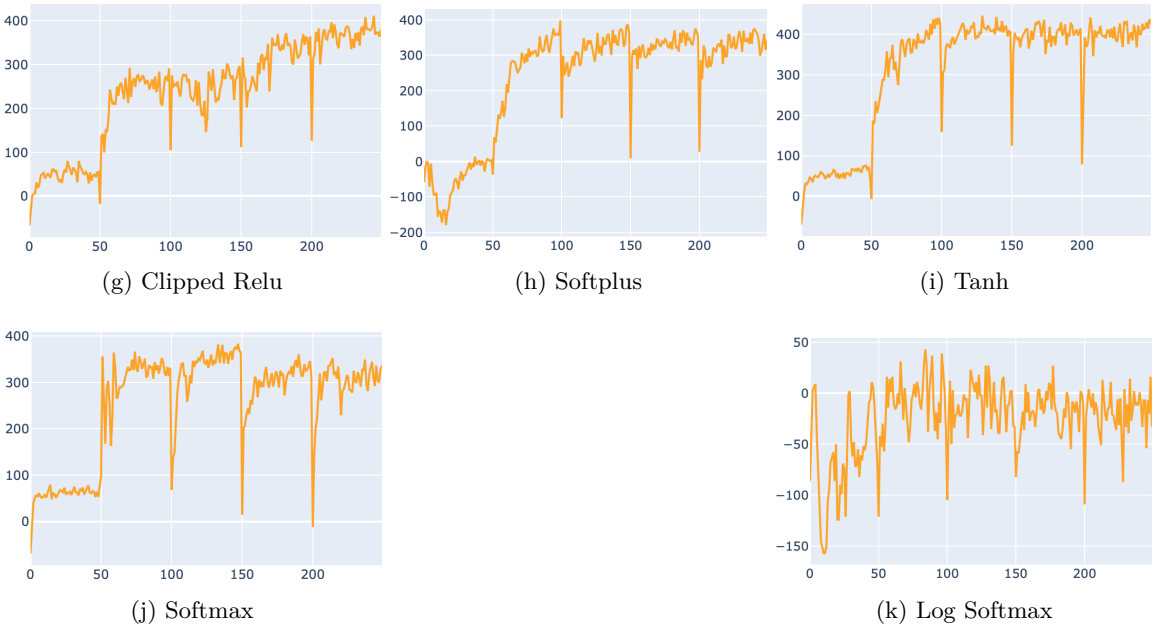
5.4.5 Model 4: Optimization of M2

The process to select the best optimized model in this section is the same process seen before: comparing reward and loss charts to conclude with the best two-hidden-layers model. Seeing reward graphs, softmax and log softmax models have a non-constant behavior, specially in the log one which is in negative values during most of its training time. There are other models which have a better performance, but it is not the best one that can be obtained, like all the linear unit functions. The models with the best reward behavior are the sigmoid ones, softplus and tanh.



