République Tunisienne
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université de Sousse

Institut Supérieur des Sciences
Appliquées et de Technologie
de Sousse

DÉPARTEMENT INFORMATIQUE

MÉMOIRE DE MASTER

En vue de l'obtention du diplôme de Master de Recherche en :

Informatique

Option : Systèmes Pervasifs Intelligents

Deep Learning Stock Market Simulator To Study The Behaviour Of traders In A Crisis Situation

Elaboré par :

Mtir Achref

Soutenu le 08/04/2021 devant le jury :

Président :	**Mr Ben Salah Chokri**	**ISSAT de Sousse**
Examinateur :	**Mr Bouden Mondher**	**ISSAT de Sousse**
Encadrant :	**Mme Kanzari Dalel**	**ISSAT de Sousse**

Année Universitaire : 2020/2021

Dedications

I dedicate the fruit of my work to my lovely parents Habiba and Kmaies, who supported me and appreciated my hard work and dedication to my dear brothers and sisters who helped me in my journey with every bit of love and support and pushed me to become a better version of myself.
to all my friends, to all my loved ones and to those I love, I offer you this modest work.

Achref

Acknowledgments

First of all, I would like to thank Dr. Kanzari Dalel who supervised me throughout my dissertation, for their judicious advice, for providing me with a serene working environment, encouraged throughout this dissertation. I would like to address my thanks to the people who have helped me and who have thus contributed to the development of this research. My thanks also go to the president and members of the jury who agreed to Evaluate my work. Finally, I thank all those who participated directly or indirectly in the realization of this work.

Abstract

The stock market price variation is a complex subject due to the numerous variables included like news, social media and pandemics.

In our study we focus on the behaviour of traders in the stock market during the COVID-19 pandemic to predict future prices and explain the stock crash causes.

A stock crash is where the prices of an asset drops quickly and suddenly.

We attacked this problem by developing a deep learning multiagent system called “Stock Market Simulator”.

We took the Bitcoin market as a subject for our study, we collected historical close prices from YahooFinance, then we built our simulator with three different behaviours in mind.

We compared the output of simulations with the reality of the stock market and we were successful to meet the reality in a lot of points.

Our hypothesis was right as we found out that the market is composed of three main behaviours (rational, emotional, memetic).

The behaviour responsible for a stock crash is the memetic because it can magnify the price quick change.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

GENERAL INTRODUCTION

Stock markets are delicate and change quickly due to a lot of parameters.
Traders are always trying to prevent falling into a stock crash, to do so researches have done numerous researches in the IT field and data analyzing, most of these researches are traditional finance based.

This field can be seen as purely financial (through the study of the economic issues they represent), mathematical (through the study of the properties of price series) or human studies (through the study of the psychology of the economic actors involved in it).

Yet none of these disciplines can yet offer a reliable approach that would overcome the complexity of markets and fully understand them.

The study of crisis prevention in the finance field is an active area of Multi-Agent Systems (MAS) research.In fact , MAS models allow for cost-effective simulations of changing organizational structures, enabling analysis of the implications for enactment during crisis escalation with respect to roles and communication structures.

Pandemics affect the economy and lead to financial crisis, therefore it is an interesting topic to study the situation, in similar cases, decision-making is the most important act an individual may take.

We model a decision-support system based on a deep learning approach and then we involve it in the multi-agent systems paradigm to simulate the human behaviour.

A decision support system (DSS) is a computerized program used to support determinations, judgments, and courses of action in an organization or a business. A DSS sifts through and analyzes massive amounts of data, compiling comprehensive information that can be used to solve problems and in decision-making.

Our aim is to analyse the behaviours of deep-learning based agents, that represents traders in the stock market, and to examine the impact on the market stability.

CHAPTER 1 GENERAL PROJECT CONTEXT

Introduction

In this chapter we provide a general overview of the project context. First, we start by defining the problem statement and the project goal. Additionally, we present the report structure.

1.1 Problem Discussion

Traders all around the world are trying to maximize gains and minimize their losses.
They tend to choose the best action to take regarding the stock market whether it's holding, buying or selling, so for this action to be the best, market traders act in a certain way depending on the nature of their reasoning.

Traders in stock markets as agents, take actions (buy, hold, sell) and behave in a way that influences the stability of a stock market and it can lead to a financial crisis with other causes like wars, diseases ...

The 2020 stock market crash, also referred to as the Coronavirus Crash, was a major and sudden global stock market crash that began on 20 February 2020, and ended on 7 April.[w1]

The crash was the fastest fall in global stock markets in financial history, and the most devastating crash since the Wall street crash of 1929. The crash however would only cause a short-lived bear market, and in April global stock markets re-entered into a bull market.[22]

Future prices of an asset are unknown so we need to process huge historical prices datasets to get precise predictions.

There are some difficulties when trying to represent human behaviour by an IT system so we hope that we manage to get low errors when benchmarking our multiagent system.

1.2 Hypothesis

There is a direct correlation between stock market crashes and the presence of certain behaviours.

The stock price changes to represent the balance between the selling and the buying actions, when the selling is dominating, the price will be dropping and this can lead to a crash.

The memetic behaviour, where a trader follows what the majority of other traders have taken as actions, is the main behaviour responsible for the stock market crash and the fast shifting of prices.

The marketplace is a mixture of different behaviours with a slight dominating side.

1.3 Project goals

The main questions of this project are as follows:

“How can machine learning methods be used to simulate investors' reasoning?”

“How can investors' behaviors impact the price variation ?”

“What is the most common trader’s behaviour ?”

“What is the behaviour that is responsible for the stock market crash ?”

These broad goals lead to more specific questions that assist with the evaluation of the results with regard to the already described overarching goal.

The research questions enquired at the beginning of this thesis are the following:

• Is a Deep Learning approach able to reflect the effective reasoning of agents in a dynamic environment?

• What are the main factors that form the stock trading environment of Deep Learning?

• How can we model the stock market?

1.4 Outline

Following this introductory chapter , the remainder of the thesis is structured as follows:

- Chapter 2 provides the theoretical background about Deep learning. Also more aspects of Deep Learning algorithms are discussed. The chapter also discusses multi-agent systems. The purpose of this chapter is to clarify all these concepts and build all the knowledge before describing the proposed system.

- In chapter 3 we investigate the state of art in the field of our research to get a detailed description of what researchers have done to resolve the problem of predicting stocks price and prevent crisis formation.

- In Chapter 4, we introduce the proposed approach developed in this work. Firstly, a description of the designed environment is presented. Then, the multi-agent system , its structure and all its specifications are explained and discussed.

- In Chapter 5, all experiments are shown and described. This chapter refers to all parameters of the environment, agents and experiments.

Summary

In this chapter we presented an overall look on the project context, main problem and objectives.The chapter that follows focuses on the theoretical concepts pertaining to this project which will eventually justify the different theoretical choices made.

CHAPTER 2 THEORETICAL BACKGROUND

Introduction

In this chapter we are going to define the basic fundamentals of stock markets, multiagent systems and other aspects so we can understand the work environment of this research.

2.1 Stock market fundamentals

Here we will try to explain some of the basics of the stock market :

Ownership:

The most basic concept of the stock market is the idea that each share of stock represents a small portion of ownership of a corporation. While most businesses are founded by small groups of people, when a company "goes public" its owners decide to sell shares of stock and, in turn, receive cash from buyers. A company may have thousands of investors, but each one has the right to profit from the company's success and each runs the risk of losing money if the company performs poorly. Stockholders receive updates from the company and can vote for board members to influence the business's activities.[1]

Stock Trading:

Trading is another key concept behind the stock market. Despite the name, trading refers to buying and selling shares of stock for cash, not actually trading them for other stocks. Stock trading takes place on open markets, in which anyone can participate. Most stock markets only allow brokers to place buy and sale orders, but anyone with access to a broker, including automated electronic brokers that operate online, can trade on the market. Since anyone can participate in stock trading, buyers and sellers are free to make transactions for any price they agree to.[1]

Supply and Demand:

A stock's price depends on many factors, the most basic of which is supply and demand. When a company’s stock is in high demand, prices for its stock will rise. When more people want to sell shares than there are buyers for those shares, prices for those shares will fall. Demand is dependent upon how likely other investors think it is for a company's stock to rise in value. In a typical transaction,

the seller thinks the stock is at its peak price, while the buyer expects it to rise in value at some point in the future.[1]

Investing:

For stock market investors, the basic concept of how the market operates has special significance. Since stocks are tied to individual companies, they are far more likely to change in value than other investments, such as currency, commodities and mutual funds. This makes the stock market a highly volatile, but potentially profitable, place to invest. Putting some money into the stock market is a way to diversify one's investments and take advantage of the chance for profits while keeping risk at a reasonable level by investing elsewhere at the same time.[2]

Index:

Put simply, an index is a list of stocks meant to capture a common theme. For example, the S&P 500 (one of the most well-known indexes) is made up of the 500 largest publicly-traded companies in the U.S., such as 3M (MMM), Amgen (AMGN), Kohl's (KSS), Lowe's (LOW), PayPal (PYPL), Pfizer (PFE), and many more. The Dow Jones Industrial Average (DJIA) is composed of 30 large U.S. stocks, such as McDonald's (MCD), Cisco Systems (CSCO), Coca-Cola (KO), Caterpillar (CAT), and Exxon-Mobil (XOM), and others. The Russell 2000 index comprises 2000 small stocks. The prices of all the underlying stocks in an index are used to create the overall price of the index. Investors use indexes to measure a certain section of the stock market. For example, want to know how small cap stocks are doing? Look at the Russell 2000. Or, to get a complete view of the American stock market, you can look at the Wilshire 5000 Index which represents all publicly-traded companies in the U.S.[w2]

Market cap:

When investors talk about "market caps," they're simply referring to how big a company is. The full phrase is "market capitalization," which is the total value of the company. You arrive at the value by multiplying the number of shares it has on the stock market by the price per share. Since share prices fluctuate each day, the market cap of the underlying company also fluctuates. For example, a big company like Apple (AAPL) is considered a large-cap stock. As of today, Apple has 4,519,180,000 shares outstanding on the public markets. Their stock price is at $209.12. If you multiply those two numbers together, you arrive at a market capitalization of $945 billion. Wow. A small cap stock is a small company, such as GoPro (GPRO). As of today, their market cap is only $553 million. Here

are the official categorizations for companies based on their size: The nano-cap stock definition is companies with a market cap between $0M — $50M.

The micro-cap stock definition is companies with a market cap between $50M — $300M.

The small-cap stock definition is companies with a market cap between $300M — $2B

The mid-cap stock definition is companies with a market cap between $2B — $10B.

The large-cap stock definition is companies with a market cap between $10B — $300B.

The mega-cap stock definition is companies with a market cap of $300B and above.

Most people simply call big companies “large-cap stocks,” mid-size companies “mid-cap stocks”, and small companies “small-cap stocks.” Investors typically don’t get as specific as “nano-cap” or “mega-cap.”[w2]

Stocks:

Shares of ownership in a publicly-traded company. Usually perform well over long periods of time, but investors risk making mistakes when buying and selling stocks without a strategy.[3]

Market crash:

A “stock market crash” is a broad term that basically means a violent selloff and a significant decline in stock market prices over a short period of time. Crashes tend to be driven by panic and often draw the attention (and participation) of common investors. Crashes often happen at the end of a long bull market when investors have gotten greedy and careless after years of upward stock prices. They can also be driven by underlying weakness in the U.S. economy, such as the subprime mortgage crisis of 2008–2009. A stock market crash is different from a market “correction,” which is defined as a 10%+ decline in market prices from a recent high. While a correction may be unpleasant, it’s a normal and healthy part of the stock market. Market corrections typically don’t happen in a single day but over a period of days, weeks, or months. Another common term is a “bear market,” which is defined as a 20%+ decline in market prices over a two-month period. When investors discuss a correction or a bear market, they’re often talking about the stock market overall. They’ll say, “The S&P 500 is now in correction territory.” This means the S&P 500 has declined by more than 10%.

However, these terms can also be applied to individual stocks. So, if you hear an analyst say, “Walmart (WMT) is currently in a bear market.” She’s saying that Walmart stock has declined more than 20% and stayed there for at least two months.[3]

2.2 Multiagent systems and Machine Learning Algorithms

2.2.1 Multiagent systems

Mas

A multi-agent system is a computerized system composed of multiple interacting intelligent agents. Multi-agent systems can solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. Intelligence may include methodic, functional, procedural approaches, algorithmic search or reinforced learning.[4]

Intelligent agent

In artificial intelligence, an intelligent agent (IA) refers to an autonomous entity which acts, directing its activity towards achieving goals, upon an environment using observation through sensors and consequent actuators, Intelligent agents may also learn or use knowledge to achieve their goals.

They may be very simple or very complex. A reflex machine, such as a thermostat, is considered an example of an intelligent agent.

Intelligent agents are often described schematically as an abstract functional system similar to a computer program.

Researchers such as Russell & Norvig in “Artificial Intelligence: A Modern Approach” consider goal-directed behavior to be the essence of intelligence, a normative agent can be labeled with a term borrowed from economics, "rational agent". In this rational-action paradigm, an AI possesses an internal "model" of its environment.

This model encapsulates all the agent's beliefs about the world. The agent also has an "objective function" that encapsulates all the AI's goals. Such an agent is designed to create and execute whatever plan will, upon completion, maximize the expected value of the objective function.[21]

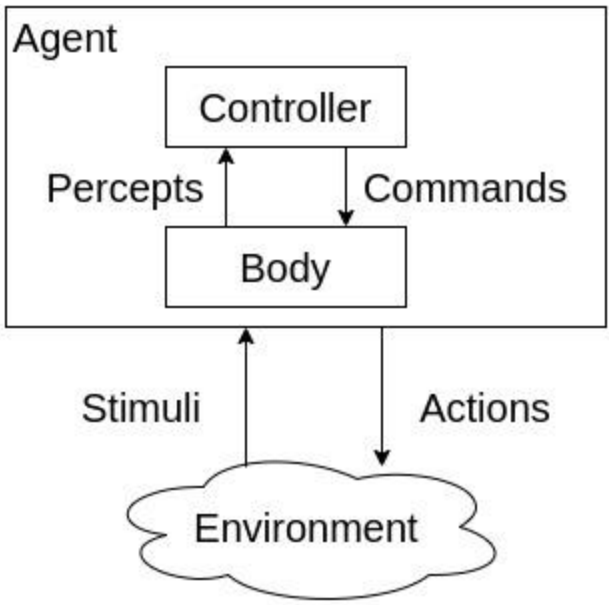

Figure 1: Agent architecture[5]

In our study we will be using intelligent agents to simulate traders in the stock market, the AI agent will be representing a human being trader that interacts with the stock environment.

Each agent has a controller which is the module responsible for taking actions (buy, hold, sell), as seen in figure 1.

The body module is the core of the agent which is the neural network in our case.

The percepts are the output of the neural network so in our case it is the close price prediction of the bitcoin.

The commands are the decision taken by the controller.

The stimuli is the data viewed by agents which is the bitcoin historical close price dataset coming from the environment and the actions are the output of the agents regarding the final decision.

2.2.2 Machine learning (ML)

Machine Learning is an artificial intelligence technology that enables computers to learn without having been explicitly programmed to do so. To learn and grow, however, computers need data to analyze and train on. Indeed, Big Data is the essence of Machine Learning, and it is the technology that unlocks the full potential of Big Data. Find out why this technique and Big Data are interdependent.[6]

Supervised learning

Although the two types of learning come under artificial intelligence, in the first case a researcher is there to “guide” the algorithm on the path of learning by providing it with examples that he considers convincing after having them. previously labeled with expected results. Artificial intelligence then learns from each example by adjusting its parameters (the weights of the neurons) so as to reduce the difference between the results obtained and the expected results. The margin of error is thus reduced over the training sessions, with the aim of being able to generalize your learning to new cases.[6]

Unsupervised learning

In the case of unsupervised learning, machine learning takes place completely autonomously. Data is then communicated to the machine without providing it with the examples of expected output results. While this solution seems ideal on paper because it does not require large labeled datasets (whose expected results are known and communicated to the algorithm), it is important to understand that these two types of learning are not inherently not suitable for the same types of situations.[6]

2.3 Deep Learning

2.3.1 Core concept

Deep learning is a branch of machine learning, which is based on a group of algorithms that seek to shape high-level abstractions of data using a deep graph with multiple layers of processing, consisting of several linear and non-linear alterations.

Deep Learning works on the computer system to perform tasks such as speech recognition, Predictions, image identification, and projection. Rather than organizing information to act by predetermined equations, this learning determines the basic patterns of that information and teaches computers to develop by identifying patterns in processing layers.

This type of learning is a complete branch of machine learning methods based on learning representations of information.

