UNIVERSIDAD
NACIONAL
DE COLOMBIA

Short-Term Forecasting of Financial Time Series with Deep Neural Networks

Andrés Ricardo Arévalo Murillo

Universidad Nacional de Colombia
Faculty of Engineering, Department of Systems and Industrial Engineering
Bogotá D.C., Colombia
2016

Short-Term Forecasting of Financial Time Series with Deep Neural Networks

Andrés Ricardo Arévalo Murillo

Thesis presented as partial requirement to obtain the degree of:
Master in Systems and Computer Engineering

Advisor:
German Jairo Hernandez Perez, Ph.D.

Research lines:
Applied Computing and Intelligent Systems

Universidad Nacional de Colombia
Faculty of Engineering, Department of Systems and Industrial Engineering
Bogotá D.C., Colombia
2016

This thesis is dedicated to my parents, my sisters and everyone who made this possible...

Abstract

In this work, a high-frequency strategy using Deep Neural Networks (DNNs) is presented. The input information to the DNN consists of: (i). Current time (hour and minute); (ii). the last n one-minute pseudo-returns, where n is the sliding window size parameter; (iii). the last n one-minute standard deviations of the price; (iv). The last n trend indicator, computed as the slope of the linear model fitted using the transaction prices inside a particular minute. The output DNN prediction is the next one-minute pseudo-return, this output is later transformed to obtain the next one-minute average price forecasting. The DNN predictions are used to build a high-frequency trading strategy that buys (sells) when the next predicted average price is above (below) the last closing price.

This high-frequency trading strategy is only applicable to high liquidity stocks, because it requires to open and close positions in a time interval equal or less than one minute. For experimental testing, this work uses three datasets: (i). Apple stock (ticker: AAPL) from September to November of 2008. (ii). Apple stock (ticker: AAPL) from August of 2015 to August of 2016. (iii). Google stock (ticker: GOOG) from August of 2015 to August of 2016. Apple Inc. and Google Inc. are high liquidity stocks.

The period of the first dataset covers the stock crash during the financial crisis of 2008. During this crash, the AAPL price suffered a dramatic fall from 172 to 98 dollars. This first dataset was chosen intentionally for demonstrate the performance of the proposed strategy under high volatility conditions. Whereas the second and third datasets were chosen in order to test the proposed strategy in normal market conditions.

Multiple DNNs with different sliding window size parameter n and number of hidden layers L were trained. The best-performing-found DNN has a 65% of directional accuracy.

Keywords: Short-term Forecasting, High-frequency Trading, Computational Finance, Deep Neural Networks

Contents

List of Figures

Chapter 1

Introduction

Financial Markets modelling has caught a lot of attention during the recent years due to the growth of financial markets and the large number of investors around the world in pursuit of profits. However, modelling and predicting prices of Financial Assets is not an easy work, due to the complexity and chaotic dynamics of the markets, and the many non-decidable, non-stationary stochastic variables involved [19] [6]. Many researchers from different areas have studied historical patterns of financial time series and they have proposed various models to predict the next value of time series with a limited precision and accuracy[16, 18, 22].

Over decades, researchers have used linear methods such as Linear Regression (LR), Exponential Smoothing (ES), Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroscedasticity (GARCH), and non-linear methods such as Hidden Markov Models (HMMs), Artificial Neural Networks (ANNs), among others. Techniques using ANNs, have been the most preferred and the most widely used for forecasting financial time series over the last 20 years, due to an ANN has the ability to extract essential features and to learn complex information patterns in high dimensional spaces [9, 16, 22, 25].

Figure **1-2** shows a brief time line of Artificial Neural Networks and some applications in finance are presented.

Since late 1980s, ANNs have been a popular theme in data analysis. Although ANNs have existed for long time and they have been used in many disciplines, only since early 1990s they are used in the field of finance [15]. Only since 1988, the first known application of ANN in finance "Economic prediction using neural networks: the case of IBM daily stock returns" was published in [28].