(a) Sigmoid (b) Hard Sigmoid (c) Elu

(d) Selu (e) Relu (f) Leaky Relu

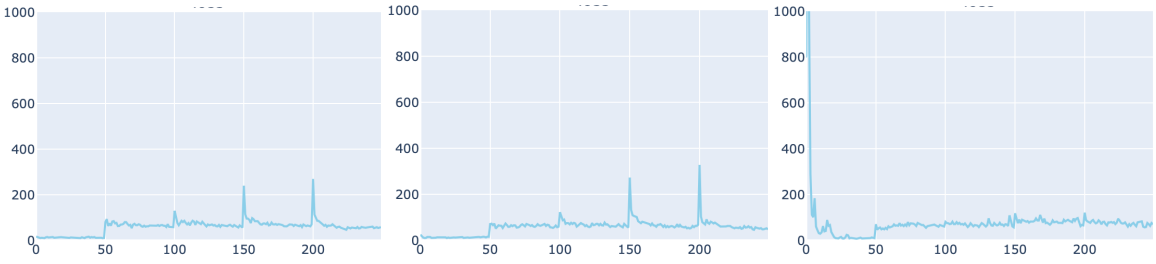


(g) Clipped Relu (h) Softplus (i) Tanh

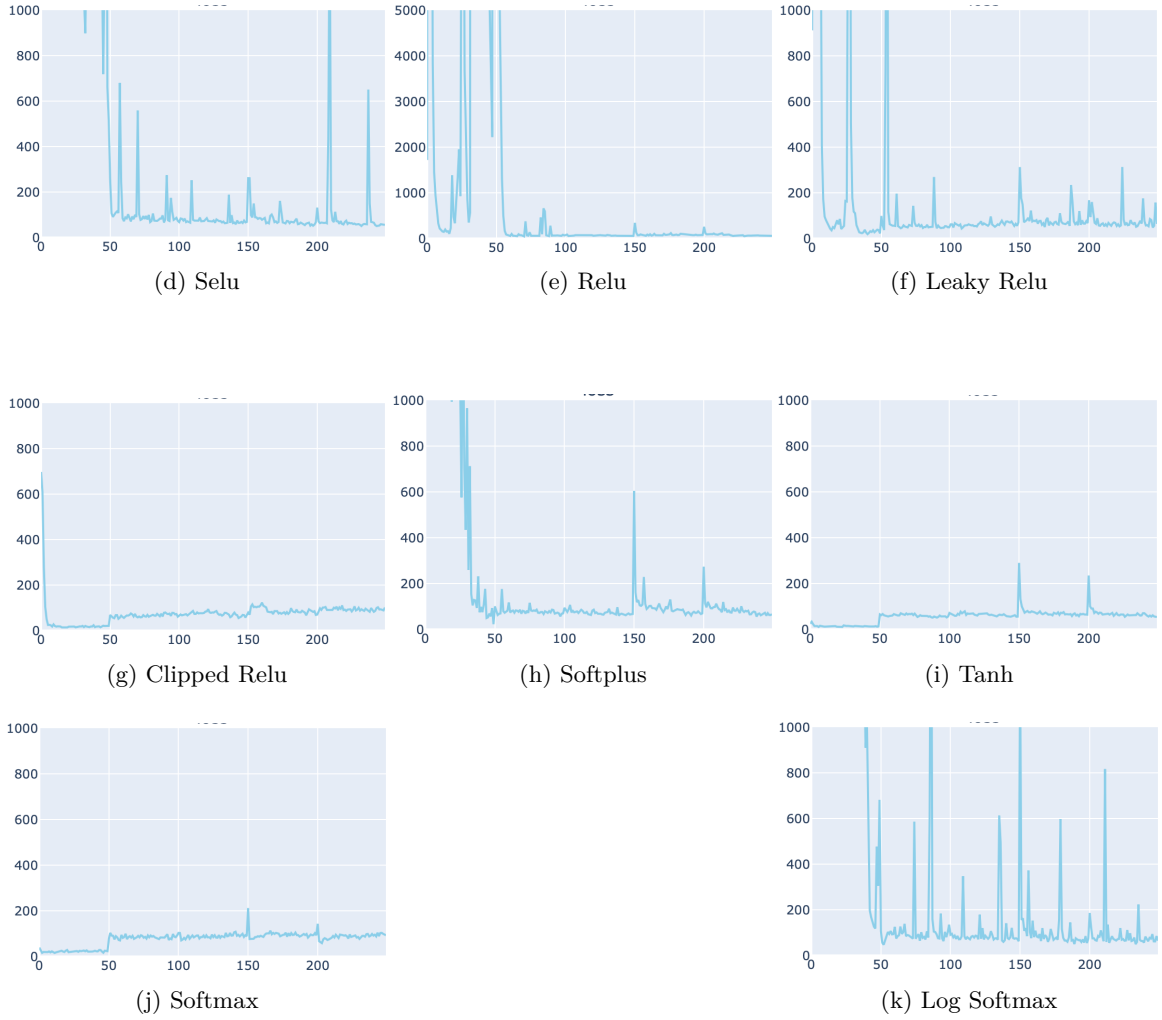
(j) Softmax (k) Log Softmax

Figure 5.31: Model 2 Hyperparater Optimization - Reward Charts

Comparing the loss charts of best models, it is seen that softplus has a strange behavior in its first training periods, having several high peaks that do not transmit high confidence. Then, the sigmoids and the tanh functions are much more stable than the softplus one. Specifically, sigmoid function seems like the model with fewer losses. **For the model number 4, sigmoid model is chosen to represent the double-hidden-layer algorithm**, because of its minimum losses added to the constant and high reward performance.



(a) Sigmoid (b) Hard Sigmoid (c) Elu



(d) Selu (e) Relu (f) Leaky Relu

(g) Clipped Relu (h) Softplus (i) Tanh

(j) Softmax (k) Log Softmax

Figure 5.32: Model 2 Hyperparater Optimization - Reward Charts

5.4.6 Model Comparison

Finally, to determine what algorithms are the best ones, a result collection is exposed in this section. All the test datasets are passed to the models, with the objective of obtaining the different testing values obtained on each model. During last sections, different models are obtained from evaluating different neuron sizes and hyperparameters. It is important to underline the different hidden layer structures of every model reached, to understand better the comparison done:

- **Model 0a:** one-neuron hidden layer.
- **Model 0b:** double-one-neuron hidden layers.
- **Model 1:** 50-neuron hidden layer.

- **Model 2:** double-50-neuron hidden layers.
- **Model 3:** model 1 optimized with tanh hyperparameter.
- **Model 4:** model 2 optimized with sigmoid hyperparameter.

There are different conclusions that can be extracted from the analysis realized in Table 5.3. Firstly, it is easy to see the null result of models 0a and 0b, in reward and profit values. This result is caused by the small neuron structure that models 0 have. They are incapable of taking any decision with their low neurons quantity. Following, models 1 and 2 can process market variations and they take decisions from them. Model 1 is more efficient than model 2, because profit values in model 1 are much better than in the other model. Even if the good reward values obtained, model 2 makes losses in all trend cases, so it is not a reliable algorithm. Finally, models 3 and 4 are the only ones optimized with hyperparameters. Then, model 3 obtains better results than the model 4, having a very high reward and profit values. The conclusion is that **the 50-neuron hidden layer model optimized with tanh hyperparameter is the best deep neural network with reinformcement lerning obtained**, followed by the double-50-neuron hidden layers model optimized with sigmoid hyperparameter.

Trend	0a		0b		1		2		**3**		4	
	R	P	R	P	R	P	R	P	R	P	R	P
Bullish	0	0	0	0	173	168	260	-230	**319**	**560**	209	394
Bearish	0	0	0	0	15	7	97	-116	**94**	**163**	87	-4
Lateral	0	0	0	0	88	100	235	-20	**268**	**375**	258	127

Table 5.3: Market tests comparison *(R=reward, P=profit)*

5.5 Testing Analysis

With the two best models obtained during the testing comparison done (models 3 and 4), it is the moment to check which one has a better performance in different market situations. This testing section checks behaviors in different use cases, using different trends obtained from dataset companies, all of them explained during the dataset section of this chapter. Between all trends available in stock markets, there are three main trends that are studied

during this chapter: bullish trends, bearish trends and lateral trends. Models 3 and 4 are the ones which are tested with those price datasets extracted, showing their behavior into several price graphs, and also collecting all the results obtained in tables.

5.5.1 Bullish Trend

A bullish market is the one that has an increasing stock price values. This tendency is easy seen when, in a chart, the maximum and minimum peaks of the stock are growing, and a trend line joining all these peaks can be drawn. That is the main tendency of the American stock market during the last decades, increasing continuously during all the XXI century, thanks to most of the American companies that have got an increasing economic balance and a good growth with the past of the years. In the Figure 5.33 an example of a bull trend is represented, with its highs, lows and consolidations or corrections.

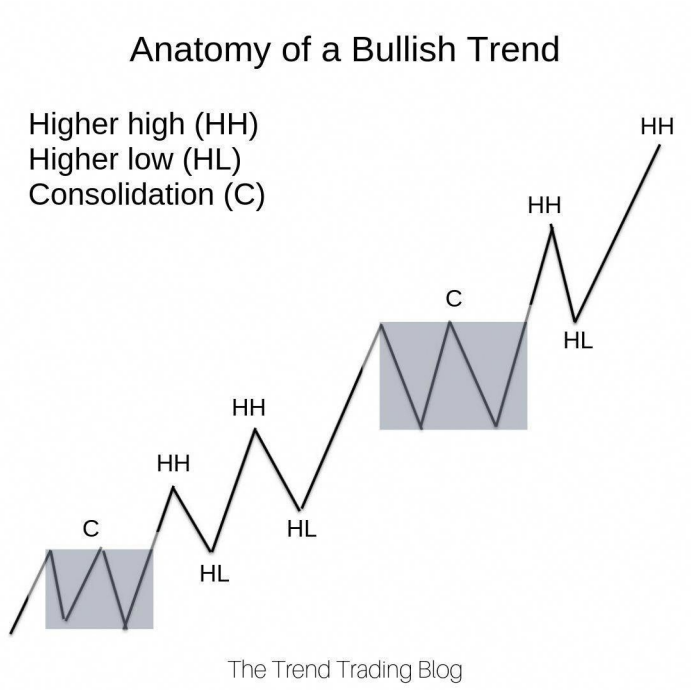
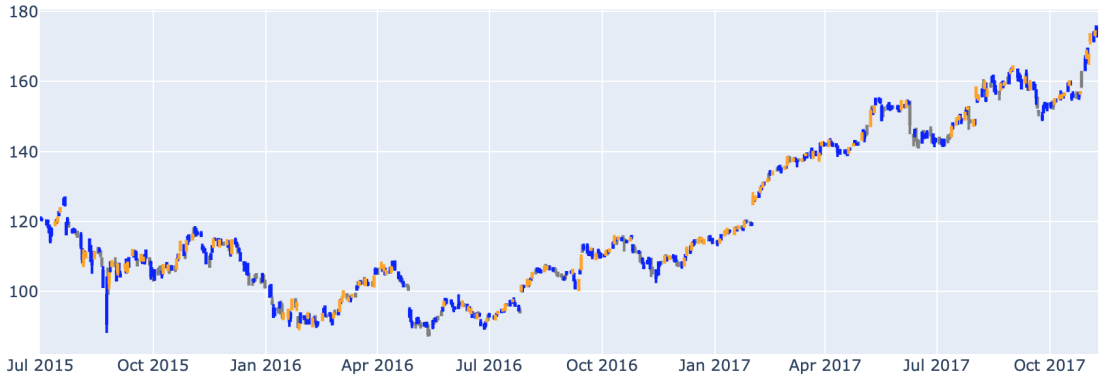