In this sense, Deep Learning is a set of machine learning algorithms that attempt to integrate multiple levels, which are recognized statistical models that correspond to different levels of definitions. Lower levels help define many notions of higher levels.[7]

2.3.2 Mathematical and Algorithmic Understanding of Deep Learning

A neuron

An artificial neuron, figure 2, is a mathematical function conceived as a model of biological neurons.

It is a bloc of mathematical operations linking entities.

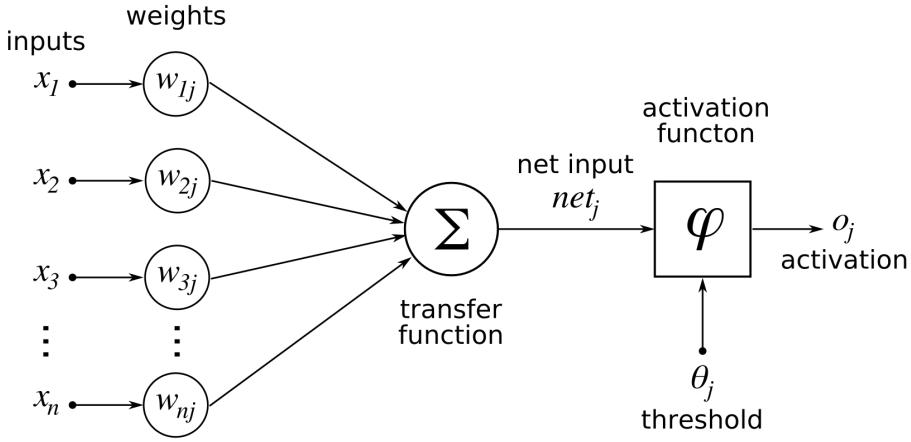

Figure 2: Artificial Neuron architecture[7]

inputs: data to be processed and learned on.

weights: random values given to each segment of data to decide its importance.

transfer function: translates the input signals to output signals.

activation function: outputs a small value for small inputs, and a larger value if its inputs exceed a threshold.

MLP (multilayer perceptron)

In general, neural networks better known as MLP, for 'Multilayer Perceptron', is a type of direct formal neural network organized into several layers in which information flows from the input layer to the output layer only.

Each layer consists of a defined number of neurons, we distinguish:

- the input layer
- the hidden layers
- the output layer

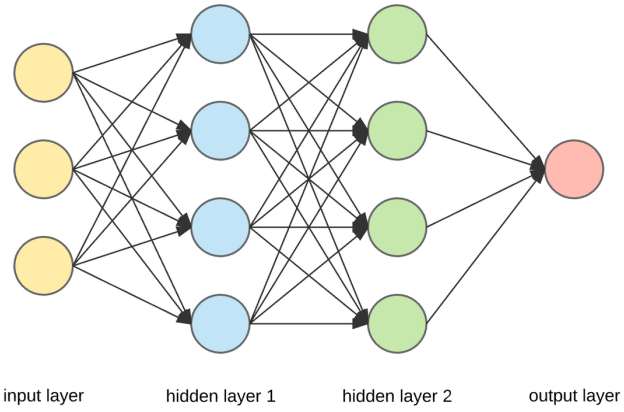

Figure 3: Deep learning architecture[7]

Data Preprocessing

In any machine learning project in general, we divide our data into 3 sets:

- Train set: used to train the algorithm and construct batches
- Dev set: used to finetune the algorithm and evaluate bias and variance
- Test set: used to generalize the error/precision of the final algorithm

we split the data according to its size:

- If the size of our dataset is between 100 to 10,00,000, then we split it in the ratio 60:20:20. That is 60% data will go to the Training Set, 20% to the Dev Set and remaining to the Test Set.

- If the size of the data set is greater than 1 million then we can split it in something like this 98:1:1 or 99:0.5:0.5

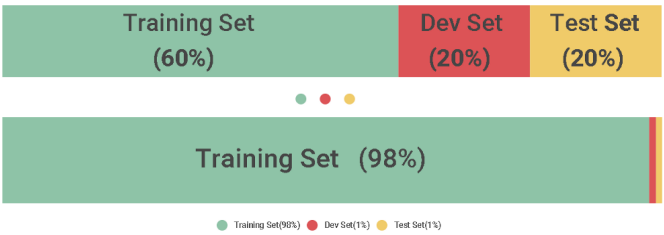

Figure 4: Data split[7]

Learning Algorithm

Learning in neural networks is the step of calculating the weights of the parameters associated with the various regressions throughout the network, see figure 3.

In other words, we aim to find the best parameters that give the best prediction/approximation, starting from the input, of the real value.

For this, we define an objective function called the loss function and denote 'J' which quantifies the distance between the real and the predicted values on the overall training set.[w4]

We minimize 'J' following two major steps:

- Forward Propagation: we propagate the data through the network either in entirely or in batches, and we calculate the loss function on this batch which is nothing but the sum of the errors committed at the predicted output for the different rows.

- Backpropagation: consists of calculating the gradients of the cost function with respect to the different parameters, then applying a descent algorithm to update them. We repeat the same process a number of times called epoch number. After defining the architecture, the learning algorithm is written as follows:

- Initialization of the model parameters, a step equivalent to injecting noise into the model.
- `For` i=1,2...N: (N is the number of epochs)
  - Perform `forward propagation` :
    - $\forall i$, Compute the predicted value of x_i through the neural network: $\hat{y}_i^\theta$
    - Evaluate the function : $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}_i^\theta, y_i)$
      where m is the size of the training set, θ the model parameters and $\mathcal{L}$ the cost$^{(*)}$ function
  - Perform `backpropagation` :
    - Apply a descent method to update the parameters :

$$\theta =: G(\theta)$$

Figure 5: Learning Algorithm[7]

Activation function

Activation functions are a kind of transfer functions that select the data propagated in the neural network. The underlying interpretation is to allow a neuron in the network to propagate learning data (if it is in a learning phase) only if it is sufficiently excited.

Here is a list of the most common functions:

- **ReLU**:

$$\psi(x) = x\mathbf{1}_{x \geq 0}$$

- **Sigmoid**:

$$\psi(x) = \frac{1}{1+e^{-x}}$$

- **Tanh**:

$$\psi(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$$

- **LeakyReLU**:

$$\psi(x) = x\mathbf{1}_{x \geq 0} + \alpha x\mathbf{1}_{x \leq 0}$$

Figure 6: Activation Functions[7]

Optimization algorithm

It is a procedure which is executed iteratively by comparing various solutions till an optimum or a satisfactory solution is found.

- **Risk**

Let us consider a neural network denoted by *f*. The real objective to optimize is defined as the expected loss over all the corpora:

eq1:

$$R(f) = \int p(X,Y)\mathcal{L}(f(X),Y)dXdY$$

Where X is an element from a continuous space of observables to which correspond to a target Y and $p(X,Y)$ being the marginal probability of observing the couple (X,Y).

- **Empirical risk**

Since we can not have all the corpora and hence we ignore the distribution, we restrict the estimation of the risk on a certain dataset well representative of the overall corpora and consider all the cases equiprobable.

In this case: we set m to be the size of the representative corpora, we get :∫=∑ and $p(X,Y)$=1/m. Hence, we iteratively optimize the loss function defined as (eq.2.):

eq2:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}_i^{\theta}, y_i)$$

Plus we can assert that:

eq3:

$$min_f R(f) \approx min_\theta J(\theta)$$

There exist many techniques and algorithms, mainly based on gradient descent, which carries out the optimization. In the sections below, we will go through the most famous ones. It is important to note that these algorithms might get stuck in local minima and nothing assures reaching the global one.[w7]

- **Gradient descent with momentum**

A variant of gradient descent which includes the notion of momentum, the algorithm is as follows:

- Initialize $V_{dW} = 0_{dW}, V_{db} = 0_{db}$
- On iteration k:
  - Compute dW and db on the current mini-batch
  - $V_{dW} = \beta V_{dW} + (1 - \beta)dW; V_{db} = \beta V_{db} + (1 - \beta)db$
  - Update the parameters:
    - $W := W - \alpha dW$
    - $b := b - \alpha db$

Figure 7: Gradient descent[7]

(α,β) are hyperparameters.

Since dθ is calculated on a mini-batch, the resulting gradient ∇J is very noisy, this exponentially weighted averages included by the momentum give a better estimation of derivatives.

- **Adam**

Adam is an adaptive learning rate optimization algorithm designed specifically for training deep neural networks. Adam can be seen as a combination of RMSprop and gradient descent with momentum.

It uses square gradients to set the learning rate at scale as RMSprop and takes advantage of momentum by using the moving average of the gradient instead of the gradient itself as the gradient descends with momentum.

The main idea is to avoid oscillations during optimization by accelerating the descent in the right direction.

The algorithm of Adam optimizer is the following:

- Initialize: $V_{dW} = 0$, $S_{dW} = 0$, $V_{db} = 0$, $S_{db} = 0$;
- On iteration k:
  - Computation of dW and db through backpropagation
  - Momentum:
    - $V_{dW} = \beta_1 V_{dW} + (1 - \beta_1) dW$
    - $V_{db} = \beta_1 V_{db} + (1 - \beta_1) db$
  - RMSprop:
    - $S_{dW} = \beta_2 S_{dW} + (1 - \beta_2) dW^2$
    - $S_{db} = \beta_2 S_{db} + (1 - \beta_2) db^2$
  - Correction:
    - $V_{dW} = \frac{V_{dW}}{1 - \beta_1^k}$
    - $S_{dW} = \frac{S_{dW}}{1 - \beta_2^k}$
    - $V_{db} = \frac{V_{db}}{1 - \beta_1^k}$
    - $S_{db} = \frac{S_{db}}{1 - \beta_2^k}$
  - Parameters' update:
    - $W = W - \alpha \frac{V_{dw}}{\sqrt{S_{dW}} + \epsilon}$;
    - $b = b - \alpha \frac{V_{db}}{\sqrt{S_{db}} + \epsilon}$

Figure 8: Adam optimizer algorithm[7]

2.3.3 Long-term short-term memory LSTM

Long-term short-term memory networks (LSTMs) are an extension for recurrent neural networks, which expands their memory. Therefore, it is well suited for learning from important experiences which have very long delays in between.

The units of an LSTM are used as building units for the layers of an RNN, which is then often referred to as an LSTM network.

LSTMs allow RNNs to remember their inputs over a long period of time. This is because LSTMs contain their information in memory, which is very similar to computer memory because LSTM can read, write, and delete information from its memory.

This memory can be seen as a gated cell, where gated means that the cell decides to store or delete information (for example whether it opens the doors or not), depending on the importance it assigns to the information. . The assignment of importance is done through weights, which are also learned by the algorithm. It just means that it learns over time which information is important and which is not.

In an LSTM you have three doors: entry, forget and exit door. These doors determine whether or not to let a new entry enter (front door), delete the information because it is not important (forget the door) or let it influence the exit at the current time step (door Release).[8]

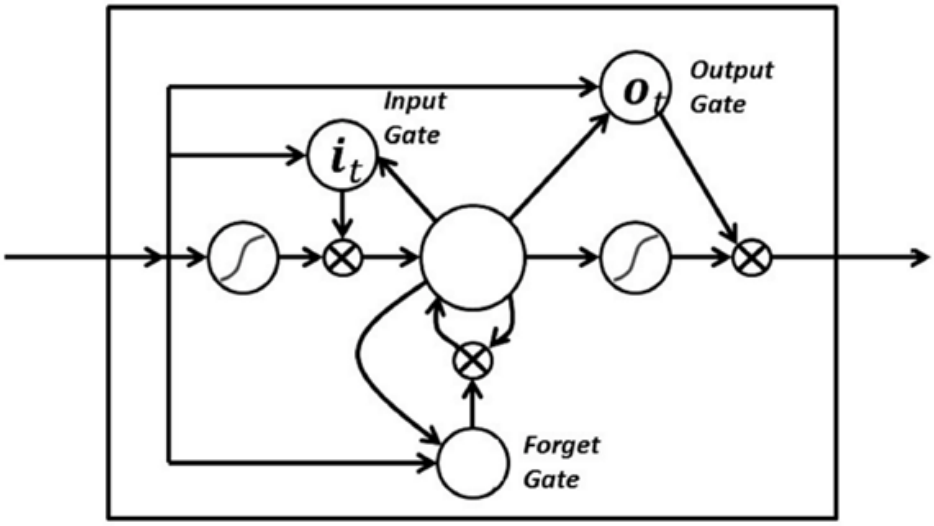

Figure 9: An illustration of an RNN [8]

Summary

In this chapter we gave a general theoretical background for our project. We discussed some mathematical background of deep learning, multi-agent systems and LSTM. In the next chapter we will discuss the state of art and the methodology used.

CHAPTER 3 STATE OF THE ART

Introducion

In this chapter we will be talking about the work that has been done in the past regarding the field of stock market price prediction and crisis prevention.

We will cite and detail the different existing approaches and techniques.

3.1 A Bat-Neural Network Multi-Agent System For Stock Price Prediction

The aim of this work is to Create a software that can predict stock prices accurately in a robust way is the interest of a lot of researchers and investors, but this problem is a complex subject because of its multiple parameters and variables which can be social, political, economical …

So this work is aiming to create a complex system to predict stock market prices accurately, using a bat-neural network multi-agent system (BNNMAS).[9]

In an abstract way, this model consists of four layers.

- Gathering Data (statistics, news…)
- Data preprocessing
- Artificial intelligence framework creation
- validation testing and reporting

To better understand the approach we will briefly explain the input and output of every layer in the Figure below:

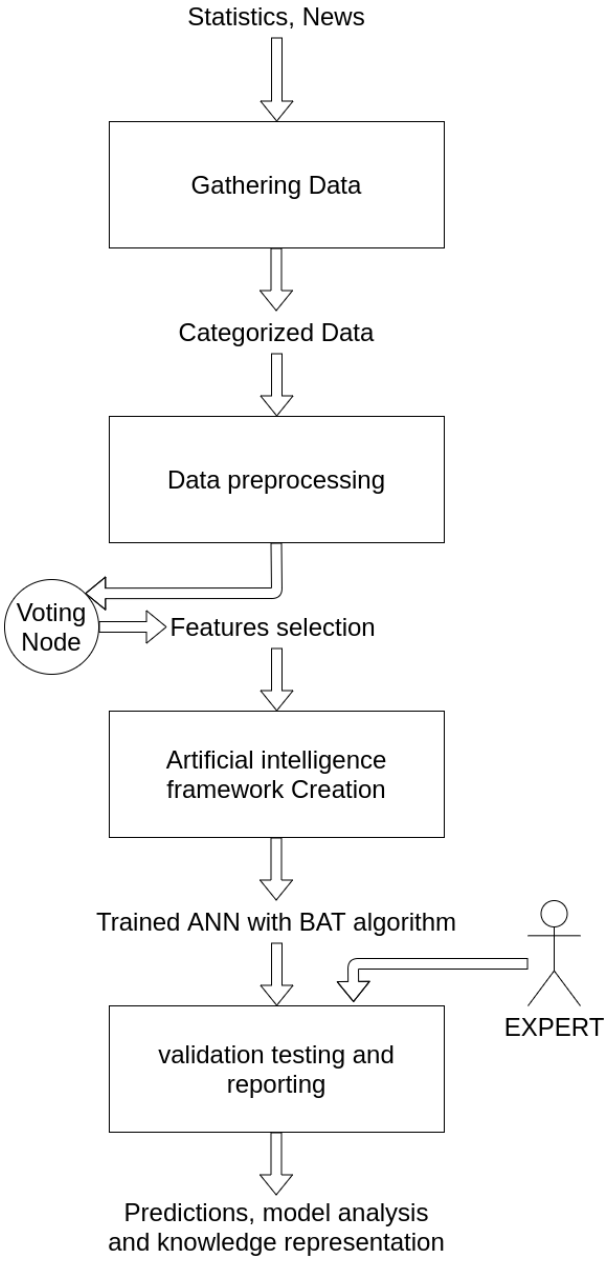

Figure 10: BNNMAS Architecture[9]

The model is tested using the DAX stock market, first data is gathered and it is the historical price from 1972 to 2012.

The DAX is a blue chip stock market index consisting of the 30 major German companies trading on the Frankfurt Stock Exchange.