ANNs are inspired by brain structure; they are composed of many neurons that have connections with each other. Each neuron is a processor unit that performs a weighted aggregation of multiple input signals, that is, each input is multiplied by a weight w and then added to a bias b. And depending on its inputs, it is activated and propagates a new output signal, through application of a non-linear function such as Unit Step, hyperbolic tangent, Gaussian, Rectified Linear, among others [11]. Figure **1-3** shows the artificial

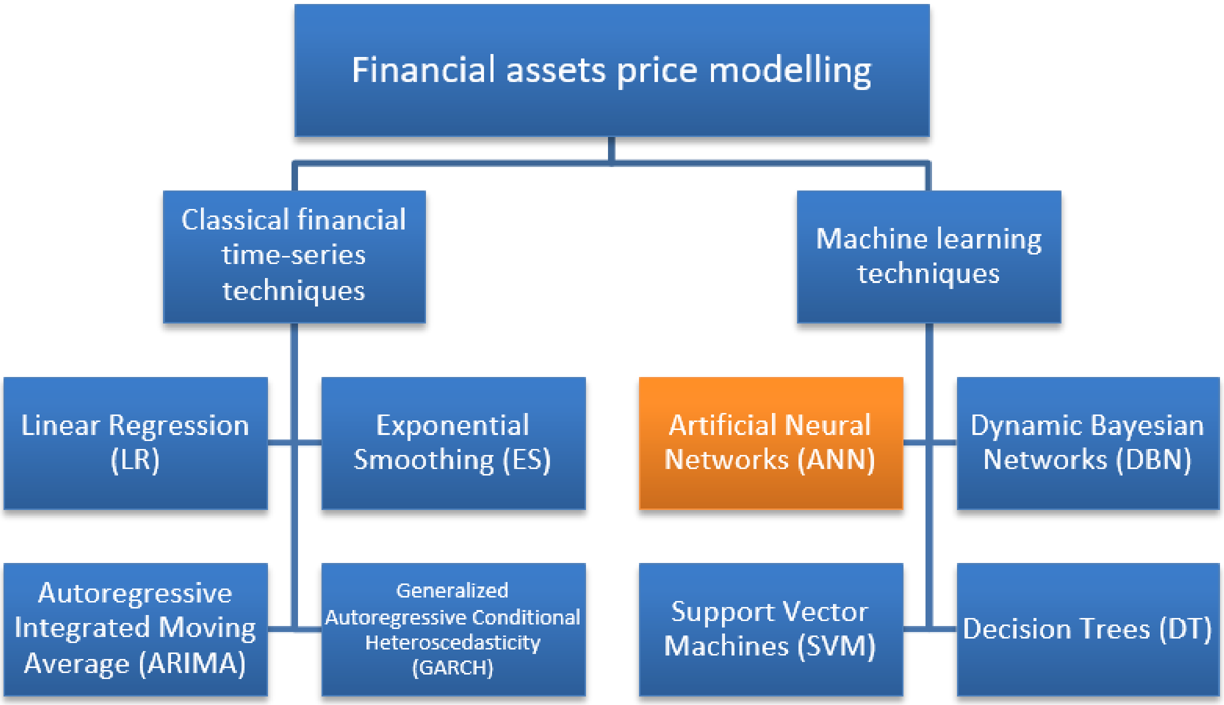

Financial assets price modelling
Classical financial time-series techniques
Machine learning techniques
Linear Regression (LR)
Exponential Smoothing (ES)
Artificial Neural Networks (ANN)
Dynamic Bayesian Networks (DBN)
Autoregressive Integrated Moving Average (ARIMA)
Generalized Autoregressive Conditional Heteroscedasticity (GARCH)
Support Vector Machines (SVM)
Decision Trees (DT)

Figure **1-1**: Financial assets price modelling

neuron structure.

The first devised ANN was the Feed-forward Neural Network (FNN), which has multiple neurons connected to each other, but there are no cycles or loops in the network. Therefore, the information always move forward from inputs to outputs nodes. Figure **1-4** shows a Multilayer perceptron (MLP), which is a FNN subtype. It is composed by an input, multiple hidden and an output layers. Each layer has a finite number of neurons that can be any positive integer; it does not imply that a layer must have the same amount as another layer. All neurons in a layer are fully connected to all neurons in the next layer [11]. The Universal Approximation Theorem says a MLP with a single hidden layer and enough neurons can approximate any function either linear or non-linear [11, 13].