Figure 5.33: General representation of a bulish trend [35]

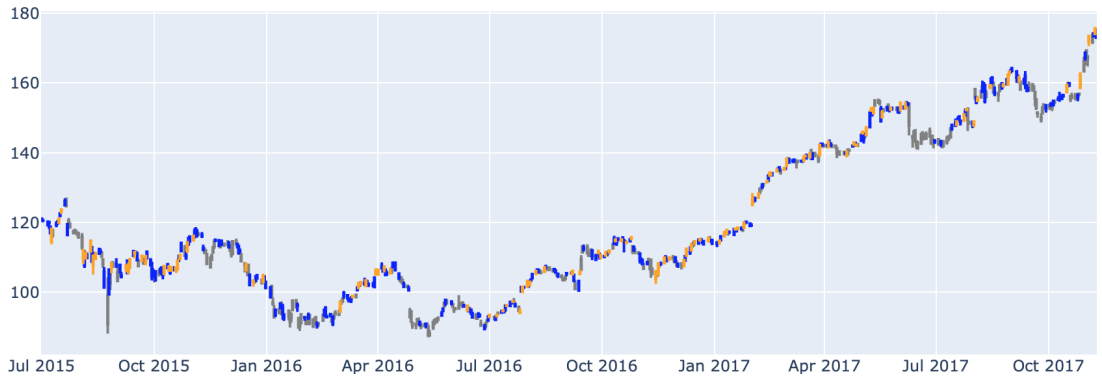
For this trend, all nine companies described in the beginning of the chapter are used to test the two models. An easy task was to detect bull trends into their price graphs, because most of them were generally in this tendency. An exception of this rule is *Advanced Micro Devices* (AMD), which is mostly contained into a lateral trend, and it was more difficult to find a bullish trend to test the algorithms.

Different charts are inserted to compare both intelligence algorithms in some situations. Inside this charts, candlesticks are printed to symbolize the daily price variation. These candlesticks are printed with different colors: gray, blue and orange. That colors corresponds with the daily action done by the deep learning intelligence: stay, buy or sell shares actions respectively.

It can be seen in Figure 5.34 that constant trends with little corrections are the best scenario for these algorithms. In this case, both algorithms have a similar behavior, buying when the share is into the bull trend, and selling in different points considered with high risk. They also buy shares when the prices go down to weigh the price, and have a mean price slower than in the beginning.



(a) Model 3

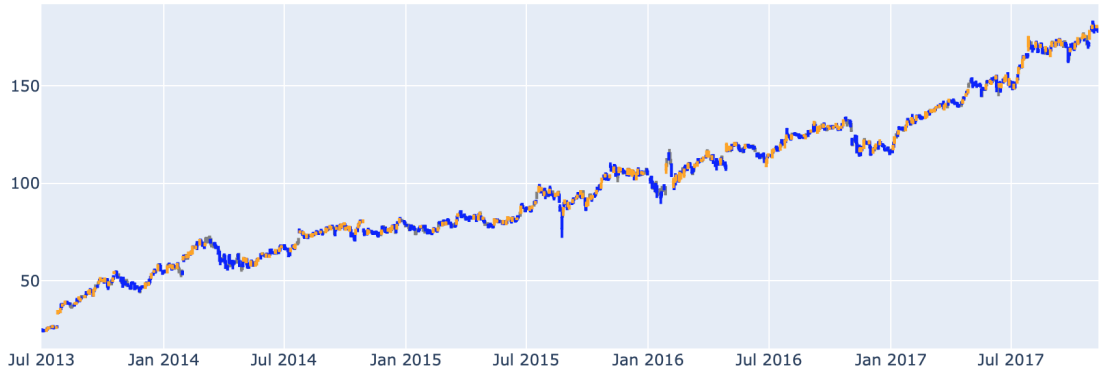


(b) Model 4

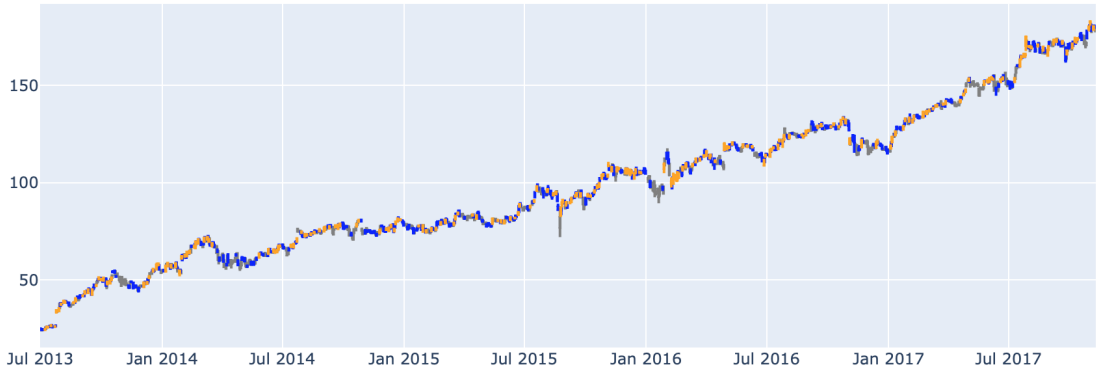
Figure 5.34: Testing with Apple datasets

Another example is with Facebook datasets in Figure 5.38, in which the models differs around 50 reward points. It is caused by a overbuying made by the one-hidden-layer algorithm, which is rewarded with more profit compared with the other model. On the other hand, the two-hidden-layers algorithm tries to stay in doubt situation, which is less risky

than the model 3 behavior. Model 4 could be chosen for people more conservative, and model 3 for people who likes to risk more money in that operations.



(a) Model 3



(b) Model 4

Figure 5.35: Testing with Facebook datasets

The Table 5.4 shows all the results obtained by both algorithms during the bullish training. It covers the reward and the profit obtained in each stock dataset passed, and it is a good comparison to analyze which has a globally better behavior. Important to underline that it is not correct to compare parameters of different shares, because reward and profit values depend on the amount of samples in each dataset, and this number is not the same in all the datasets. In additio, profit quantity depends on the share price, so it is more probable to get high values in shares with higher prices, like Amazon.

Stock	Model 3		Model 4	
	Reward	Profit	Reward	Profit
Apple	109	64	79	20
Microsoft	455	463	413	294
Coca-Cola	429	279	417	177
Walmart	327	355	301	116
JPMorgan	302	313	261	265
Amazon	53	2635	108	1966
Facebook	222	480	175	385
AMD	129	15	131	40
PG	841	433	753	286

Table 5.4: Bullish testing comparison

5.5.2 Bearish Trend

A bearish market is the opposite one of the bullish market. It has decreasing price values, and they can be seen easily in the decreasing peaks that these type of charts have. Their higher and lower peaks can be joined with lines respectively, and they form a decline channel in the chart that symbolizes this tendency. All this concepts can be seen in Figure 5.36, in which can be seen that the end of the trend is produced when the last lower high peak level is passed.