The BNNMAS is then implemented to make predictions and the results are:

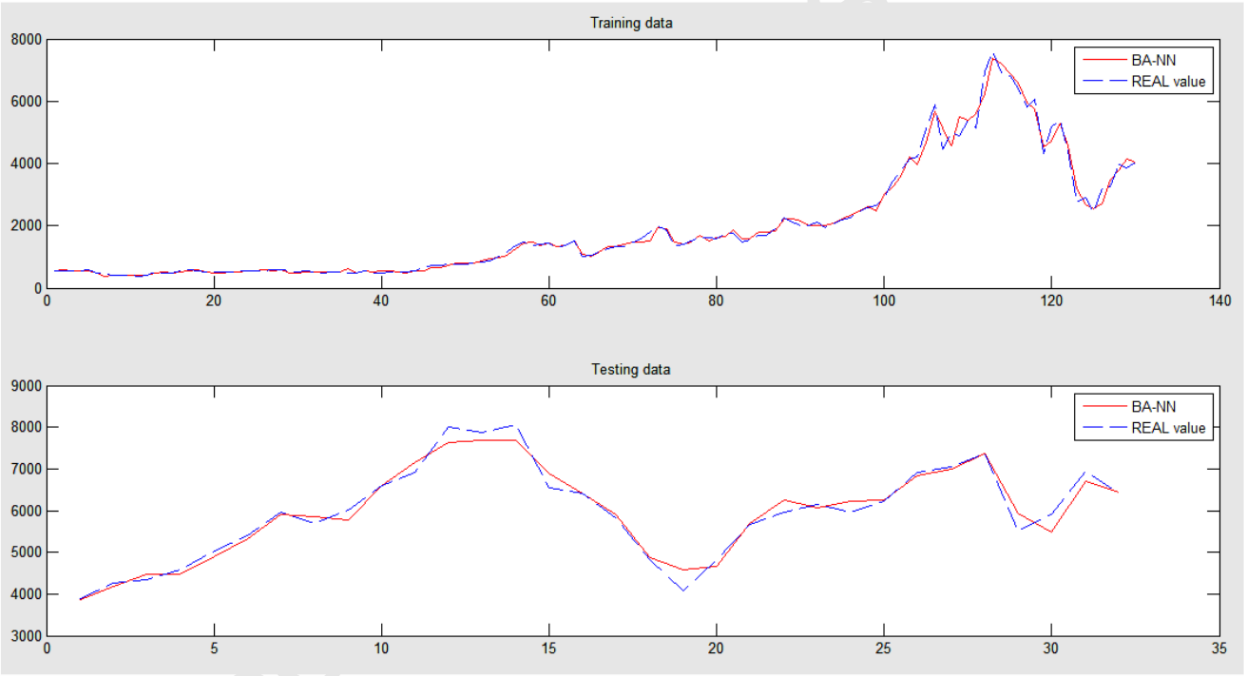

Figure 11: Experimental results of BNNMAS[9]

To calculate the precision of the model, the MAPE equation is used:

eq4:

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

M: mean absolute percentage error

n: number of times the summation iteration happens

At: actual value

Ft: forecast value

The results of MAPE for different models is shown in the graph below:

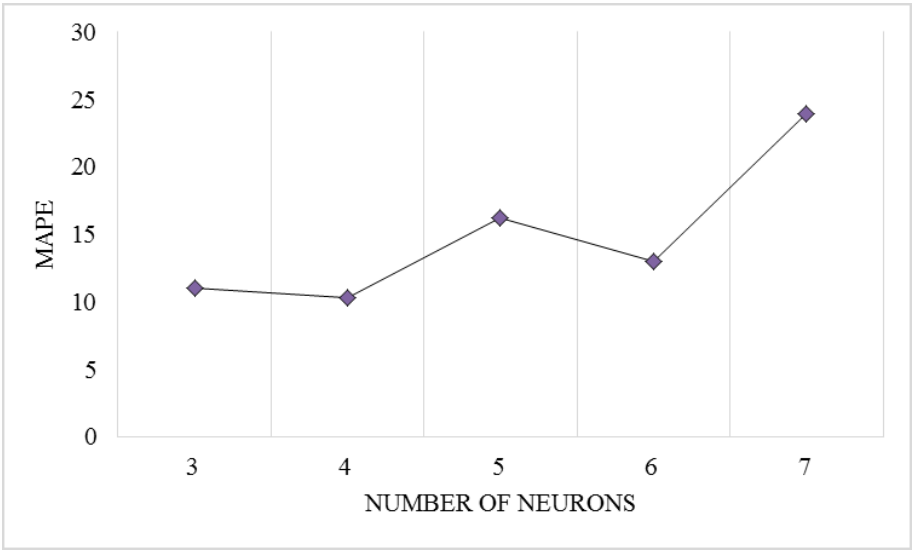

Figure 12: Performance of GA-NN for different number of neurons[9]

The BNNMAS has a MAPE value = 2.84 which is the lowest among all the models, so it is the most accurate especially when the input features are high.

3.2 An Intelligent Agent-Based Stock Prediction System

Financial prediction, such as stock forecasting, is always a popular topic for research studies and commercial applications. With the rapid growth of Internet technology in recent years, e-finance has become a vital application of e-commerce. However, in this sea of information made available through the Internet, an intelligent financial web-mining and stock prediction system can be a key to success.

In this research, a multi-agent-based stock advisory system is proposed, namely the iJADE Stock Advisor, a stock prediction system using a proposed hybrid radial basis function recurrent network (HRBFN) with the integration of the iJADE model.[10]

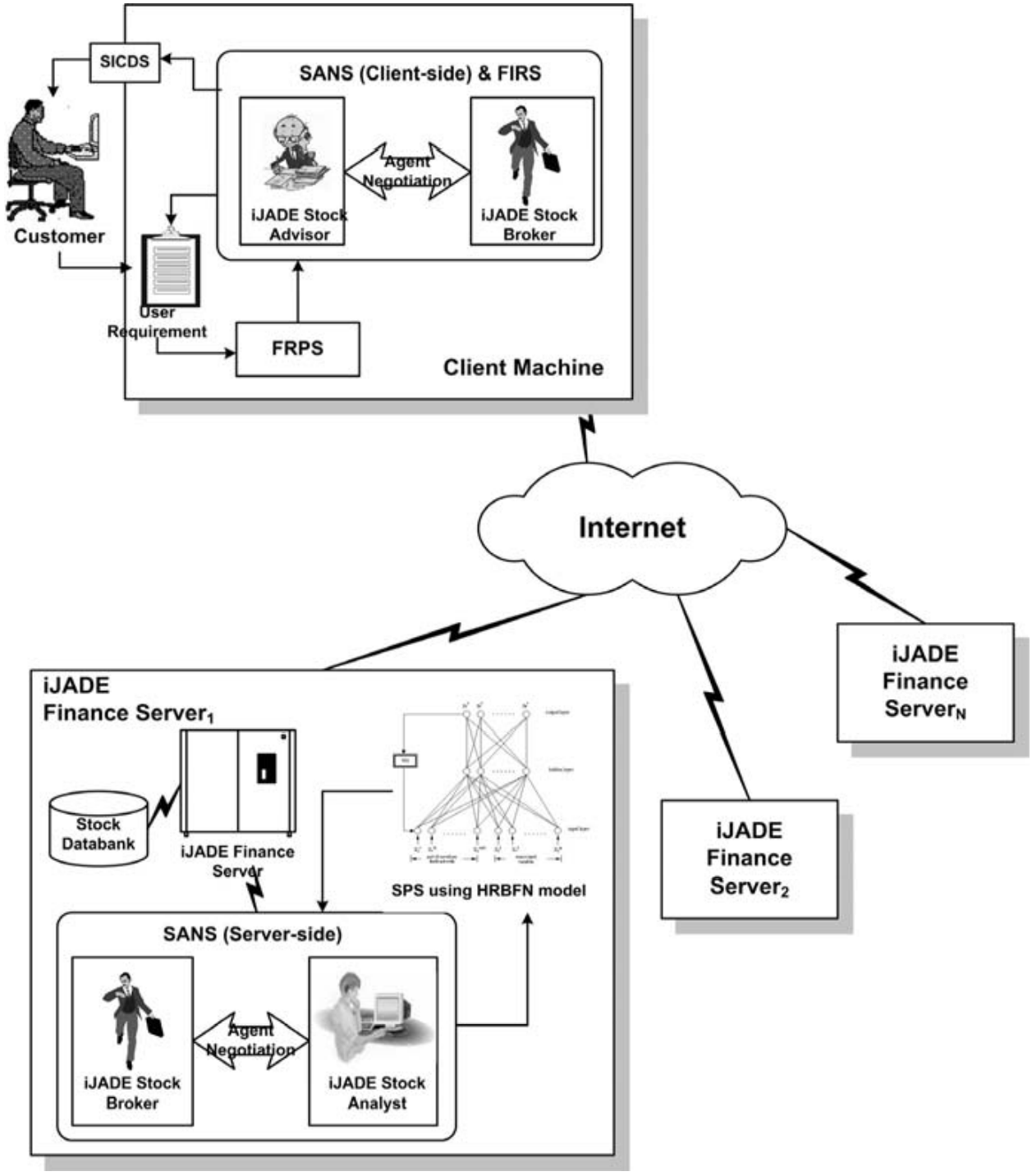

Figure 13: Schematic diagram of the iJADE Stock Advisor system[10]

The workflow of the iJADE Stock Advisor consists of the following five modules:

1. Forecast requirement processing scheme (FRPS) – done by the iJADE Stock Advisor.

2. Stock agents negotiation scheme (SANS) – between (1) the iJADE Stock Advisor and the iJADE Stock Brokers and (2) the iJADE Stock Brokers and the iJADE Stock Analysts.

3. Stock prediction scheme (SPS) using the HRBFN – performed by the iJADE Stock Analysts.

4. Forecast information reporting scheme (FIRS) – between the iJADE Stock Brokers and the iJADE Stock Advisor.

5. Stock information consolidation and display scheme (SICDS) – performed by the iJADE Stock Advisor.

Two cases were studied in the experiments, the long term trend prediction which focuses on the change pattern between a buy and a sell points, to predict where the stock market is headed.

the short term trend prediction which focuses on the daily changes of the stock prices (open,high,low,close).

The average percentage error is calculated by taking the average of the percentage errors of the prediction results for the four typical stocks, namely HSBC Holdings, Tian On China Investment, Hong Kong Telecom, and Cheung Kong Holdings Ltd. in the period 1990–1993.

RMSE = Root Mean Square Error.

The results are presented in the table below:

Table 1: iJADE Stock Advisor prediction results[10]

Long-Term Trend Prediction			Short-Term Price Prediction		
Window Size (days)	Av. %[1] Error (%)	RMSE[2]	Window Size (days)	Av. %[1] Error (%)	RMSE[2]
10	2.345	6.45	2	2.347	12.75
15	1.075	4.37	3	1.473	9.05
20	1.073	4.31	4	1.476	9.37
25	1.238	4.63	6	1.487	9.56
30	2.487	6.71	8	2.746	13.22
40	5.347	10.06	10	3.248	14.29

3.3 Short-term Stock Market Price Trend Prediction Using a Customized Deep Learning System

It is always a field of interest to researchers, investors and traders when talking about machine learning and stock markets. So predicting stock prices using machine learning technologies is always an interesting topic, in this research authors have proposed a fine-tuned stock market price trend prediction system with developing a web application as the use case.[11]

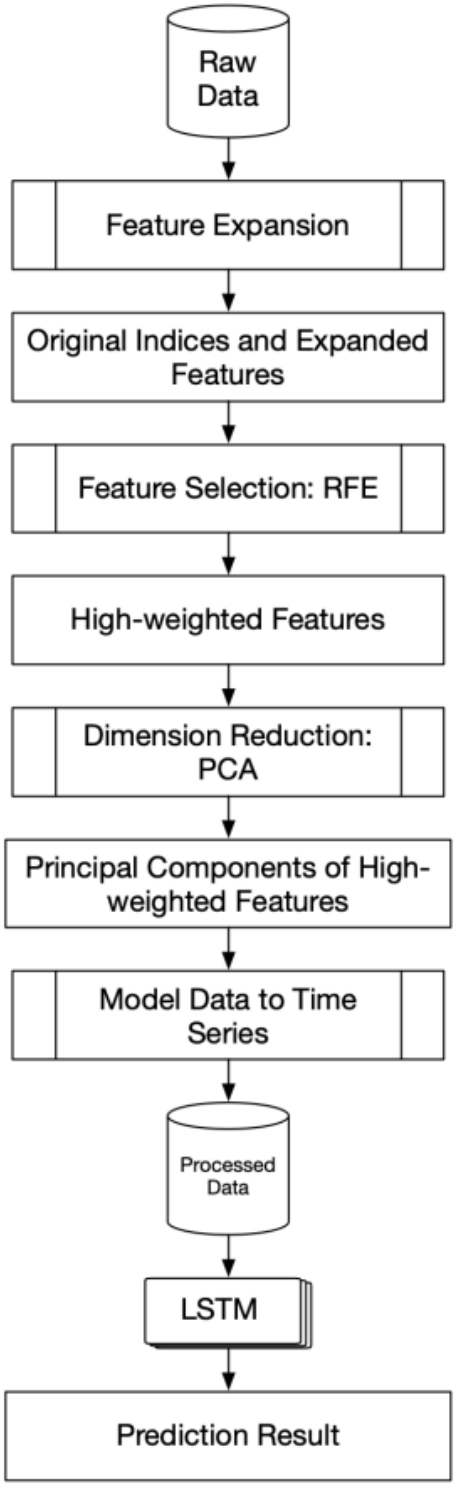

Figure 14: High-level Architecture of Proposed Solution[11]

The high-level architecture of their proposed solution could be separated into three parts.

First is the feature selection part, guaranteeing the selected features are all effective. Second,

They look into the data and perform the dimensionality reduction. And the last part which is

The main contribution of this work is to build a prediction model of target stocks.

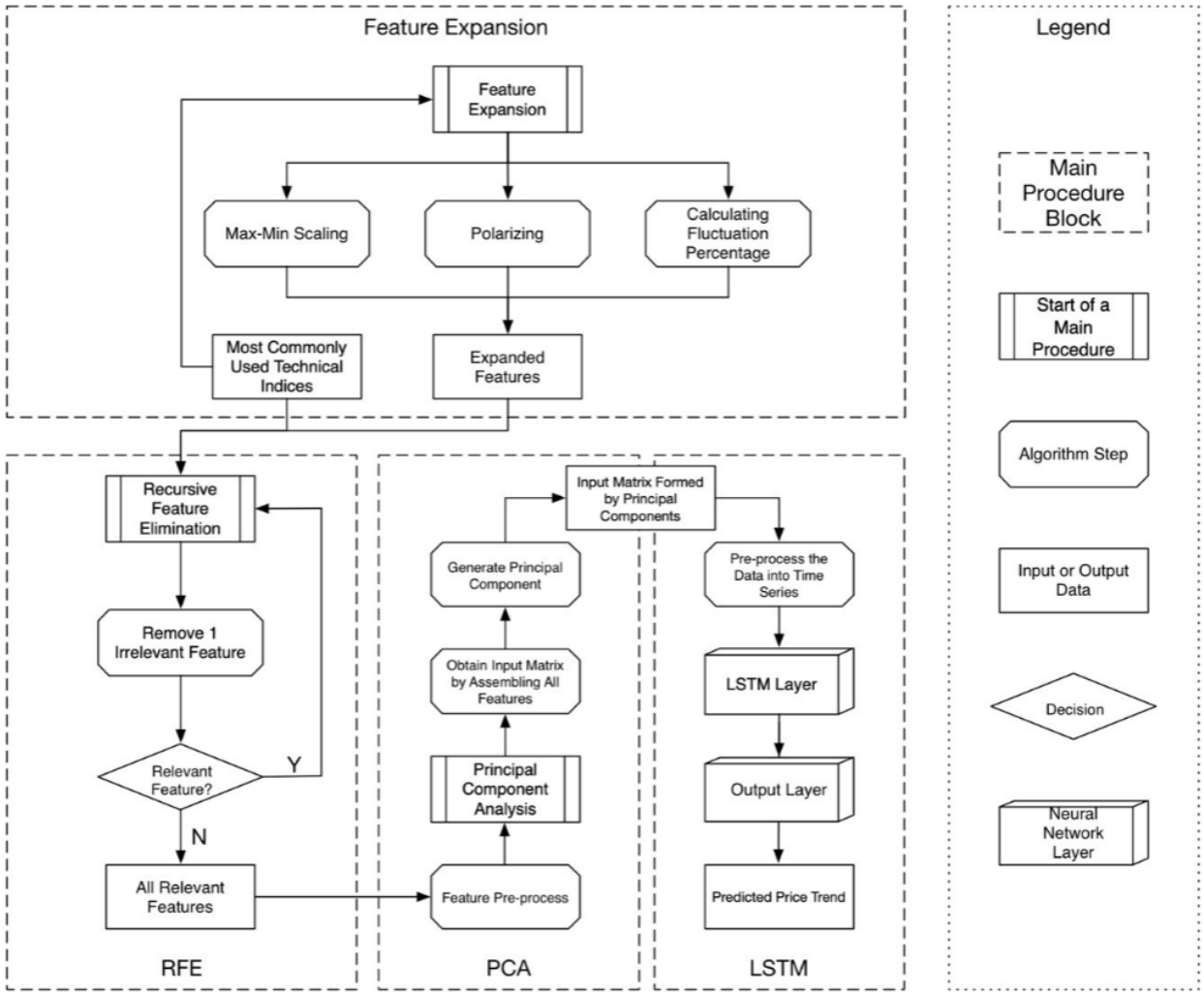

Figure 15: Detailed Technical Design[11]

For further explanation the main idea is to:

- now the most commonly used technical indices then feed them into feature extension procedure to get the expanded feature set.