Overall, ANNs have several limitations, due to it is difficult to find good weights and biases that enable the ANN correctly approximate the target data, because the training process is really an optimization; it gets stacked on local minima avoiding the ANN to converge to optimal solution. And also, over-training causes over-fitting, which means the ANN trend to memorize the data and then it fails to generalize the data [24, 33].

Traditionally, ANNs are trained with the back-propagation algorithm, which consists in initializing the weights matrices of the model with random values. Then the error between network output and desired output is evaluated. In order to identify the neurons that contributed to the error, the error is propagated backwards from the output layer to all neurons in the hidden layers that contributed to it. This process is repeated layer by layer, until all

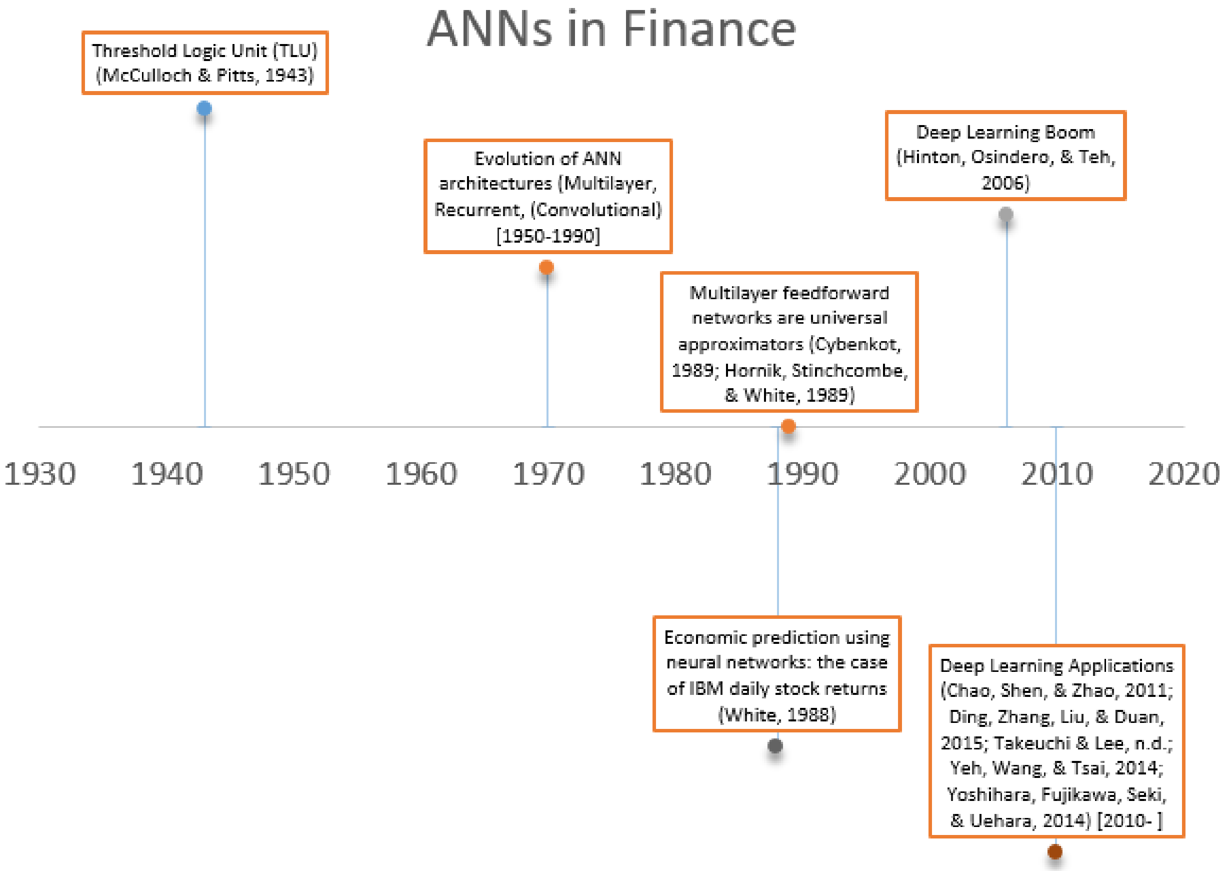

ANNs in Finance
Threshold Logic Unit (TLU) (McCulloch & Pitts, 1943)
Evolution of ANN architectures (Multilayer, Recurrent, (Convolutional) [1950-1990]
Multilayer feedforward networks are universal approximators (Cybenkot, 1989; Hornik, Stinchcombe, & White, 1989)
Deep Learning Boom (Hinton, Osindero, & Teh, 2006)
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
Economic prediction using neural networks: the case of IBM daily stock returns (White, 1988)
Deep Learning Applications (Chao, Shen, & Zhao, 2011; Ding, Zhang, Liu, & Duan, 2015; Takeuchi & Lee, n.d.; Yeh, Wang, & Tsai, 2014; Yoshihara, Fujikawa, Seki, & Uehara, 2014) [2010-]

Figure **1-2**: A brief time line of Artificial Neural Networks

neurons in the network have received an error signal describing their relative contribution to the total error. Later, the weights are updated in order to try to reduce the error. Then the error is calculated again and this process is repeated until a tolerable error or maximal number of iterations is reached [23].

A serious problem of back-propagation algorithm is that the error is diluted exponentially as it passes through hidden layers on their way to the network beginning. In a deep MLP (a MLP with many hidden layers), only the last layers are trained, while the first ones have barely changed.

Given the difficulty of training deep MLP, they have been useless. This challenge had remained unsolved. But in 2006, it was solved by [12], who successfully included paradigms of Deep Learning (DL) in Computer Science. Furthermore, it was possible due to the great advances in computing capabilities achieved by Graphics Processing Units (GPUs), which allow to accelerate training processes [3, 5, 17].

In recent years Deep Learning (DL) has emerged as a very robust machine learning technique, improving limitations of ANNs and training algorithms, such as back-propagation algorithm.