It is very difficult to find a bearish trend in the different stocks that the American market has. That is because the American market is contained into a bullish trend since several decades ago, and most of the principal stocks follows or imitates that tendency. In addition, during this project the share datasets selected are based in big companies with low volatility, which makes much harder the task. For example, bearish trends were found only in Microsoft, Coca-Cola, JPMorgan and AMD shares. So it means that there was no bearish trend in shares like Apple, Walmart or Amazon.

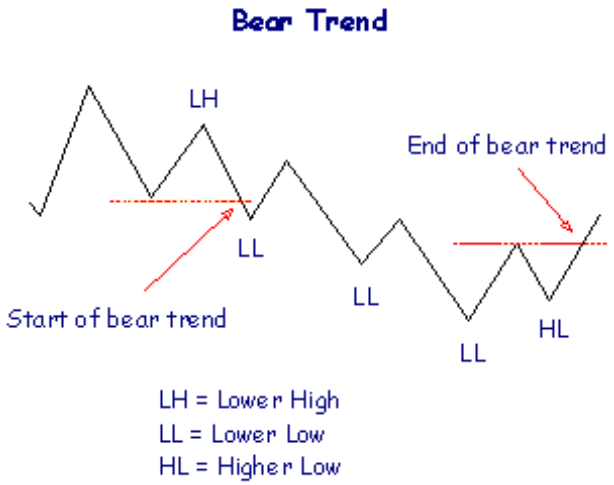
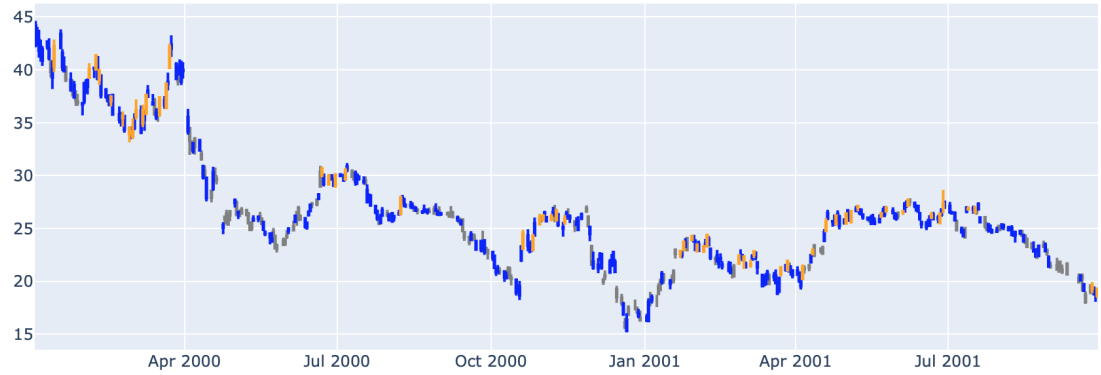


Figure 5.36: General representation of a bearish trend [31]

To finally obtain this bearish trend in some examples, different stocks were examined with Tradingview web tool to find a bearish stock chart portion, and then cut it with the dataset tools. Important to underline that short bear trend, that were around 6 or less months, are not chosen. That is because in most of the cases, these trends define a correction period that the bullish market has.

Seen the final results in Table 5.5, it is interesting to analyze the behavior of both models with Microsoft share. Both models have low reward and profit values. But with double-hidden-layer model or model 4, it is the first case in which this algorithm has negative profit. If both chart images are compared, it can be seen that the model 3 applies the technique of buying when the share falls down. It is much more risky than not to buy, but it implies that the mean price of the total position decreases. So it helps when the position is closed with the next mini-bullish correction, because it is possible to take profit from it.

Model 4 applies the "Away From Market" technique: it does not buy more positions, and then the deep learning algorithm sells the position when the price reaches lower values. It implies that the intelligence gets losses from its market operation. It can be said that the simplest technique is the most effective in this bearish trends.

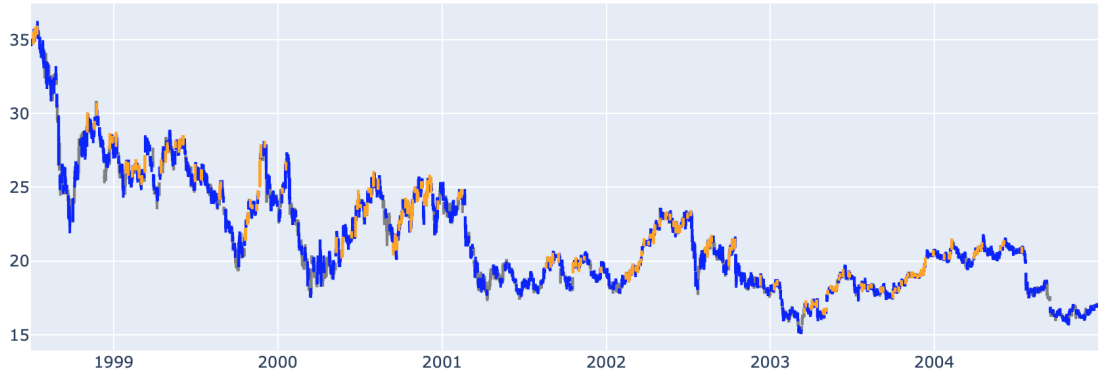


(a) Model 3

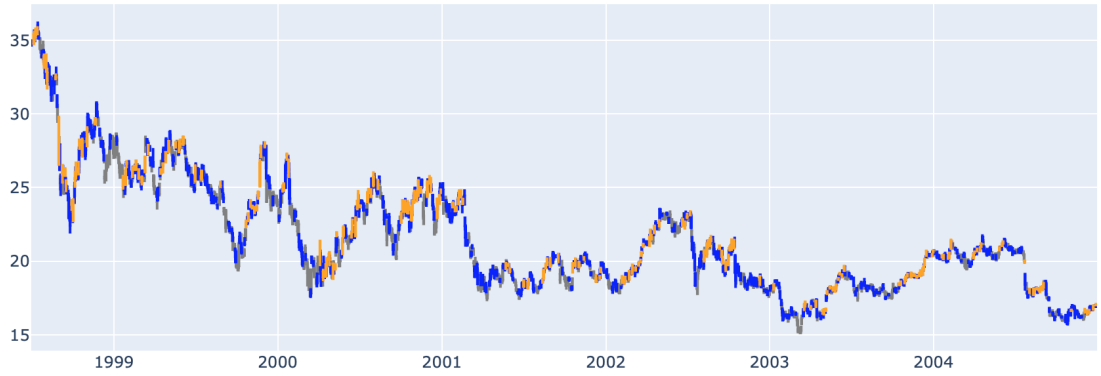
(b) Model 4

Figure 5.37: Bearish trend with Microsoft datasets

The opposite situation happens when the Coca-Cola dataset is taken by the models. They have good reward and profit values, as it can be seen in Table 5.5. That is principally caused by the share price, which is falling down very slow, and this tendency is probably contained into a mix of bear and lateral trend. It allows the models not to have high losses during this period, having also profits on it.