- select the most effective i features from the expanded feature set.

- feed the data with selected features into the PCA algorithm to reduce the dimension into j features.

- get the best combination of i and j, process the data into a finalized feature set and feed them into the LSTM model to get the price trend prediction result.

Two separate algorithms were used in this work to reach the goals, an FE+RFE+PCA algorithm and an LSTM algorithm.

After gathering the results from the proposed model, it was compared with other models output to benchmark the performance of the proposed model.

Table 2: Model Performance Comparison – Metric Scores[11]

Model	F1 Score	Binary Accuracy	TPR (recall)	TNR (specificity)	FPR (fall-out)	FNR (miss rate)
LR	0.90	0.90	0.92	0.88	0.08	0.12
SVM	0.90	0.90	0.92	0.88	0.08	0.12
NB	0.89	0.89	0.90	0.87	0.10	0.13
MLP (Single hidden layer)	0.90	0.90	0.92	0.88	0.08	0.12
MLP (Three hidden layers)	0.90	0.90	0.92	0.87	0.08	0.13
MLP (Five hidden layers)	0.90	0.90	0.92	0.88	0.08	0.12
RAF	0.88	0.88	0.94	0.81	0.06	0.19
LSTM (Proposed Model)	0.93	0.93	0.96	0.90	0.04	0.10

3.4 Critics

All of the above studies did not take into consideration the nature of investors (their mindset) nor the financial crisis aspect for stock markets that seem to be appearing more often due to numerous factors such as political, diseases, social...

The models have not been tested on real markets, they do not present behaviors diversified by heterogeneous agents and their impact on market price variation.

A simple deep learning prediction isn’t sufficient to make a decision support approach.

Predicting the future will never be 100% accurate, so having a single process prediction isn’t the best choice.

All of the previous studies are one dimensional and don't meet the reality of stock markets precisely where this domain is multidimensional and should be treated by multi reasoning approaches.

So in our study we will be covering these points and study the behavior of investors during crisis period versus a stable period and conclude the best way to act in such situations to be in more of a safe place financially.

We will model three types of agents: rational, behavioral and mimetic to align with the reality of financial markets.

Summary

In this chapter we stated some of the studies that have been done in our field of research and the used technologies.

In the next chapter we will discuss the proposed model and a detailed description of the developed solution architecture.

CHAPTER 4 PROPOSED METHOD

Introduction

In this chapter we will discuss the proposed method and pair it with technical background and the developed model description, as we go further in this chapter we will be discussing the environment setup and the underlying simulation mechanisms.

4.1 Model description

In this study we will be collecting data from the yahoo finance database that contains stock prices of different assets like gold, petrol and companies like ford, apple, samsung and other cryptocurrencies like euthirum, bitcoin.

We will be using in our study the bitcoin historical close prices from 10 years ago and it will be stored in the historical asset prices.

After that we will be creating three different agent types with different core concept :

- Rational behaviour is what a well experienced trader will be acting like in the stock market, it goes only by statistics and true or false policy it will be giving close price prediction and taking actions in the market based on the last 60 days price change.
- Emotional behaviour is what an intermediate experienced trader is, it will be giving predictions and taking actions based on the last 40 days price change but with a variable confidence level “C” which will be simulating the emotions so it can go out of rational behaviour boundaries.
- Mimetic behaviour is what a newbie trader will be acting like in the stock market it will be taking actions and giving predictions based on the last 20 days of historical bitcoin price change, we will be simulating the mimetic behaviour by making this trader view other traders actions and follow the most frequent action taken in the trading phase.
-

Hypothesis :

the output of our “Stock Market Simulator” will be gathered and compared to real world scenarios to determine the resemblance to the reality and to explain the real stock market behaviour combination and what are the major causes that lead to a market crash.

This system will be also capable of being used by experts to help them in their decisions and to study the stock market price variation.

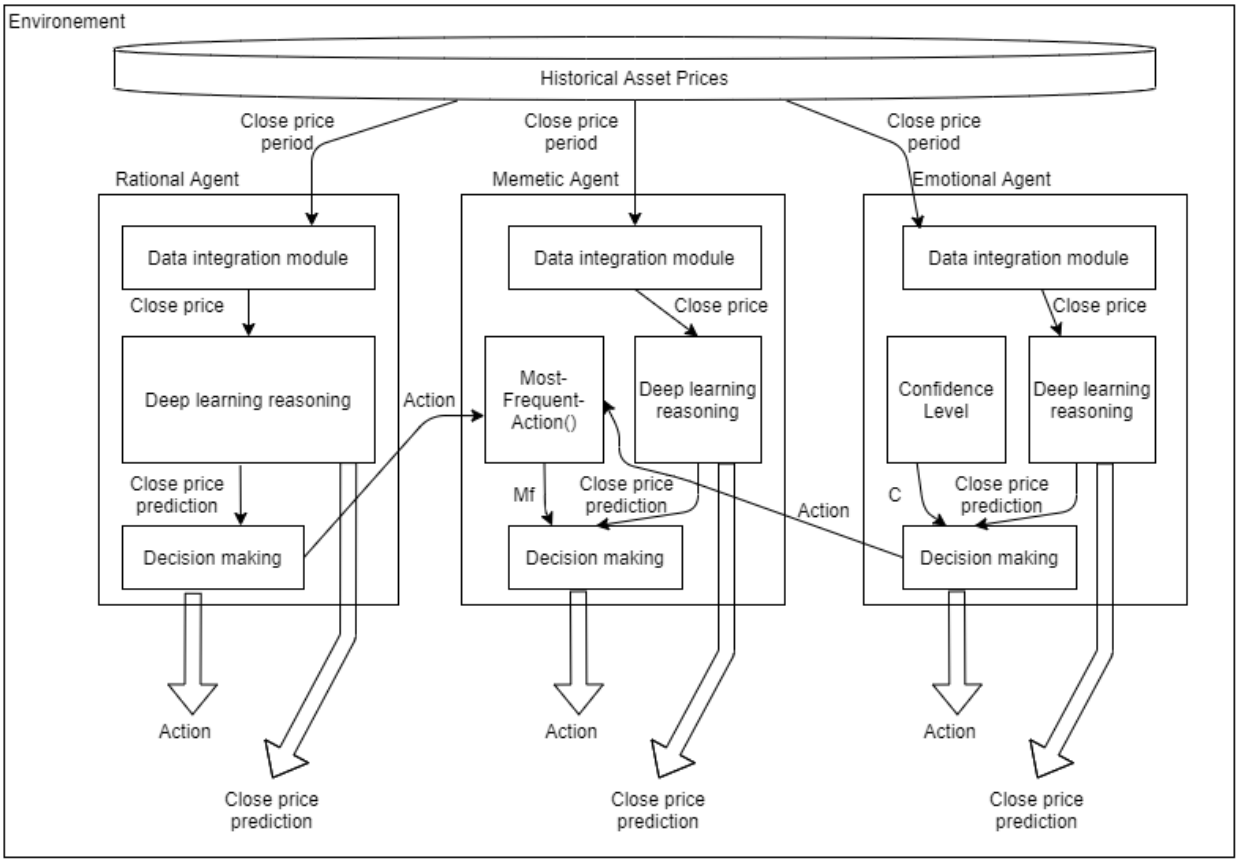

Figure 16: Intelligent Stocks Simulator Architecture

The environment is a marketplace for trading assets that contains historical prices and traders as AI agents.

There are 3 different agent types that represent traders which are rational, emotional and memetic.

The interaction of the agents with their environment leads to close price prediction and decision help (buy, hold, sell).

Based on the output of the simulation we will be able to explain the market composition during a stable period and a crisis period, and the main reasons for a crisis formation.

The goal of this study is to model a deep learning reasoning and to involve it into cooperative multi-agents behaviors. The aim of this study is to model a decision support system based on a "deep Learning" algorithm, to create a complex architecture of cooperative behaviors based on this model via multi-agent systems, and to evaluate the impact of collaborative behavior on market stability.

The multiagent system is called “Intelligent Stocks Simulator”, it aims to simulate stock market investors behavior during a stable period versus a crisis period.

In order to achieve our goal we modeled and implemented a multiagent system that is capable of:

- Collecting and processing data from the network.
- Simulating three different agent types (rational, memetic, emotional).
- Predict the close price of a given stock market based on a deep learning method.
- Take action (buy, hold, sell) on a given trading phase.
- Examining the price change based on agents actions.
- Study the combination of the stock market using a price variation method.

In our work, we are going to focus on the closing price of an asset and the short term trend prediction since our agents are going to be trading on a daily basis.

For this to be ideal we are going to develop our own multi agent environment, and we use the LSTM.

LSTMs are used to model univariate time series forecasting problems. These problems consist of a single series of observations and a model is required to learn from the series of past observations to predict the next value in the sequence.[w5]

4.2 The agents models

The agents interacts with the environment to get the historical asset prices of a given trading period, the body is composed of an LSTM neural network that has an output as a prediction of the asset close price during each day of the period and based on that with the help of the decision making module we get the action of the agent (buy, hold, sell).

- Rational agent

This type of agent is 100% confident in the output of the LSTM model within its body, so the decision making is based on the pure output of the neural network.

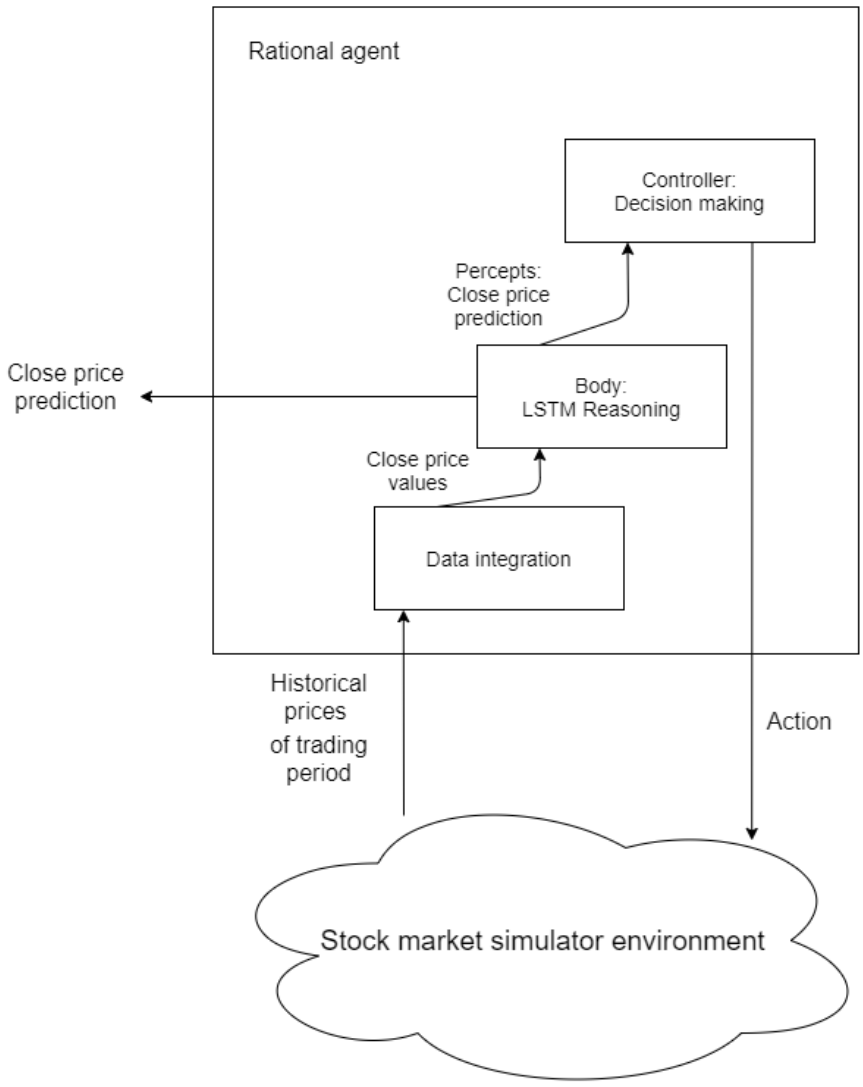

Figure 17: Rational agent architecture

- Emotional agent

This agent type is equipped with a variable “ C ” that defines the confidence level of a trader.

There are 4 levels of confidence (not confident , unsure, super confident, exaggerating) each level is represented with a Float variable. This parameter plays a huge role in the nature of the decision making process where it can lead the agent to exceed the threshold and take actions that are totally emotional.

These levels of confidence do not include the confident value “C=1” because the emotional agent starts acting like a rational agent.

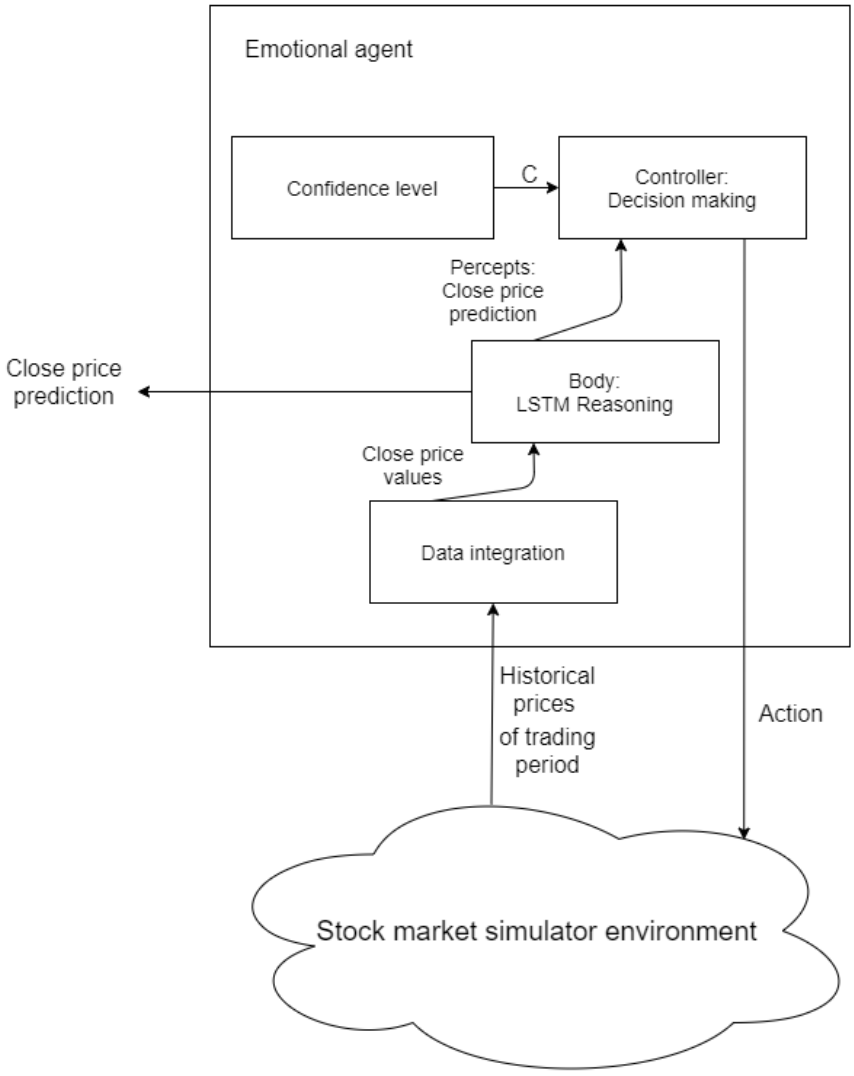

Figure 18: Emotional agent architecture

- Memetic agent

This agent has a method called Most_frequent_action( ) which determines the decision that a trader will make.

The agent processes data using its LSTM neural network and takes a primary decision, then it applies the most frequent method on the actions viewed from the environment including its own primary actions and takes a final decision based on what the majority of other agents decided.