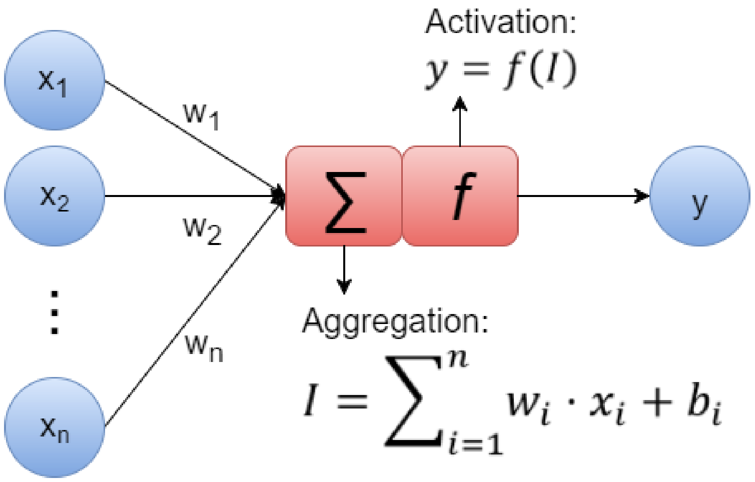


Figure **1-3**: Artificial Neuron Structure

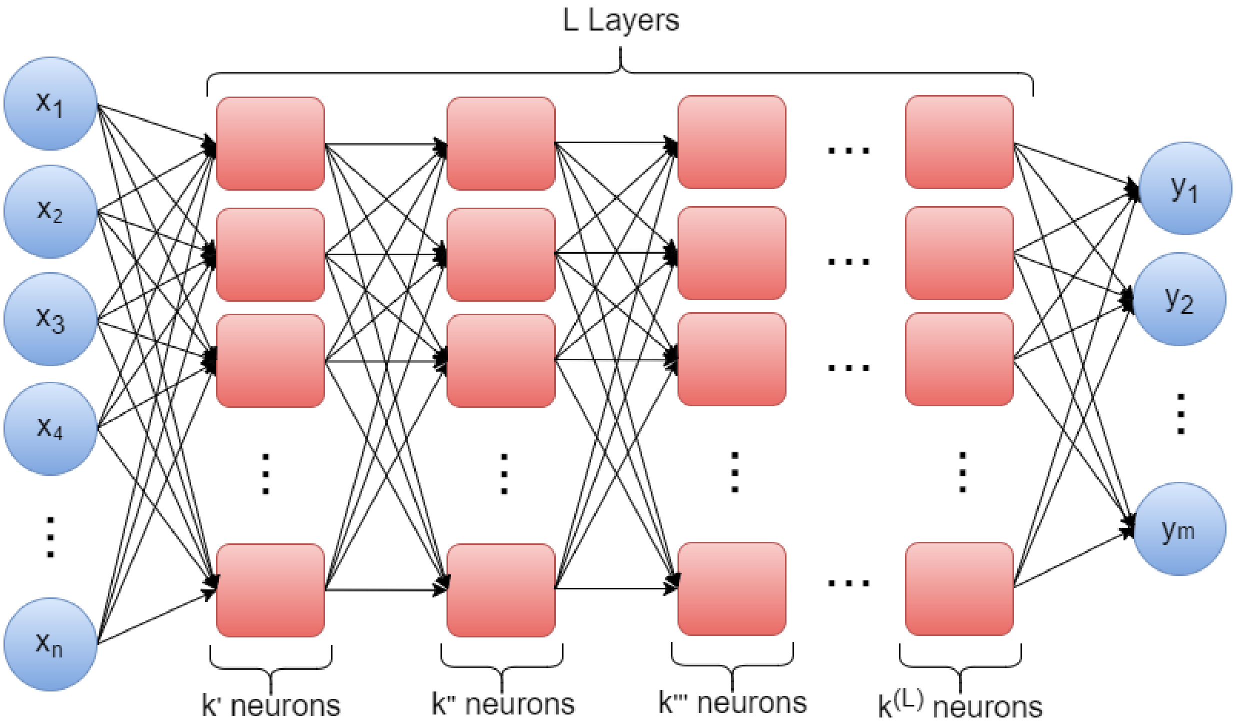


Figure **1-4**: Multilayer Perceptron

Deep Learning models are characterized by having wide inputs and deep architectures. Although in the literature reviewed, there are many discussions about what is or is not deep, [24] labels a MLP architecture as the first DL system.

A Deep Neural Network (DNN) is a deep MLP (with many layers), which uses DL training techniques. In a DNN, the data inputs are transformed from low-level to high-level features. The input layer is characterized by having many inputs. At each hidden layer the data is encoded in features of less dimensions by non-linear transformations; then, the next layers refine the learned patterns in high-level features of less dimensions, and so on until it is capable of learning complex patterns which are of interest in this work. This type of ANN can learn high-level abstractions from large quantities raw data through intermediate processing and refinement that occurs in each hidden layer [3] [24].