(a) Model 3



(b) Model 4

Figure 5.38: Bearish trend with Coca-Cola dataset

Finally, similar to the bearish case, there is a Table 5.5 which contains the principal values obtained during the testing process of both models with five bearish stock datasets. The profits are much lower than in the bullish case, and it is caused by the model training, based principally in a bull tendency, which was forced by the American market to all the principal company shares.

Stock	Model 3		Model 4	
	Reward	Profit	Reward	Profit
Microsoft	54	29	37	-83
Coca-Cola	183	210	205	81
JPMorgan	99	207	65	8
AMD	43	206	41	-23

Table 5.5: Bullish testing comparison

5.5.3 Lateral Trend

Finally, lateral trend is in the middle of bullish and bearish trends. This tendency is characterized by its price values, that are usually contained between a high and a low price, and the price fluctuates between them. Normally, high and low peaks are the responsible of determine the maximum and minimum levels of this trend. The Figure 5.39 shows an example of this behavior in a share:

Figure 5.39: General representation of a lateral trend [11]

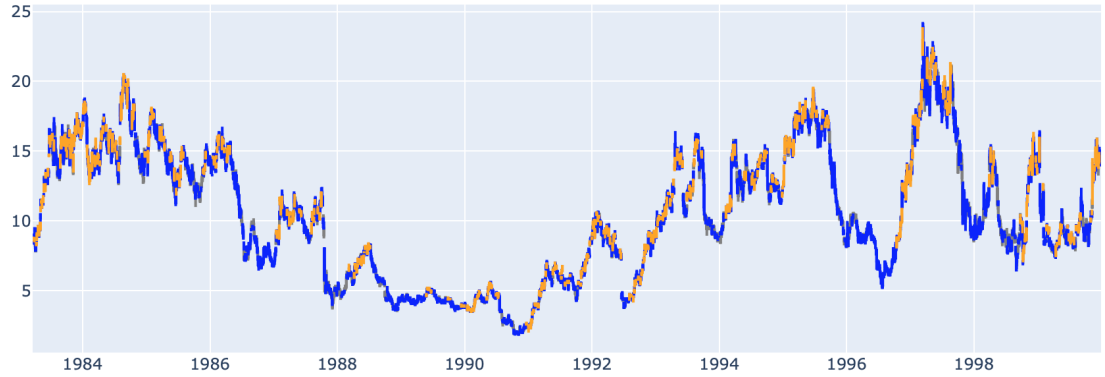
These type of trends has the ability of being formed by bullish and bearish trends inside them, and this is reflected in the results obtained by both models chosen for the testing purpose. The Table 5.6 shows the rewards and final results reached, and they are higher than the bearish case, but lower than the bullish testing. It is seen that there are not all the companies in the table, and this is because there are no lateral tendencies in all the companies shares.

Stock	Model 3		Model 4	
	Reward	Profit	Reward	Profit
Microsoft	288	210	290	111
Walmart	292	526	283	243
JPMorgan	195	347	151	145
Facebook	35	123	41	11
AMD	531	668	524	116

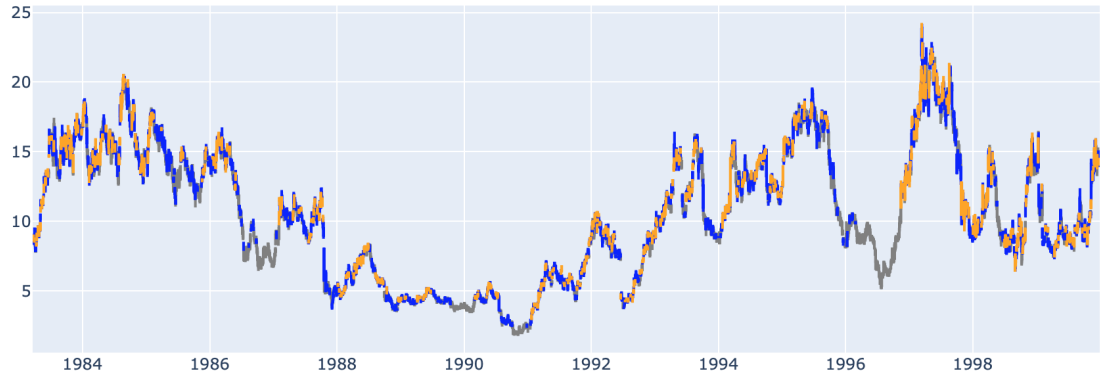
Table 5.6: Bullish testing comparison

To end this section, a lateral behavior example of both models is exposed in Figure 5.40. AMD price dataset is the one chosen for this example, because during last point it was proven that this stock is the less similar to the main index (S&P500). The algorithms applies the techniques explained in bullish and bearish sections: the one-hidden-layer model tries to use more the buy option than the two-hidden-layers model, which prefers to stay out

of the market and not to generate high losses. In this case, model 3 takes more advantage of AMD trends, having much more profit than the model 4 (668 vs 116, respectively).



(a) Model 3



(b) Model 4

Figure 5.40: Lateral trend with AMD dataset

CHAPTER 6

Conclusions and Future Work

On this last chapter, a revision of all points covered during the intelligence algorithm development is done, also added to a list of future improvements that can be done after this project.

6.1 Achieved Goals

This final project consists in the development and application of a deep learning intelligence that automatizes the problem of operating in stock markets. These deep learning technique has been implemented using different Python libraries, being Chainer library the most important one. During this final thesis, different points are reached:

- Firstly, an analysis about the state of art was done, in which different theoretical concepts were described to be applied during the deep learning algorithm development. Furthermore, after this chapter there was an explanation of all the enabling technologies applied and utilized to reach the final objective.
- The beginning of the deep learning development was produced in the moment when the project architecture was define. This architecture consists in a *Deep Neural Network* (DNN) with reinforcement learning methodology, that contains different elements which work in group to reach the behavior desired. Principally, there are three main elements utilized:
  - The dataset of all the American stocks needed to the project. All these companies selected to the analysis have the characteristics of being solvent, and they also have low volatility as time goes by, which facilitate the training and the final analysis of the testing task.
  - The deep learning intelligence used to train and test the particular circumstances, which consists in a DNN with different hidden layers and neuron quantities.
  - The environment applied on this project, which covers the task of being the reinforcement learning of the DNN. It returns the price changes in the stock which is being analyzed, and the reinforcement value derived from the action selected by the deep learning algorithm, all of it with daily frequency.
- When the main architecture and its procedure was defined, the next task was to define the training and the testing process. This training process consists in two different analysis:
  - The comparison and optimization of the different neural structures selected, in which there were different possibilities of neural hidden layers and number of neurons in each layer. This comparison is done with reward and loss values obtain from each model configuration.
  - Then, the next step was to test that structure with different layer hyperparameters. From this analysis, different models were obtained and matched to obtain

the most optimum structure combination.

- Finally, the two best deep learning algorithms were applied into some test datasets with different trends, to join and compare those results obtained with some theoretical conclusions.

6.2 Conclusions

Thanks to the different optimization process applied and the several market stock variants analyzed during the length of that memory, there are different conclusions that can be exposed. This results are also connected with all theoretical points explained during the thesis memory, to understand better every behavior made by the DNN.