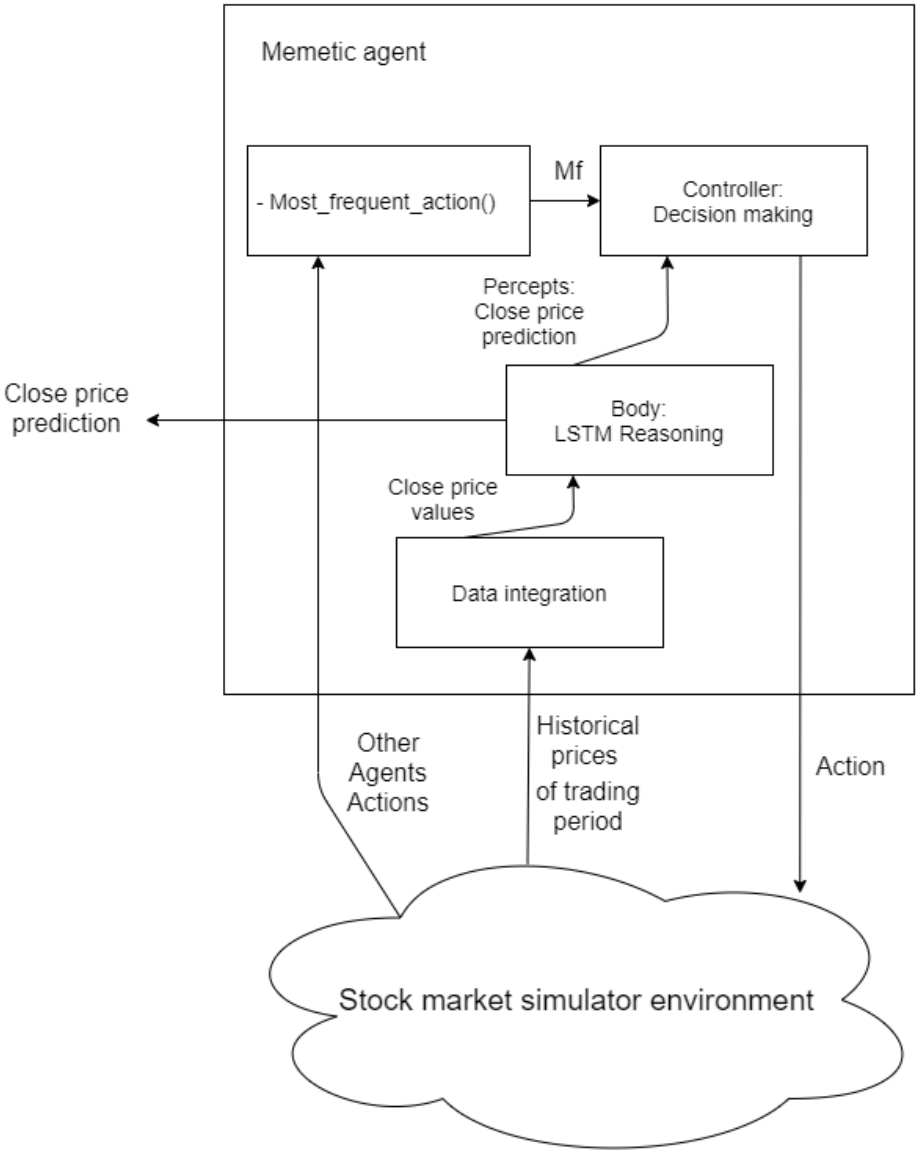

Figure 19: Memetic agent architecture

4.3 Structure of the LSTM

Long Short Term Memory cells are like mini neural networks designed to allow for memory in a larger neural network.

This is achieved through the use of a recurrent node inside the LSTM cell. This node has an edge looping back on itself with a weight of one, meaning at every feedforward iteration the cell can hold onto information from the previous step, as well as all previous steps.

Since the looping connection's weight is one, old memories won't fade over time like they would in traditional RNNs.

LTSMs and recurrent neural networks are as a result good at working with time series data due to their ability to remember the past. By storing some of the old state in these recurrent nodes, RNNs and LSTMs can reason about current information as well as information the network had seen one, ten or a thousand steps ago.[w3]

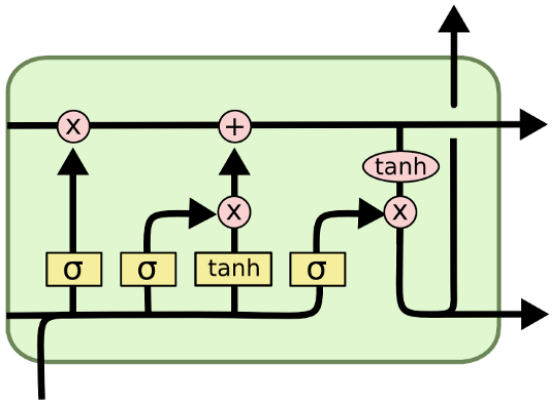

Figure 20: LSTM cell components[w3]

The model has an input layer of 50 lstm cells with a true return-sequence that return the hidden state output for each input time step.

It has a first hidden layer of 50 lstm cells with a true return-sequence.

A second hidden layer of 50 lstm cells with a false return-sequence

A third hidden layer which is a dense layer with 25 neurons.

Finally an output layer which is a dense layer with 1 neuron.

Table 3: Model summary

```
Loaded model from disk
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 60, 50)            10400
_________________________________________________________________
lstm_1 (LSTM)                (None, 60, 50)            20200
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense (Dense)                (None, 25)                1275
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 26
=================================================================
Total params: 52,101
Trainable params: 52,101
Non-trainable params: 0
_________________________________________________________________
```

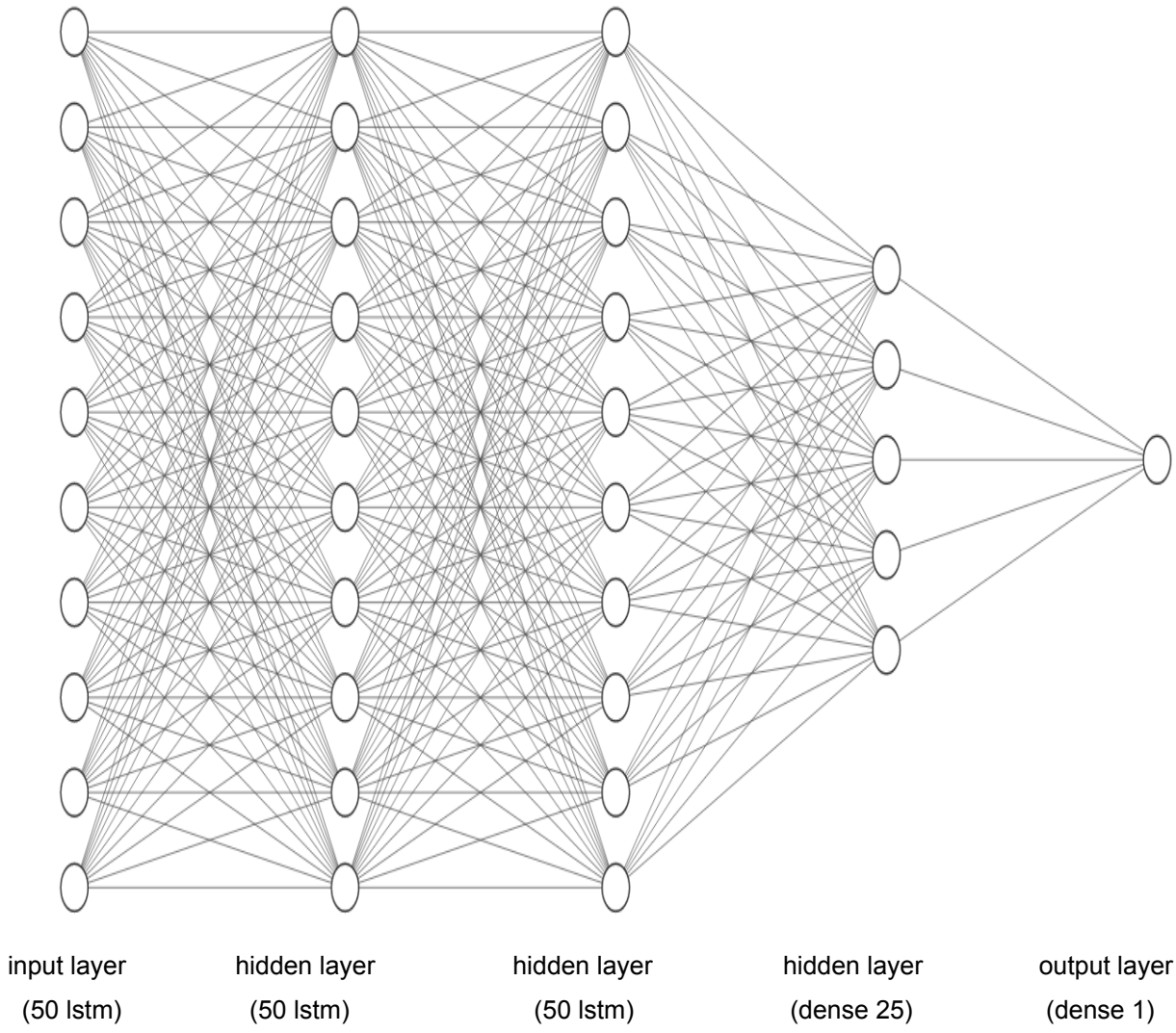

Figure 21: Neural network architecture

4.4 DL Environment

To achieve our goals we develop our own deep learning environment which can host our agents to simulate the trading phase and manage the inputs and outputs of the agents.

We decided to create our own environment because some of the existing solutions do not meet our requirements. We started the development using the “openAI gym” environment which is a toolkit for developing and comparing reinforcement learning algorithms, but it is poorly documented so making a custom environment is a hustle.

4.4.1 General environment

The environment consists of three levels, Data collection to get the historical asset price, agents to process data and train on it to give a prediction and a third level that runs the simulation for two periods, a stable versus a crisis.

4.4.2 Action space - state space

Each agent has three possible actions to take: HOLD,BUY,SELL. see “Figure 1”.

So the agent buys when he thinks that he will be making a profit, he sells when he thinks he is making a profit or avoiding a loss, he holds when he thinks that he will not make profit or he is not obliged by a loss.

The hold action is doing nothing yet it is considered as an action.

Each different action taken puts the environment in a new state.

4.5 Stock Market Simulator Components

Each agent consists of three layers, the first layer is the data integration layer which is made to process cloud historical prices from the environment and refine it to its needs.

The second layer is a deep learning neural network, in our case we use LSTM, which makes the agent able to predict the closing price of a given asset.

The third layer is an optimization layer which allows us to get realistic and accurate predictions that will help taking actions when running a trading simulation.

4.5.1 Agents types

Predicting stock market prices is a complex problem, so we tried to outangle the subject by creating three types of agents each one of them represents an investor.

- **Rational agent:**

It represents a well experienced trader that knows the domain and all its fundamentals.

It is equipped with a deep learning neural network with an LSTM structure.

The agents train on the Dataset and give a prediction based on the last 60 days of historical prices, Then it takes an action (BUY, HOLD, SELL).

- **Emotional agent:**

It represents an intermediate experienced trader that follows the domain with a limited knowledge.

It is equipped with the same neural network as all the agents, on top of that it has an emotion coefficient to simulate the behavior of an emotional trader that has some level of confidence (agent is confident, agent is more confident, agent is not confident, agent is less confident).

The agents train on the Dataset and give a prediction based on the last 40 days of historical prices, Then it takes an action (BUY, HOLD, SELL).

- **Memitic agent:**

It represents a new domain trader with no experience with basic knowledge.

It is equipped with the same neural network as all the agents.

The agents train on the Dataset and give a prediction based on the last 20 days of historical prices, Then it takes an action (BUY, HOLD, SELL) based on the most taken action among the other group of agents, in theory the mimetic agents always causes the stock markets trading to shift quickly in prices whether it drops or it elevates.

Summary

In this chapter we gave a detailed description of the proposed method and the techniques used to achieve our goals, in the next chapter we will clarify our experiments and results.

CHAPTER5 EXPERIMENTS AND RESULTS

Introduction

In this chapter we will discuss the implementation of our method, also the experiment setup and the test environment, then the results of simulations.

5.1 Test environment

5.1.1 Google colab

While developing our solution we used Google Colab or Colaboratory, it is a cloud service, offered by Google (free), based on Jupyter Notebook and intended for training and research in machine learning.

This platform allows you to train machine learning models directly in the cloud.

So without needing to install anything on our computer except a browser.

Colab supports many popular machine learning libraries which can be easily loaded in your notebook, it gives us access to GPU and it is totally free.

5.1.2 Programming language

Python is a multipurpose programming language, it is well known among Ml and AI developers due to its wide range of artificial intelligence and machine learning libraries support.

In this context we use python as a programming language and we take advantage of libraries which are Keras and TensorFlow, also other libraries such as numpy, pandas, random, scipy, matplotlib, collections.

5.2 Experiment setup

5.2.1 Hyper-parameter tuning

Hyperparameter tuning is the process of determining the right combination of hyperparameters that allows the model to maximize model performance.

Setting the correct combination is the only way to extract the maximum performance out of models.

Previously we discussed the architecture of the model and some parameters are given, In the table below we provide all choices made regarding the training phase:

Table 4: hyper-parameters tuning

Training Data	Test Data	batch-size	epochs	Threshold
80%	20%	1	10	0.015

we divide data to 80% training and 20% test, with a batch-size of 1 which means the model trains line by line from the historical price dataset which has proven to improve the precision of the model, finally epochs = 10 which means that the training is repeated 10 times to refine the accuracy.

The threshold parameter is the point where the agent decides to take an action, the agent compares the error between today's asset closing price and tomorrow's closing price, in this case the agent takes action when there is a 1,5% change in price.

5.2.2 Neural network and MAS

With the architecture discussed in "Figure16" the optimizer used to process changes during the learning phase is the Adam optimizer and it is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

The multi agent environment contains:

- The data to be observed by agents is the historical asset closing price, the asset is variable, and it is extracted from Yahoo Finance.[w8]

- The number of each type of agent is present in the environment and we have three types.
- The simulation will be executed on a variable interval of days.

- The output, The actions taken by agents after the trading simulation is done.

5.2.3 Simulation Cycle

The process that the system performs to simulate an episode is described in algorithm 3 .

The simulation cycle is executed as many times as the number of episodes set for an experiment.

Algorithm 3: simulation cycle for stock trading.

```
#Running the trading
for i in range(0, len(dayslist)):
      for ri in range(0, ragent):
            action = self.getAction(Ragents[ri], dayslist[i], dayslist[i])
            Ractions.append(action)
            ra.append(action)
      for ei in range(0, eagent):
            action = self.getAction(Eagents[ei], dayslist[i],dayslist[i])
            Eactions.append(action)
            ea.append(action)
      for mi in range(0, magent):
            REList = list(itertools.chain(Ractions, Eactions))
            mf = max(set(REList), key = REList.count)
            Mactions.append(mf)
            ma.append(mf)
      actions = list(itertools.chain(ra, ea, ma))
      newprice = (real + (x*(buy - sell)))
      newpricelist.append(newprice)
      real = newprice
```

The getAction() method is the process that every agent does to predict a given day's asset price prediction and calculate the error between the real price and the predicted price and finally compare it to a threshold point to decide whether to (Hold, Buy, Sell).

Algorithm 4: taking action by agents algorithm.

```
#Threshold
e = (pred_price - real_price)/real_price
#Action
if (agent.id == 'r'):
  if e<-0.015:
    action = "Sell"
  elif e>0.015:
    action = "Buy"
  else:
    action = "Hold"
  return action
#Action
elif (agent.id == 'e'):
  C = agent.C
  e = e * C
  if e<-0.015:
    action = "Sell"
  elif e>0.015:
    action = "Buy"
  else:
    action = "Hold"
  return action
```

5.3 Results

In this section we will be showing results from simulations, and compare the output to reach an optimized parameters input.

5.3.1 Comparison

Hidden layer comparison

To discover the best configuration we tried three different numbers of hidden layers with the same rest of parameters for the neural network. see “Figure 21”.

We trained an algorithm with the same parameters but with different structure:

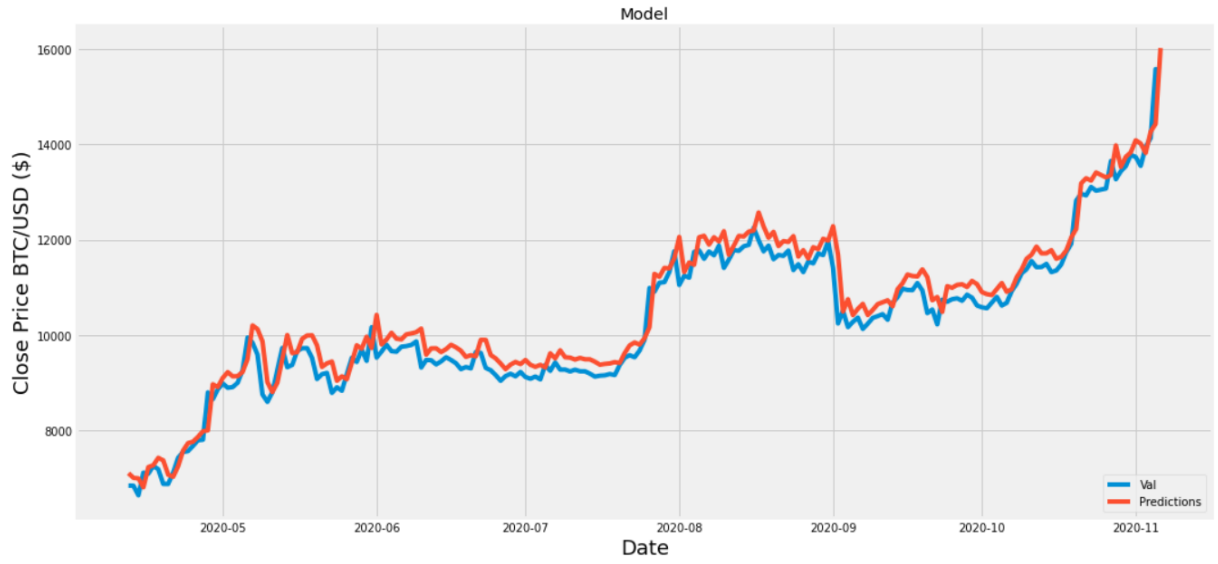

Figure 22: BTC/USD prediction with 2 hidden layers

In this figure we can see the real historical bitcoin close prices by the blue color and our “Stock Market simulator” predicted bitcoin close price by the red color.