Models based on DL have begun to arouse the interest of the general public because they are able to learn useful representations from raw data and they have shown high performance in complex data, such as text, images and even video [1, 3]. However, applications of DL in computational finance are limited [4, 7, 26, 30, 31].

The thesis is organized as follows: Chapter 2 presents some important definitions of key concepts in this work. Chapter 3 describes the data preprocessing used. Chapter 4 presents the DNN modelling for forecasting the next one-minute average price of Apple, Inc. within financial crisis of 2008, when a high volatility behaviour was evidenced. Chapter 5 describes the proposed trading strategy algorithm. Chapter 6 presents the strategy performance with the described dataset. Chapter 7 presents the strategy validation with two other recent datasets. Moreover, Chapter 8 presents some conclusions and recommendations for future research.

Academic Products

- Arévalo, A., Niño, J., Hernández, G., & Sandoval, J. (2016). High-Frequency Trading Strategy Based on Deep Neural Networks. Intelligent Computing Methodologies, 424-436. `http://dx.doi.org/10.1007/978-3-319-42297-8_40`

- Arévalo, A. (2016). A High-Frequency Trading Strategy Using a Deep Multilayer Perceptron One-Minute Average Price Predictor. The 7th Annual Stevens Conference on High Frequency Finance and Analytics. Hoboken, NJ, USA.