The first point is related to the neural optimization task. It is easy to conclude that the size of the first hidden layer usually determines the size of the second hidden layer. Any divergence between them can complicate the training task. For example, 50&25-neuron hidden layers model has a good behavior during the training step. But with less neurons, the performance is worse. Other double hidden layer models have the same size in both hidden layers, and it usually makes a great behavior, but only if the neuron quantity is sufficiently big. Another interesting point is that there are no linear relation between the neural size and the results: having a big neuron quantity in each layer does not imply a better performance. In addition, the optimum structure is reached with not a big number of neurons in the layer combination.

If the final tests realized are analyzed in depth, the main conclusion obtained from them is that a DNN is very influenced with the principal market trend of the dataset. It is very difficult to have an algorithm which operates efficiently in every trend (bullish, bearish and lateral). That is because each trend has its own methodology to apply, related with entrance and exit market points, and different candlesticks patterns. For example, if a deep learning intelligence trained in a bullish trend tries to operate in a bearish trend, it can be seen during last chapter that the algorithm tries to wait to sell until the market goes up. But in a bearish trend, most of the time the price goes down, so it is very risky to apply this operation model.

On the other hand, it is seen that the reinforcement learning added to a deep learning intelligence makes a great combination to elaborate an algorithm that operates in the stock market. Reinforcement model is more adapted to this kind of problems than prediction or

classification models. This theory point is because it is important to receive some feedback of the environment which is operated, in this case the American market, according to the decisions taken as the time goes by.

6.3 Future lines of work

Some improvement points of the algorithm are exposed below to guide future tasks with the main goal of making the behavior of the development more successful:

- Improve the reinforcement learning process using **technical indicators**. There are different indicators (like Bollinger lines) that can predict when the share can turn its trend. In addition, there is the probability of having several behaviors of the same indicator which can report future trends. For example, when different Simple Moving Average lines with different frequency (for example, 200 and 50 periods) are crossed each other, it could be an indicator of the beginning of a trend. This behaviors could be implemented to improve the results.
- **Using the transaction volume** to confirm different behaviors. It is known that volume indicator is one of the most famous in technical market analysis. With this information, the trader can evaluate if the price movement is confirmed with volume, or if it is not supported by any other indicator.
- **Analyzing automatically the support and resistance lines of the graph.** These are lines in which the prices has some difficulties to pass through, and are the most important indicators in technical analysis, because it reflects the psychological actions of the traders and investors.
- **Combine technical analysis with fundamental analysis** (balance sheets, cash flow...), to determine the health of the company analyzed.

APPENDIX A

Project Impact

This appendix express the social, economic, environmental and ethical impact that the project has, reasoning each section.

A.1 Introduction

This final thesis is covered into the intelligent systems department (also called GSI) of the Polytechnical University of Madrid (UPM), and is in charge of elaborating a tool to automatize the trading task, and improving it in the best way that is possible. The project has its origin in a self idea of the owner of this document, with the main goal of analyze the viability of these ideas combined with machine learning algorithms.

Following with this chapter, the next sections will cover all the relevant points about the analysis of this solution for diverse external factors.

A.2 Social Impact

The principal social impact that this project has is the idea of facilitating the trading task to any person who wants to introduce his savings into this type of financial option. It is an alternative of putting money on banks products, that usually have null transparency for the customer.

The deep learning intelligence wants to facilitate to the users the task of investigating and operating in different stock companies, using its high processing capacity. It return different profits because of optimizing their money circulation.

A.3 Economic Impact

As it was said in last section, the economic impact that this tool could have to its users is very high. All people who had their money without movement in their banks are losing around 2% of their money value because of the inflation (that is the mean inflation value in the world). And the most important point is that those people do not know it, or they do not want to take action with it.

Facilitating this task, a normal used could be introduced into this investing world, which is really complex (specially at the beginning). The mean objective that is possible to reach is to eliminate the inflation losses that a used can have because of the money devaluation, but it could be possible to maximizing the profits using this algorithm, reaching percentage values around 6 or 10% of the money invested.

It is important to underline that, with this first version of the algorithm, the task has some manual jobs that are needed to automatize (like searching enterprises with an interesting possibility of growing the next years, for example).

A.4 Environmental Impact

Related to the environmental point, there is no an important advantage or disadvantage found during its development. Only to underline the energy consumption that the computer could have during when the deep learning intelligence is working, and the different equipment necessary to turn on this algorithm.

A.5 Ethical Impact

The main ethical implication that this project could have is that it could substitute human jobs if an intelligence reach a high efficiency ratio with its operation. Nowadays, this type of tasks are done by people with a specific work profile (such as economists), which have the most relevant knowledge to perform this activity.

The problem becomes when, in a non-specific future, all the knowledge of this economic ambit is passed to that algorithms, which would have the real and most optimum responses to every market situation.

APPENDIX B

Project Budget

This appendix evaluates the different costs that the project has to be applies, including human resources, physical resources like computers, indirect costs and taxes.

B.1 Physical Resources

To carry out this final project, the most important point is to specify the electronic resources that are needed. Specifically, for this project is necessary to have a powerful computer to the task of training the intelligence, which is the activity that most resources need. During this project, a 13 inch MacBook Pro was used to the train and test processes. To quantify the price of the computer needed, it is necessary to list the characteristics that were needed, in this case, with an Apple laptop:

- **CPU:** Intel i5 processor, double-core.
- **RAM:** 8 GB.
- **Hard Disk:** 256 GB minimum.

To work as the same time as the deep learning algorithm is trained (or tested) without any problem, it could be necessary to upgrade the RAM quantity to 16 GB. Also a CPU upgrade could be a good option to reduce the training and testing times. The price of a laptop with this specifications could reach 800€ or 1000€ in Spain country. It is possible to complement this laptop with a screen to use it as a extended window, and it will cost around 120-150€ (for example, a high quality 21 inch screen of Hewlett-Packard).

B.2 Human Resources

The price of taking on a person to develop this task is very variable. It depends on his experience and his aspirations. It is possible to calculate all the time inverted in this project and make an estimation based on it.

If a Master thesis has a quantity of 30 ECTS (and 1 ECTS corresponds to 30 working hours), the time inverted during this thesis is equivalent to the next operation: 30 ECTS x 30 hours per ECTS = 900 hours of work. Assuming that each month has 22 working days (with 8 hours per day), then the project covers 5,11 months of work (6 months are going to be taken to calculate the final cost, adding more time to finish the debugging of the algorithm).

It is possible to finish this task with one employee in half-year. If the employee is a junior engineer, with a 24000€ annual salary (which is the mean salary for that profile), this project will cost around 12000€ in human resources.

B.3 Indirect Costs

There are indirect costs related to the project, which include electricity and internet connection. If an enterprise covers this project, not all the indirect resources are paid only for this engineer. It can be supposed only a 5% of the total indirect costs.

If the Internet connection used has a 100 Mbps bit rate that costs 30€ per month, and the electricity price in Spain is 0.11854€ per kilowatt, the final indirect costs will depend on the watt consumption of the laptop and its screen. If a laptop generally consumes around 180-200 watts per hour, and the screen consumes 100 watts, the final calculation is the following:

- Internet connection: 5% of 30€ = 1,5€
- Electricity consumption: 5% of (0.11854€ per watt * (200 watts + 100 watts)) = 1,78€

B.4 Taxes

This final section specifies the taxes applied in all expenses of this appendix of the project (in specific with electronic objects, internet and electricity). It is important to underline that these taxes are counted in previous prices exposed, but it is important to specify the tax percentage of each product. Taxes considered are the following ones:

- 21% of price in the laptop, screen, internet connection and electricity.
- 23,6% of gross salary to Spain social security.