We can see a great resemblance of our predicted values and the real values.

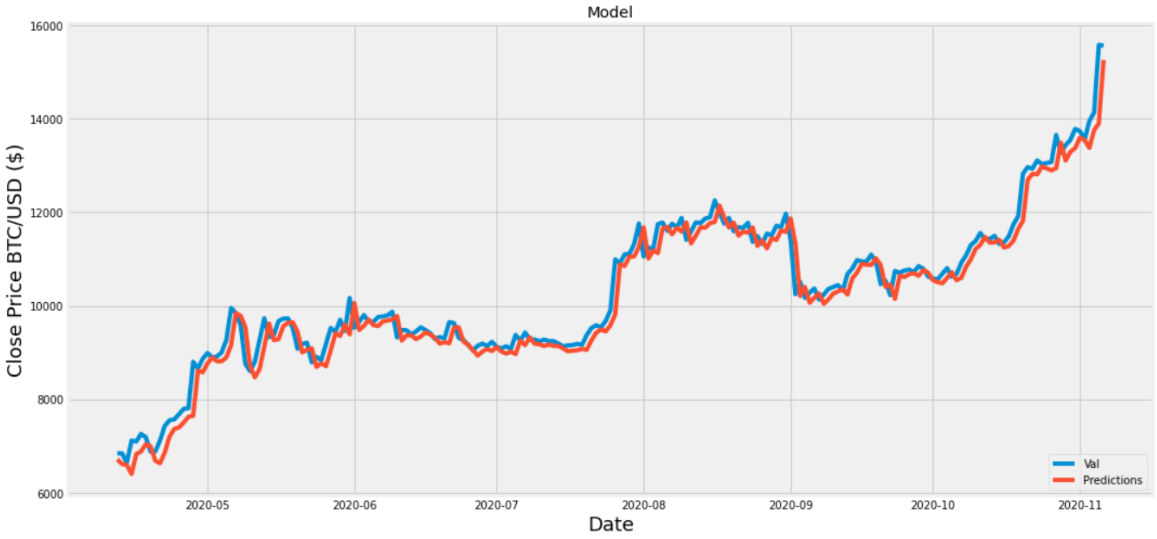

Figure 23: BTC/USD prediction with 3 hidden layers

The same can be said about this figure where the predicted values shown by red color are close to the real values shown by blue color.

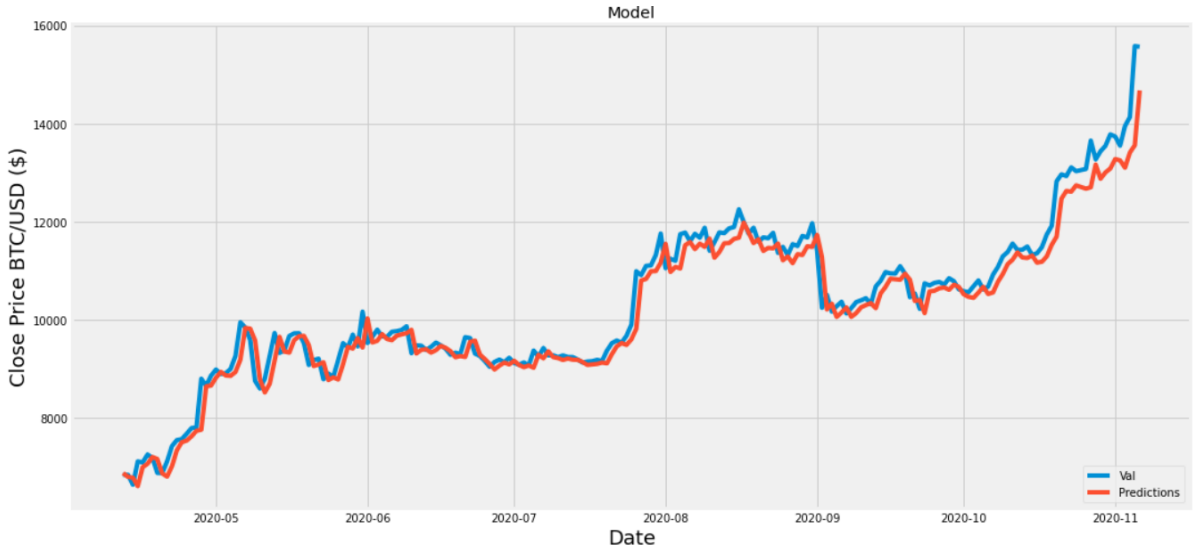

Figure 24: BTC/USD prediction with 4 hidden layers

In this figure we tried using four hidden layers in our neural network to compare the output with the past figures.

We calculated the RMSE value for each neural network structure, the lower the value the better the model.

Table 5: RMSE for each NN structure

Number of hidden layers	Root Mean Squared Error
2	3.65
3	3.28
4	3.57

So, the best Neural Network structure is the one with 3 hidden layers due to its lower RMSE value.

We can now fix the number of hidden layers on 3 layers and move on to agree on other parameters, the next comparison will be the batch-size and number of epochs.

Batch size vs epochs number comparison

The batch-size is the steps in the quantity of data to be processed in the learning phase.

The epochs number is how many times we will repeat the training until we get a low RMSE value.

So we tried 3 different epochs numbers with a fixed batch-size, and the same for the batch-size with a fixed epochs number.

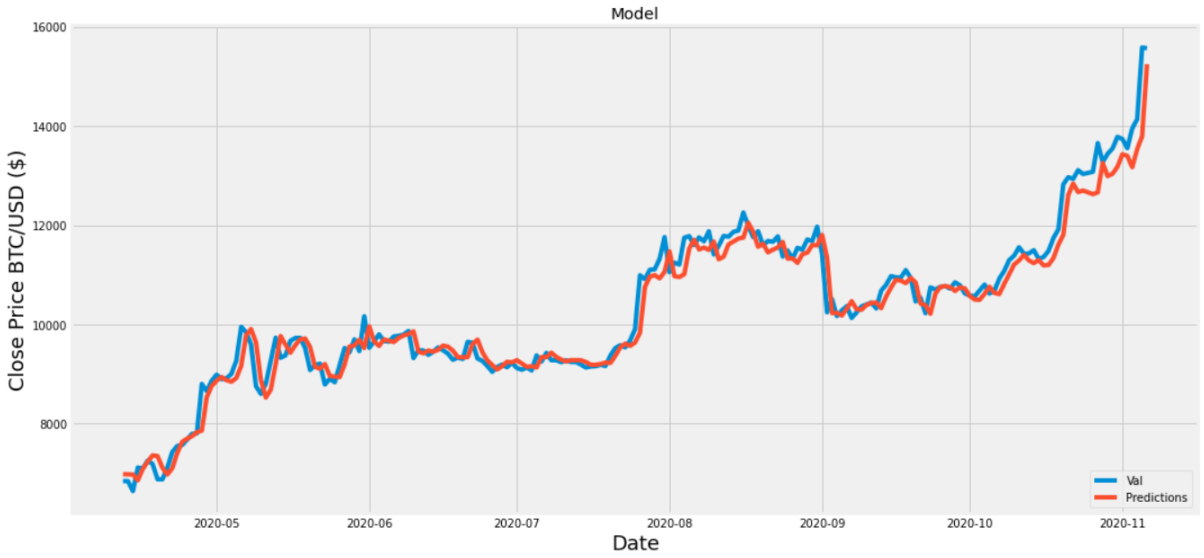

Figure 25: BTC/USD Price prediction with batch-size=1 and 5 epochs

In this figure we are gathering data from the output of the neural network when repeating the training process 5 times with each ligne of the dataset being processed.

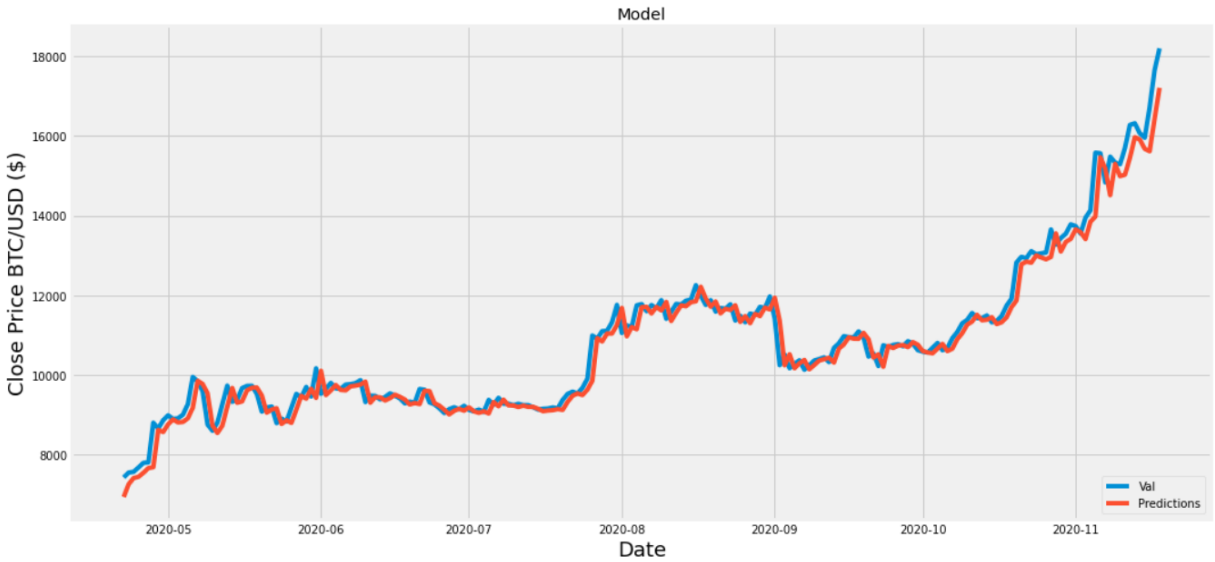

Figure 26: BTC/USD Price prediction with batch-size=1 and 10 epochs

In this figure we are gathering data from the output of the neural network when repeating the training process 10 times with each ligne of the dataset being processed.

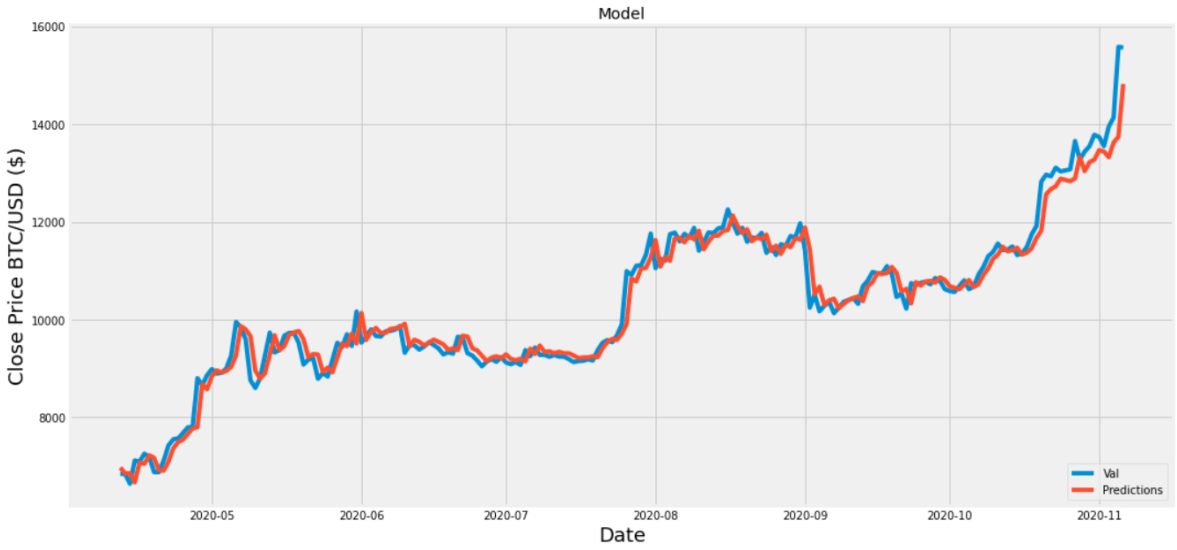

Figure 27: BTC/USD Price prediction with batch-size=1 and 15 epochs

In this figure we are gathering data from the output of the neural network when repeating the training process 15 times with each ligne of the dataset being processed.

After all these experiments we need to calculate the root mean squared error to determine the best accurate configuration for the neural network structure.

We gathered the Root Mean Squared Error data in a table to distinguish the best value for the epochs number and apply it to the model.

Table 6: RMSE values according to epochs number

Epochs Number	RMSE
5	3.28
10	3.14
15	3.36

When going from 5 epochs to 10 epochs we notice a decrease in RMSE by 3.79% which is a good boost in performance, but when jumping from 10 epochs to 15 epochs we notice an increase in RMSE which is a sign of overtraining.

We know by experience that if we go past 10 epochs we will be overtraining the model so it is a wise choice to stay there.

After fixing the number of times that our model will be trained, we will tweak the batch-size to distinguish the best configuration.

The next figure will be always showing the real values of the bitcoin close price data versus the “Stock Market Simulator” predicted values, with different batch-size configurations.

The batch-size is the number of lignes from the bitcoin historical close price dataset being processed in each step of the training process.