Finally, the total price of the main resources of the project are resumed in Table B.1:

Element	Price
Laptop	1000€
Second Screen	150€
Salary (Total)	12000€
Laptop	1000€
Internet connection	1,5€
Electricity consumed	1,78€
Total	14153,28€

Table B.1: Resource prices

Bibliography

[1] Search Enterprise AI. *Supervised Learning*. URL: https://searchenterpriseai.techtarget.com/definition/supervised-learning. (accessed: 7.05.2020).

[2] Anaconda. *Anaconda documentation*. URL: https://docs.anaconda.com/anaconda/navigator/. (accessed: 24.03.2020).

[3] ASPGems. *KAGGLE: Resolver problemas usando casi 200k científicos de datos*. URL: https://aspgems.com/kaggle-resolver-problemas-usando-casi-200k-cientificos-de-datos/. (accessed: 25.03.2020).

[4] Auquan. *Application of Machine Learning Techniques to Trading*. URL: https://medium.com/auquan/https-medium-com-auquan-machine-learning-techniques-trading-b7120cee4f05. (accessed: 2.04.2020).

[5] The GitHub blog. *The State of the Octoverse: machine learning*. URL: https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/. (accessed: 23.03.2020).

[6] Bloomberg. *Trading desk technology*. URL: https://www.bloomberg.com/professional/blog/trading-desks-technology-how-did-we-get-here/. (accessed: 1.04.2020).

[7] Facundo Bre. *Artificial Neural Network architecture*. URL: https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051. (accessed: 3.04.2020).

[8] Steve Burns. *This is the percent gain you need to break even after a loss of a certain percent*. URL: https://twitter.com/SJosephBurns/status/953610005122609153/photo/1. (accessed: 7.05.2020).

[9] Chainer. *Chainer – A flexible framework of neural networks*. URL: https://docs.chainer.org/en/stable/. (accessed: 3.04.2020).

[10] Chainer.org. *Chainer – A flexible framework of neural networks*. URL: https://docs.chainer.org/en/stable/. (accessed: 25.03.2020).

[11] James Chen. *Trading Range Definition*. URL: https://www.investopedia.com/terms/t/tradingrange.asp. (accessed: 31.05.2020).

[12] Novatos Trading Club. *Esquivando los mercados laterales.* URL: https://www.novatostradingclub.com/formacion/esquivando-los-mercados-laterales/. (accessed: 7.05.2020).

[13] El Confidencial. *3.500€/mes por 30 minutos al día: el 'trader' canario de 23 años que triunfa en la red.* URL: https://www.elconfidencial.com/tecnologia/2019-09-16/inversion-etoro-inversos-espanol-mercado_2227143/. (accessed: 5.05.2020).

[14] Fabrice Daniel. *Deep Learning for Forex Trading.* URL: https://medium.com/lusis-ai/deep-learning-for-forex-trading-ba5d466e92cb. (accessed: 6.05.2020).

[15] DeepAI. *What is Unsupervised Learning?* URL: https://deepai.org/machine-learning-glossary-and-terms/unsupervised-learning. (accessed: 7.05.2020).

[16] Desmos. *Desmos - Calculadora Gráfica.* URL: https://www.desmos.com/calculator. (accessed: 10.04.2020).

[17] Diatech. *¿Qué es el Machine Learning, tipos y donde es utilizado?* URL: https://inteligenciaartificial.io/machine-learning/. (accessed: 3.04.2020).

[18] Chainer Doc. *Functions.* URL: https://docs.chainer.org/en/stable/reference/functions.html. (accessed: 10.04.2020).

[19] Trading Economics. *Stock Market Index.* URL: https://tradingeconomics.com/spx:ind. (accessed: 1.04.2020).

[20] Stack Exchange. *What's the principal difference between ANN,RNN,DNN and CNN?* URL: https://datascience.stackexchange.com/questions/58728/whats-the-principal-difference-between-ann-rnn-dnn-and-cnn. (accessed: 13.05.2020).

[21] Yahoo Finance. *IBEX35 - Datos Históricos.* URL: https://es.finance.yahoo.com/quote/%5EIBEX/history?period1=729734400&period2=1588809600&interval=1d&filter=history&frequency=1d. (accessed: 7.05.2020).

[22] Yahoo Finance. *Nikkei - Datos Históricos.* URL: https://es.finance.yahoo.com/quote/%5EN225/history?period1=-157420800&period2=1588809600&interval=1d&filter=history&frequency=1d. (accessed: 7.05.2020).

[23] Thomas Fischer. *Reinforcement learning in financial markets - a survey.* URL: https://www.econstor.eu/bitstream/10419/183139/1/1032172355.pdf. (accessed: 10.05.2020).

[24] Futuro a fondo. *Keynes: "Los mercados pueden mantener su irracionalidad más tiempo del que tú puedes mantener tu solvencia"*. URL: https://www.futuroafondo.com/es/educacion-financiera/SFB18-keynes-mercados-pueden-mantener-su-irracionalidad-mas-tiempo-del-que-tu-puedes. (accessed: 2.04.2020).

[25] The Motley Fool. *Here's How Much Warren Buffett Has Made on Coca-Cola*. URL: https://www.fool.com/investing/2019/11/19/heres-how-much-warren-buffett-has-made-on-coca-col.aspx. (accessed: 5.05.2020).

[26] Mario Fortier. *TA-Lib*. URL: https://ta-lib.org. (accessed: 9.06.2020).

[27] GitHub. *NumPy/SciPy Tutorial*. URL: https://fgnt.github.io/python_crashkurs_doc/include/numpy.html. (accessed: 25.03.2020).

[28] GitHub. *Plotly.py*. URL: https://github.com/plotly/plotly.py. (accessed: 25.03.2020).

[29] IBM. *Recurrent Neural Network*. URL: https://en.wikipedia.org/wiki/Recurrent_neural_network. (accessed: 8.05.2020).

[30] Investing. *Gráfico de índices en tiempo real*. URL: https://es.investing.com/charts/indices-charts. (accessed: 6.05.2020).

[31] Alex Krüger. *Crypto Bear Market*. URL: https://twitter.com/krugermacro/status/1111658219905138688/photo/1. (accessed: 31.05.2020).

[32] Roberto López. *Trader versus investor*. URL: https://www.rankia.com/foros/bolsa/temas/3687797-trader-versus-inversor. (accessed: 6.05.2020).

[33] Sambit Mahapatra. *Why Deep Learning over Traditional Machine Learning?* URL: https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063. (accessed: 13.05.2020).

[34] Boris Marjanovic. *Huge Stock Market Dataset*. URL: https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs#aadr.us.txt. (accessed: 6.04.2020).

[35] Meghan. *The anatomy of a bullish trend*. URL: https://www.pinterest.es/pin/620019073684738009/. (accessed: 31.05.2020).