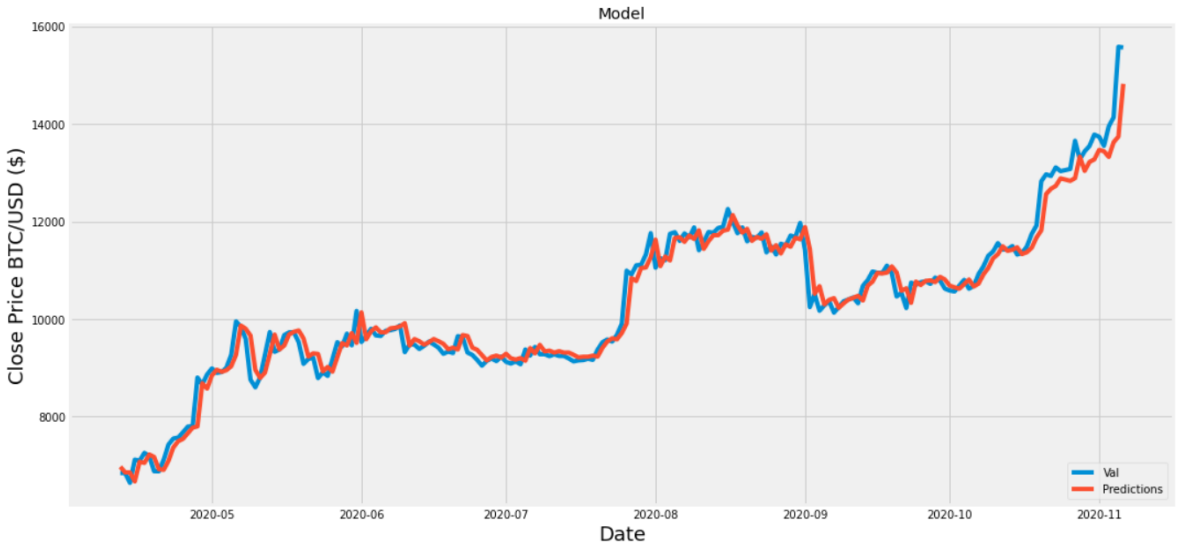

Figure 28: BTC/USD Price prediction with 10 epochs and batch-size=1

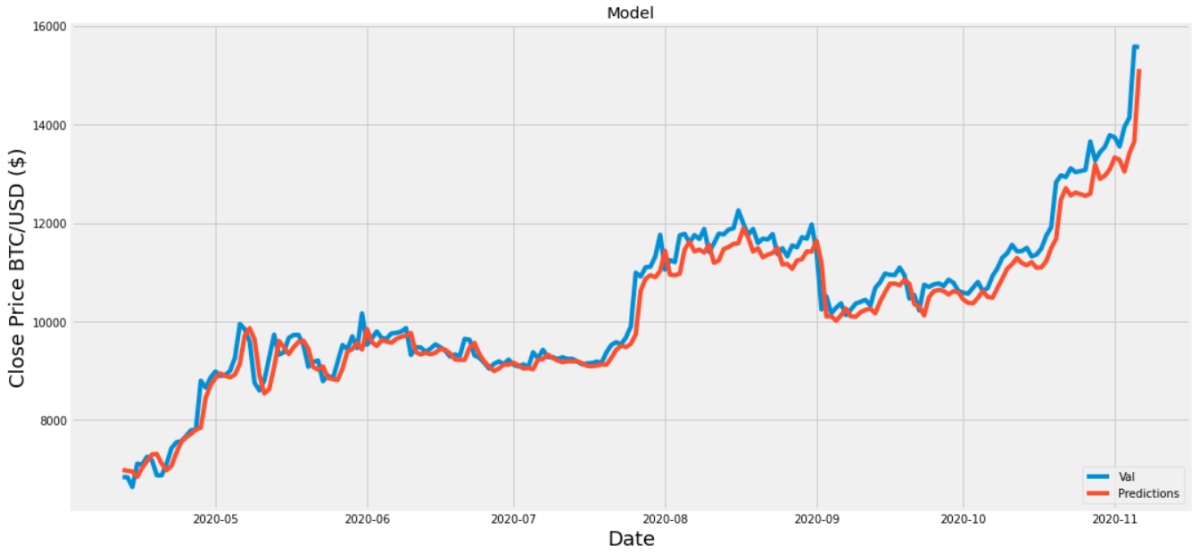

Figure 39: BTC/USD Price prediction with 10 epochs and batch-size=5

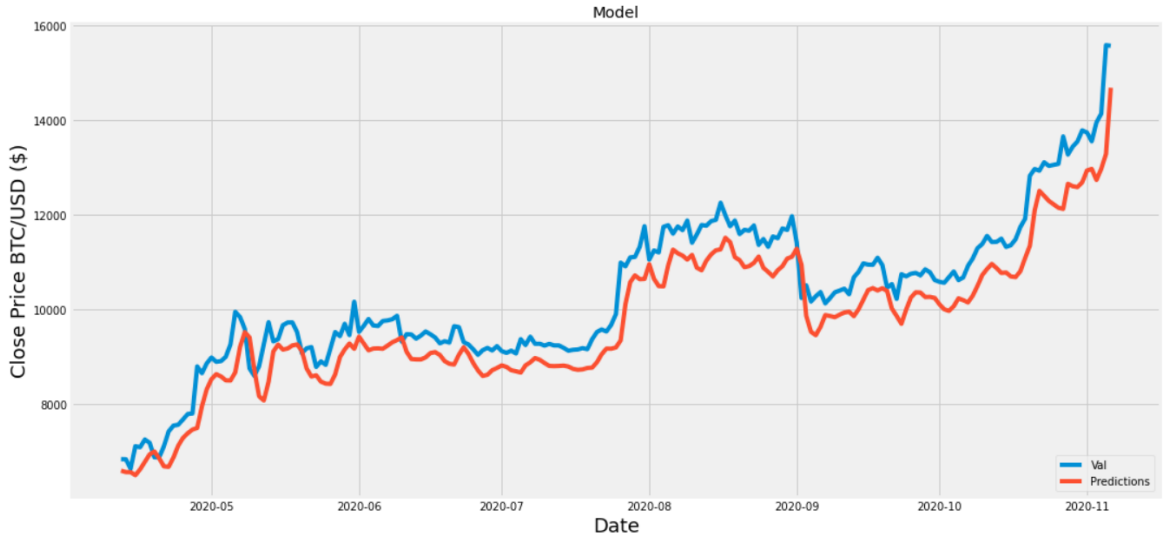

Figure 30: BTC/USD Price prediction with 10 epochs and batch-size=10

Data is collected and as usual we will be comparing the RMSE values to get the best configuration.

Table 7: RMSE values according to batch-size

Batch-size	RMSE
1	3.14
5	3.73
10	6.52

The lower the batch-size the better the model, so we fixed the batch-size value to be equal to 1.

Our model is optimized and ready now it is time to train our three agents and launch the trading simulation.

5.3.2 Stock market trading

In all of the next scenarios we will be measuring our model performance in the real world, by simulation trading phases over a stable period versus a crisis period to see if our “Stock Market Simulator” meets the reality.

Configuration

5 rational agents, 5 emotional agents, 5 mimetic agents(mixed confidence levels).

Period 30 days.

Asset BTC/USD.

Scenario 1

Stable Period

A stable period is where the price of the asset doesn’t change in a drastic way and there is a balance between the supply and demand.

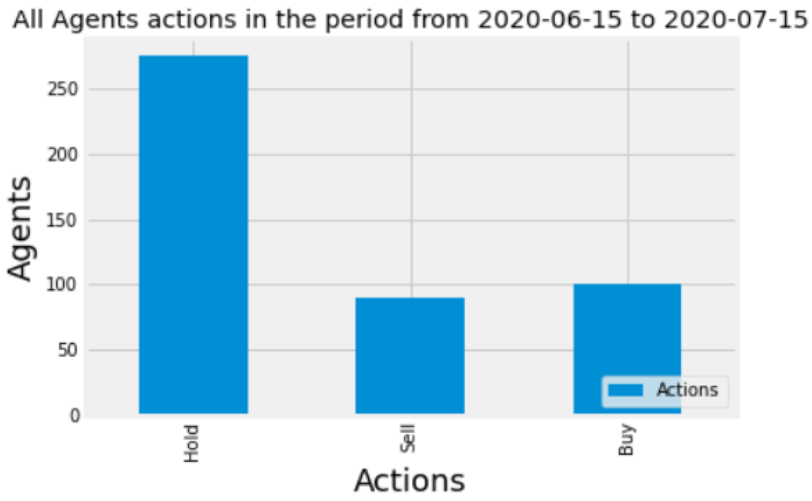

Figure 31: Agents trading in stable month

The figure 31 shows all of the 15 agents actions in the stable month, to better investigations we grouped actions by agent type.

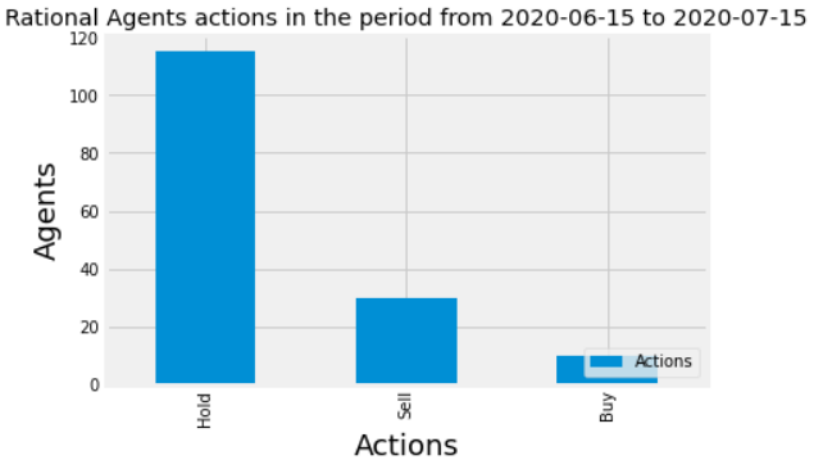

Figure 32: Rational agents trading in stable month

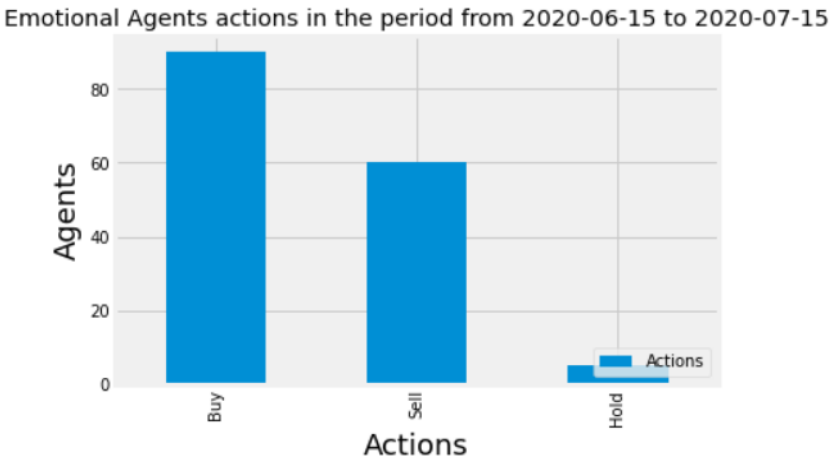

Figure 33: Emotional agents trading in stable month

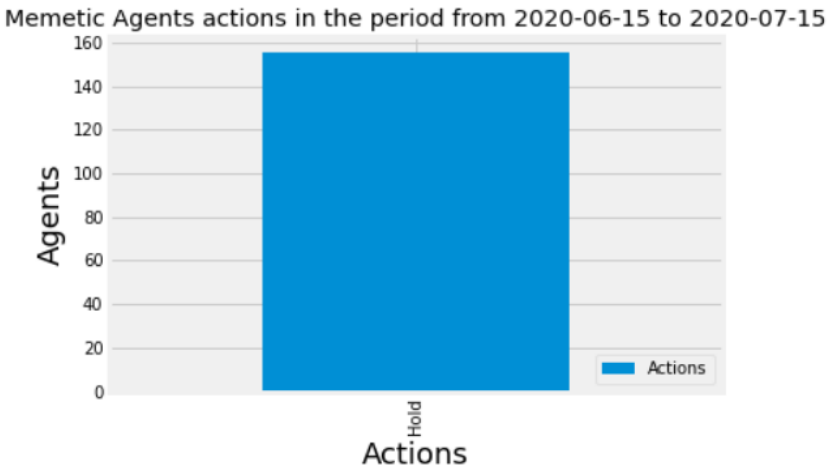

Figure 34: mimetic agents trading in stable month

We can see clearly that the agents actions are different where the rational agent is taking the holding action more than the buy and sell, we explain this by the core of the agent which is going by statistics only to take action.

The emotional agent is more active where it goes by the emotions factor, it doesn't take action rationally so it can go out of the threshold and take a sell or buy action different from the hold action taken by the rational reasoning.

The memetic agent is dominated by the most frequent action which is holding, this type of behavior can lead a stock market into a crash where the price drops dramatically and the selling action is elevated.

The simulator was able to accurately meet the reality of trading in a stable period, where there is a balance in actions taken by agents in this scenario.

Scenario 2

Crisis Period

A crisis period in stock trading is when the asset price drops dramatically in a short period of time, also known as market crash because of the fast actions taken by traders due to the fast change in price.

A crisis can be caused by a lot of factors, one of these factors is the diseases such as the COVID-19 virus that we had this year, it caused a market crash from february 2020 until april 2020.

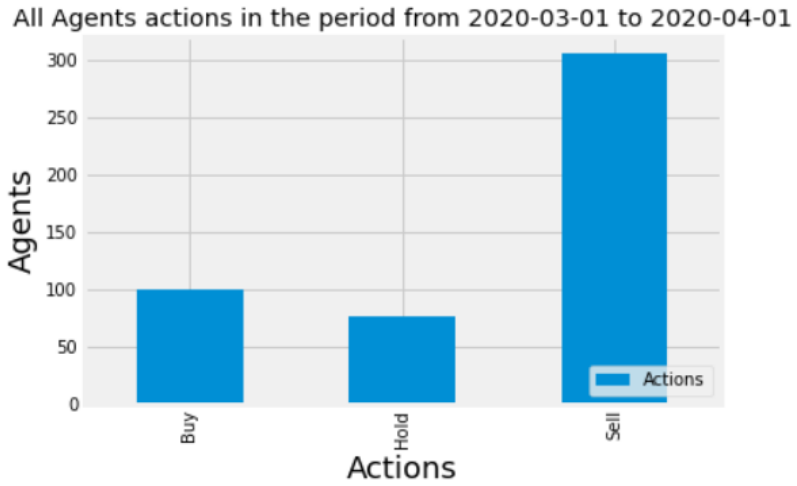

Figure 35: Agents trading in crisis month

For better understanding we will be grouping actions by agent type.

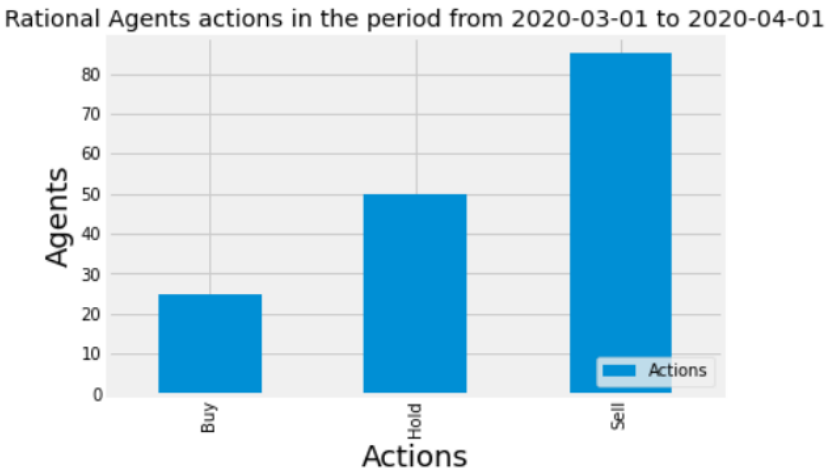

Figure 36: Rational agents trading in crisis month

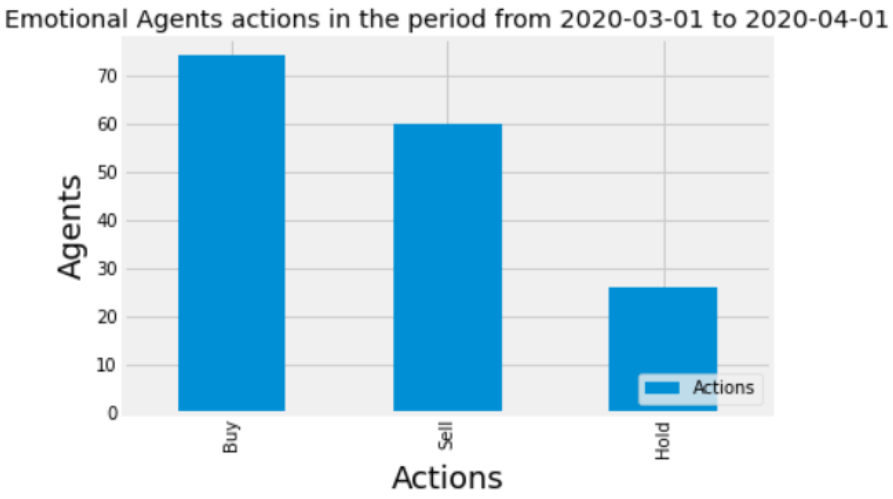

Figure 37: Emotional agents trading in crisis month

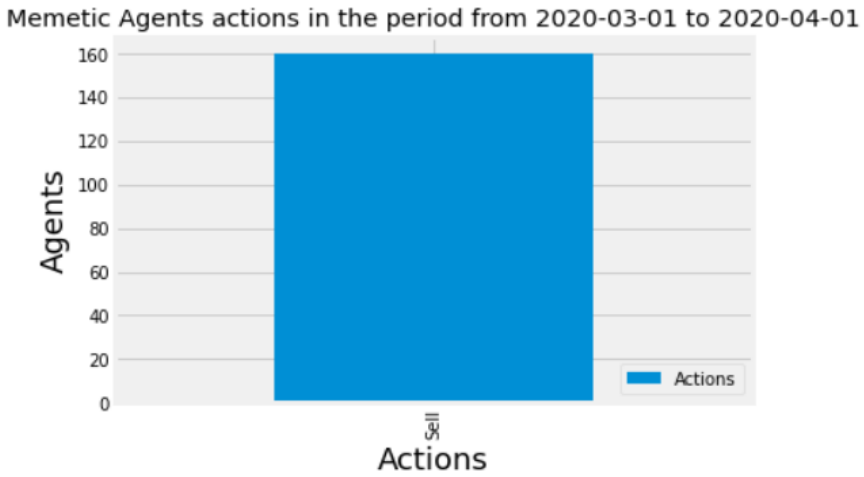

Figure 38: mimetic agents trading in crisis month

During the covid-19 pandemic the simulator has proven potential where it predicted the stock crash, and we can see that in the high selling action compared to the buying due to the fast drop of price.

As we mentioned before in a crisis period the stock price decreases dramatically due to numerous causes, in our study we focused on the COVID-19 crisis, it is clear that the selling action is taken the most by all agents except the emotional agent which is again going by emotions so it can go out of boundaries and take a

rationally wrong decision but it can lead to a profit, this is called High-Risk,High-Reward Strategy in stock trading.

5.3.3 Studying the price variation

In this section we will be looking at the real stock market traders behaviour combination to understand the balance between three types of behaviour (Rational, Emotional, Memetic).

- Hypothesis :

There is a mixture of three types of behaviour in the real world, there is no dominating side because in that case the stock market won't be functional where there are different approaches taken to let traders make gains.

now we will be applying the price variation based on the following equation:

eq 5: $P(t+1)=P(t)+\partial$ (buyin_actions(P(t)) -Selling_actions(P(t)) [20]

$\partial \leq 0.5$

buying_actions(P(t)) : The security buying at price P(t) over period t.

Selling_actions(P(t) : The security supply at price P(t) over period t.

by following this equation we can find the composition of the market regarding the agents types.

We will use the Mean Absolute Error “MAE” to calculate the precision of the output.[w6]

The lowest “MAE” configuration will be taken as the most accurate one.

We will be studying four different scenarios to conclude the closest one to reality.

Scenario 1

Equal sized agents:

33.5% rational agents, 33.5% emotional agents, 33.5% mimetic agents.

over the first 10 days of january 2020.

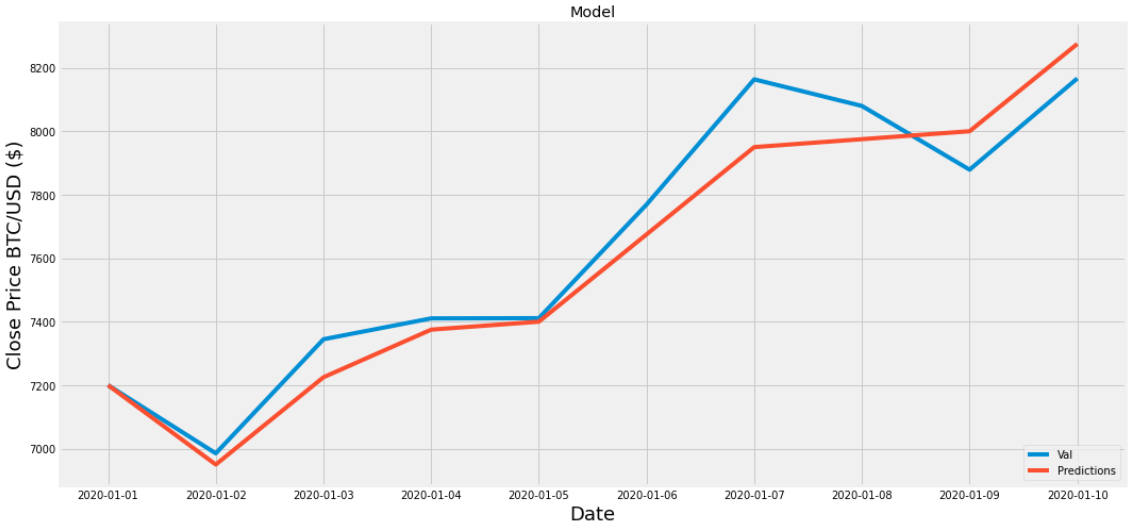

Figure 39: BTC close price variation Equal sized agents

The blue color in the graph is the real data of bitcoin close price versus the red one is the simulator bitcoin close price prediction using the price variation equation.

MAE: 84.36%, we can see that there is a strong resemblance between the real values and the predicted values.

Scenario 2

Rational dominated stock:

60% rational agents, 20% emotional agents, 20% mimetic agents.

over the first 10 days of january 2020.

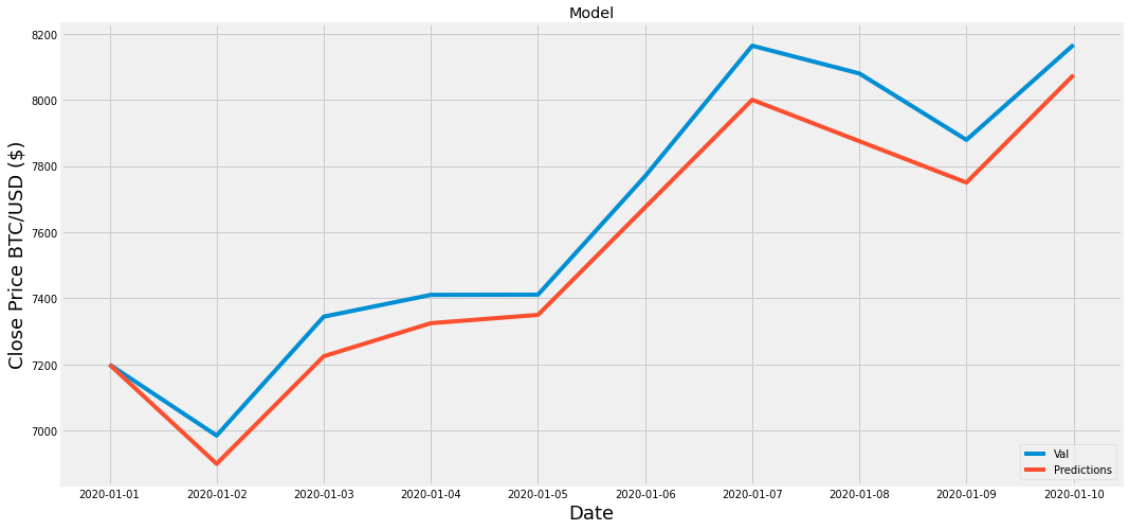

Figure 40: BTC close price variation rational dominated

MAE: 103.41%, the figure 40 shows that there is a margin of error between the predicted and the real values.

Scenario 3

Emotional dominated stock:

20% rational agents, 60% emotional agents, 20% mimetic agents.

over the first 10 days of january 2020.

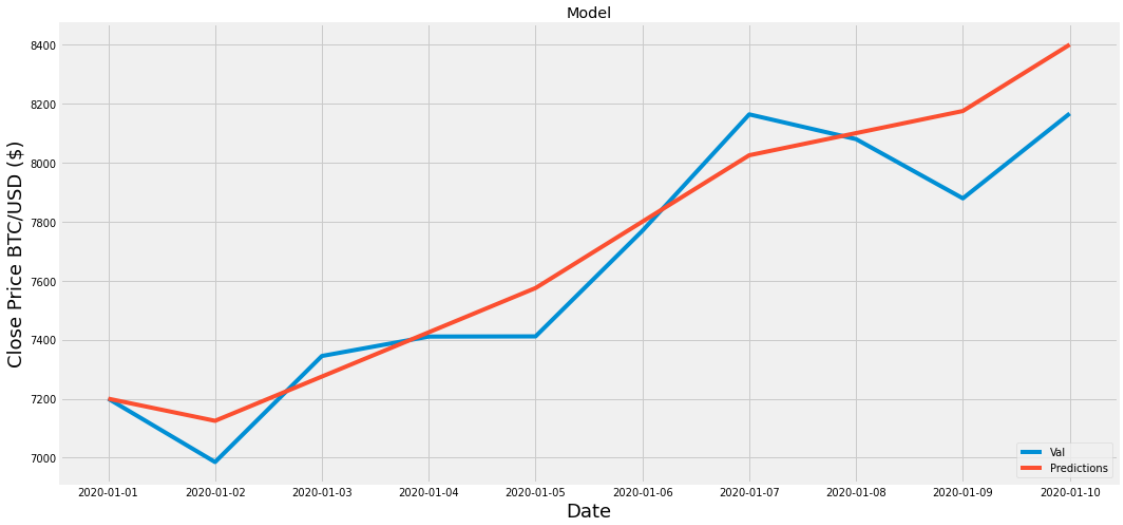

Figure 41: BTC close price variation emotional dominated

MAE: 110.73%, the figure 41 shows that the predicted values are much more different from the real values.

Scenario 4

Memetic dominated stock:

20% rational agents, 20% emotional agents, 60% mimetic agents.

over the first 10 days of january 2020.

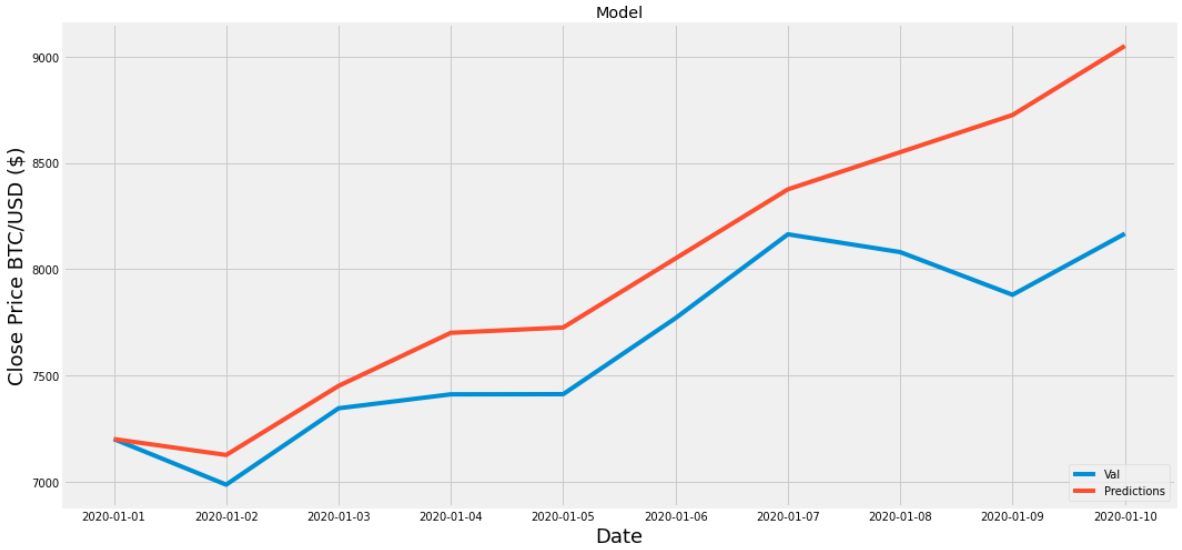

Figure 42: BTC close price variation memetic dominated

MAE: 354.08%, the figure 42 shows that the predicted and the real values are totally different.

We compare results from the four scenarios and we conclude that the closest composition to the reality is the first scenario because it has the lowest mean absolute error value among all the scenarios.

We can say that the reality of the stock market is a balanced structure where there is a balance in traders types of behaviour.

We believe that the output of the simulation is reasonably successful especially by validating it by calculating the Mean Absolute Error and finding the lowest value.

Summary

In this chapter we showed the results of our Intelligent Stocks Simulator.

In the next section we will conclude and give some perspective for future studies and what could be done to improve our work.

GENERAL CONCLUSION AND PERSPECTIVE

We are glad to say that our “Intelligent Stock Simulator” has showed promising results and it was successful in explaining the stock market crash causes, the approach of integrating the behavior of traders (Rational, Emotional, Mmetic) in our study is the key to success where we explained the composition of the stock market.

From the results in the previous chapter we can see that in the stable period the actions taken by agents were ideal for a stable stock market.

The agents are holding, with a bit of selling and buying when they find a profit.

The mimetic agents are following the most frequent action, this behavior can lead to a stock crash.

In the crisis period we can clearly see the stock market is shifting to the selling action due to the sudden fast drop in price.

As usual, the mimetic agents are causing the trading to shift to the most frequent action, which is the selling action due to the drop in price so agents are selling to minimize their loss.

We conclude that the model, see “Table 3”, matches with the market reality and we can use it for multiple purposes, whether we want to predict an asset price, study the trading in a period of time or get decision help.

We can also deduct that the worst thing to happen in a stock market is the rise of mimetic behavior traders due to their primary cause of stock crashes.

We can’t also confirm that the best behavior to take in both periods is the rational behavior, even with the robust decisions taken.

A trader can be emotional and get more profit than a rational colleague, this is caused by the fact that the stock markets are impossible to predict with a 100% accuracy, so we can take actions based on emotions and it can be the opposite of what a rational trader would take but in the end it could be more profitable.

For future work we suggest the following ideas:

- Try with different approaches (other deep learning technologies) and try to add on real time the price prediction so it can help investors to know when to buy and when to sell exactly during the 24 hours.

- Integrate the news factor and social media to the input of the model and modify it to get a better prediction.

- Make a more active simulation by giving each agent a capital and let it trade for a period of time, finally find the profits made by agents to know which behavior is the best and in what circumstances.

BIBLIOGRAPHIC REFERENCE

[1] Heaton, J., & Lucas, D. Stock Prices and Fundamentals. NBER Macroeconomics Annual, 14, 213–242. doi:10.1086/654387 - 1999.

[2] O Blanchard, C Rhee, L Summers. The stock market, profit, and investment. The Quarterly Journal of Economics - 1993.

[3] L Qi, M Khushi, J Poon. Event-driven LSTM for forex price prediction. arXiv preprint arXiv:2102.01499 - 2021.

[4] RA de Oliveira, HS Ramos, DH Dalip, ACM Pereira. A Tabular Sarsa-Based Stock Market Agent. researchgate.net - 2020.

[5] DW Lu. Agent inspired trading using recurrent reinforcement learning and lstm neural networks. arXiv preprint arXiv:1707.07338 - 2017.

[6] L Troiano, EM Villa, V Loia. Replicating a trading strategy by means of LSTM for financial industry applications. IEEE transactions on industrial,ieeexplore.ieee.org - 2018.

[7] WU, J., WANG, C., XIONG, L., & SUN, H. Quantitative Trading on Stock Market Based on Deep Reinforcement Learning. International Joint Conference on Neural Networks (IJCNN). doi:10.1109/ijcnn.2019.8851831 - 2019.

[8] Hansson, Magnus LU. On stock return prediction with LSTM networks.NEKN01 20171 Department of Economics - 2017.

[9] R. Hafezi, J. Shahrabi, E. Hadavandi, A Bat-Neural Network Multi-Agent System (BNNMAS) For Stock Price Prediction:Case Study of DAX Stock Price,Applied Soft Computing Journal - 2015.

[10] R.S.T. Lee Department of Computing, Hong Kong Polytechnic University, Hung Hom, Hong Kong, China IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans (Volume: 34, Issue: 3) - 2004.

[11] X Zhou, Z Pan, G Hu, S Tang, C Zhao. Stock market prediction on high-frequency data using generative adversarial nets. Mathematical Problems in Engineering, hindawi.com - 2018.

[12] L Wang, Z Wang, S Zhao, S Tan. Stock market trend prediction using dynamical Bayesian factor graph. Expert Systems with Applications, sciencedirect.com - 2015.

[13] Carta, S., Ferreira, A., Podda, A. S., Recupero, D. R., & Sanna, A. Multi-DQN: an Ensemble of Deep Q-Learning Agents for Stock Market Forecasting. Expert Systems with Applications, 113820. doi:10.1016/j.eswa.2020.113820 - 2020.

[14] S Ganesh, N Vadori, M Xu, H Zheng, P Reddy, M Veloso. Reinforcement learning for market making in a multi-agent dealer market. arXiv preprint arXiv:1911.05892 - 2019.

[15] Y Patel. Optimizing market making using multi-agent reinforcement learning. arXiv preprint arXiv:1812.10252 - 2018.

[16] H Yang, XY Liu, S Zhong, A Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. Available at SSRN, papers.ssrn.com - 2020.

[17] XY Liu, H Yang, Q Chen, R Zhang, L Yang, B Xiao, CD Wang. FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance. arXiv preprint arXiv:2011.09607 - 2020.

[18] Z Xiong, XY Liu, S Zhong, H Yang, A Walid. Practical deep reinforcement learning approach for stock trading. arXiv preprint arXiv:1811.07522 - 2018.

[19] Si, W., Li, J., Ding, P., & Rao, R. (2017). A Multi-objective Deep Reinforcement Learning Approach for Stock Index Futures Intraday Trading. 10th International Symposium on Computational Intelligence and Design (ISCID). doi:10.1109/iscid.2017.210 - 2017.

[20] Richard H Day and Weihong Huang. Bulls, bears and market sheep. Journal of Economic Behavior & Organization, 14(3):299–329 - 1990.

[21] Stuart russel, Peter Norvig, Artificial Intelligence: a modern approach.Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305, USA - 1996.

[22] Mazur, M., Dang, M., & Vega, M. COVID-19 and the March 2020 Stock Market Crash. Evidence from S&P1500. Finance Research Letters, 101690. doi:10.1016/j.frl.2020.101690 - 2020.

Webography

[w1] www.Investopedia.com

[w2] finance.zacks.com

[w3] www.arxiv.org

[w4] www.sciencedirect.com

[w5] www.papers.ssrn.com

[w6] www.hindawi.com

[w7] www.ieeexplore.ieee.org

[w8] finance.yahoo.com