[36] Cory Mitchell. *How to Use a Moving Average to Buy Stocks*. URL: https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp. (accessed: 19.05.2020).

[37] NumPy. *NumPy doc*. URL: https://numpy.org. (accessed: 23.03.2020).

[38] Openbank. *Roboadvisor 2020*. URL: https://www.openbank.es/inversiones/robo-advisor-gestion-carteras?gclid=EAIaIQobChMIpNHtpYGf6QIVSLTVCh3WEgbTEAAYBwE&gclsrc=aw.ds. (accessed: 6.05.2020).

[39] El Plural. *¿Por qué se ha puesto de moda el trading en los últimos años?* URL: https://www.elplural.com/economia/por-que-se-ha-puesto-de-moda-el-trading-en-los-ultimos-anos_98675102. (accessed: 5.05.2020).

[40] Valeryia Shchutskaya. *Deep Learning: Strengths and Challenges*. URL: https://indatalabs.com/blog/deep-learning-strengths-challenges?cli_action=1589379296.445. (accessed: 13.05.2020).

[41] Slickcharts. *S&P 500 companies*. URL: https://www.slickcharts.com/sp500. (accessed: 6.05.2020).

[42] Techopedia. *Reinforcement Learning (RL)*. URL: https://www.techopedia.com/definition/32055/reinforcement-learning-rl. (accessed: 7.05.2020).

[43] Tomiwa. *How My Machine Learning Trading Algorithm Outperformed the SP500 For 10 Years*. URL: https://towardsdatascience.com/the-austrian-quant-my-machine-learning-trading-algorithm-outperformed-the-sp500-for-10-years-bf7ee1d6a235. (accessed: 6.05.2020).

[44] Tradingview. *Tradingview website*. URL: https://es.tradingview.com. (accessed: 25.03.2020).

[45] Cegos Online University. *¿Por qué aumenta la demanda de formación online?* URL: https://www.cegosonlineuniversity.com/por-que-aumenta-la-demanda-de-formacion-online/. (accessed: 5.05.2020).

[46] Unknown. *Candle Chart Explained*. URL: http://hofac.appscounab.co/candle-chart-explained/. (accessed: 6.04.2020).

[47] Wikipedia. *Advanced Micro Devices*. URL: https://en.wikipedia.org/wiki/Advanced_Micro_Devices. (accessed: 20.04.2020).

[48] Wikipedia. *Amazon (company)*. URL: https://en.wikipedia.org/wiki/Amazon_(company). (accessed: 20.04.2020).

[49] Wikipedia. *Apple*. URL: https://en.wikipedia.org/wiki/Apple_Inc.. (accessed: 7.04.2020).

[50] Wikipedia. *Artificial Neural Network*. URL: https://en.wikipedia.org/wiki/Artificial_neural_network. (accessed: 8.05.2020).

[51] Wikipedia. *CUDA*. URL: https://es.wikipedia.org/wiki/CUDA. (accessed: 25.03.2020).

[52] Wikipedia. *Deep Belief Network*. URL: https://en.wikipedia.org/wiki/Deep_belief_network. (accessed: 8.05.2020).

[53] Wikipedia. *Deep learning*. URL: https://en.wikipedia.org/wiki/Deep_learning. (accessed: 3.04.2020).

[54] Wikipedia. *Deep Learning Architectures*. URL: https://developer.ibm.com/technologies/artificial-intelligence/articles/cc-machine-learning-deep-learning-architectures/. (accessed: 11.05.2020).

[55] Wikipedia. *Deep Neural Network - Deep Learning*. URL: https://en.wikipedia.org/wiki/Deep_learning#Deep_neural_networks. (accessed: 8.05.2020).

[56] Wikipedia. *Dow Theory*. URL: https://en.wikipedia.org/wiki/Dow_theory. (accessed: 6.05.2020).

[57] Wikipedia. *Facebook*. URL: https://en.wikipedia.org/wiki/Facebook,_Inc.. (accessed: 20.04.2020).

[58] Wikipedia. *Forex Exchange Market*. URL: https://en.wikipedia.org/wiki/Foreign_exchange_market. (accessed: 7.05.2020).

[59] Wikipedia. *Hyperbolic Tangent Function*. URL: https://es.wikipedia.org/wiki/Tangente_hiperb%C3%B3lica. (accessed: 31.05.2020).

[60] Wikipedia. *JPMorgan Chase*. URL: https://en.wikipedia.org/wiki/JPMorgan_Chase. (accessed: 7.04.2020).

[61] Wikipedia. *Jupyter Notebook - Project Jupyter*. URL: https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_Notebook. (accessed: 7.05.2020).

[62] Wikipedia. *Kaggle*. URL: https://en.wikipedia.org/wiki/Kaggle. (accessed: 23.03.2020).

[63] Wikipedia. *Machine Learning*. URL: https://en.wikipedia.org/wiki/Machine_learning. (accessed: 7.05.2020).

[64] Wikipedia. *Markov decision process*. URL: https://en.wikipedia.org/wiki/Markov_decision_process. (accessed: 13.05.2020).

[65] Wikipedia. *Microsoft*. URL: https://en.wikipedia.org/wiki/Microsoft. (accessed: 7.04.2020).

[66] Wikipedia. *Open-high-close-low chart*. URL: https://en.wikipedia.org/wiki/Open-high-low-close_chart. (accessed: 7.04.2020).

[67] Wikipedia. *Pandas (software)*. URL: https://en.wikipedia.org/wiki/Pandas_(software). (accessed: 23.03.2020).

[68] Wikipedia. *Pearson Correlation Coefficient*. URL: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient. (accessed: 10.04.2020).

[69] Wikipedia. *Procter & Gamble*. URL: https://en.wikipedia.org/wiki/Procter_%26_Gamble. (accessed: 20.04.2020).

[70] Wikipedia. *Project Jupyter*. URL: https://en.wikipedia.org/wiki/Project_Jupyter. (accessed: 24.03.2020).

[71] Wikipedia. *Python (programming language)*. URL: https://en.wikipedia.org/wiki/Python_(programming_language). (accessed: 23.03.2020).

[72] Wikipedia. *Reinforcement Learning*. URL: https://en.wikipedia.org/wiki/Reinforcement_learning. (accessed: 10.05.2020).

[73] Wikipedia. *Restricted Boltzmann machine*. URL: https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine. (accessed: 13.05.2020).

[74] Wikipedia. *Semi-supervised learning*. URL: https://en.wikipedia.org/wiki/Semi-supervised_learning. (accessed: 8.05.2020).

[75] Wikipedia. *Sigmoid function*. URL: https://en.wikipedia.org/wiki/Sigmoid_function. (accessed: 10.04.2020).

[76] Wikipedia. *Softmax Function*. URL: https://en.wikipedia.org/wiki/Softmax_function. (accessed: 31.05.2020).

[77] Wikipedia. *The Coca-Cola Company*. URL: https://en.wikipedia.org/wiki/The_Coca-Cola_Company. (accessed: 7.04.2020).

[78] Wikipedia. *Walmart*. URL: https://en.wikipedia.org/wiki/Walmart. (accessed: 7.04.2020).

[79] Yahoo. *Yahoo Finanzas*. URL: https://es.finance.yahoo.com. (accessed: 17.05.2020).