Chapter 2

Definitions

Below some important definitions are presented:

Definition 2.1 *Log-return: It is a term commonly used in finance. Let p_t be the current trade or close price and p_{t-1} the previous trade or close price.*

$$R = \ln \frac{p_t}{p_{t-1}} \cdot 100\% = (\ln p_t - \ln p_{t-1}) \cdot 100\% \tag{2-1}$$

From a log-return R, the original price p_t can be reconstructed easily:

$$p_t = p_{t-1} \cdot e^{\frac{R}{100\%}} \tag{2-2}$$

Definition 2.2 *Pseudo-log-return: It is defined as a logarithmic difference (log of quotient) of between average prices on consecutive minutes. On the other hand, the typical log-return is a logarithmic difference of between closing prices on consecutive minutes. Let $\overline{p_t}$ be the current one-minute average price and $\overline{p_{t-1}}$ the previous one-minute average price.*

$$\hat{R} = \ln \frac{\overline{p_t}}{\overline{p_{t-1}}} \cdot 100\% = (\ln \overline{p_t} - \ln \overline{p_{t-1}}) \cdot 100\% \tag{2-3}$$

Furthermore, pseudo-returns can be reconstructed just as log-returns.

$$\overline{p_t} = \overline{p_{t-1}} \cdot e^{\frac{\hat{R}}{100\%}} \tag{2-4}$$

Definition 2.3 *One-minute Trend Indicator: It is a statistical indicator computed as the slope of the linear model (price $= at + b$) fitted of transaction prices in a particular minute, where t is the time in milliseconds inside the particular minute; A small slope, close to 0, means that in the next minute, the price is going to remain stable. A positive value means that the price is going to rise. A negative value means that the price is going to fall. Change is proportional to distance value compared to 0; if distance is too high, the price will rise or fall sharply.*

Chapter 3

Data Preprocessing

This high-frequency trading strategy is only applicable to high liquidity stocks, because it requires to open and close positions in a time interval equal or less than one minute. For experimental testing, this work uses Apple stock (ticker: APPL). Apple is one of the World's Blue Chips [8] therefore it complies with the high liquidity condition.

From the TAQ database of the NYSE [14], all trade prices for Apple ordinary stock (ticker: AAPL) were downloaded from the September 2nd to November 7th of the year 2008. Figures **3-1** shows price behaviour in the selected period.

The selected period covers the stock crash during the financial crisis of 2008. During this crash, the AAPL price suffered a dramatic fall from 172 to 98 dollars. This period was chosen intentionally to demonstrate the performance of the proposed strategy under high volatility conditions. During a financial crisis, market behaviour is strongly impacted by external factors to the system, such as news, rumours, anxiety of traders, among others. If a DNN can identify and learn patterns under these difficult conditions, it can achieve equal or even better performance with other market conditions without a financial crisis.

The tick-by-tick dataset is composed by 14,839,395 observations. It has a maximum price on 173.5 dollars and a minimum one on 85 dollars. Figure **3-2** shows the one-minute

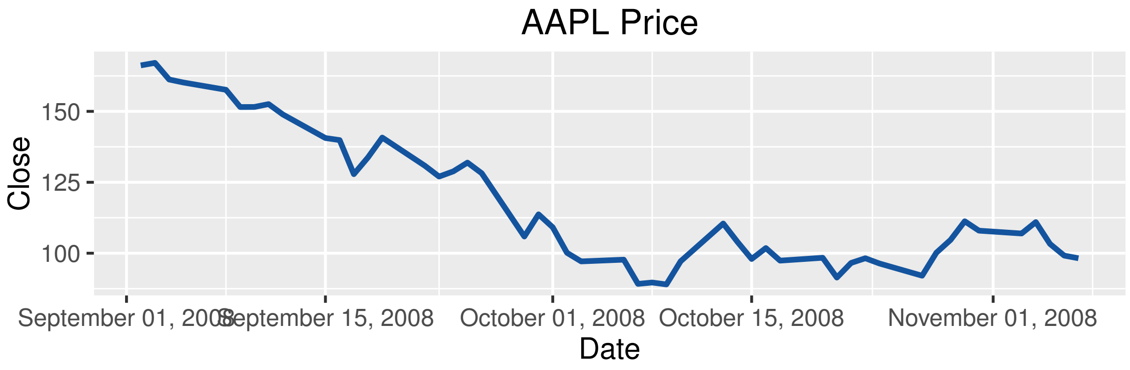


Figure **3-1**: Apple Stock Price.

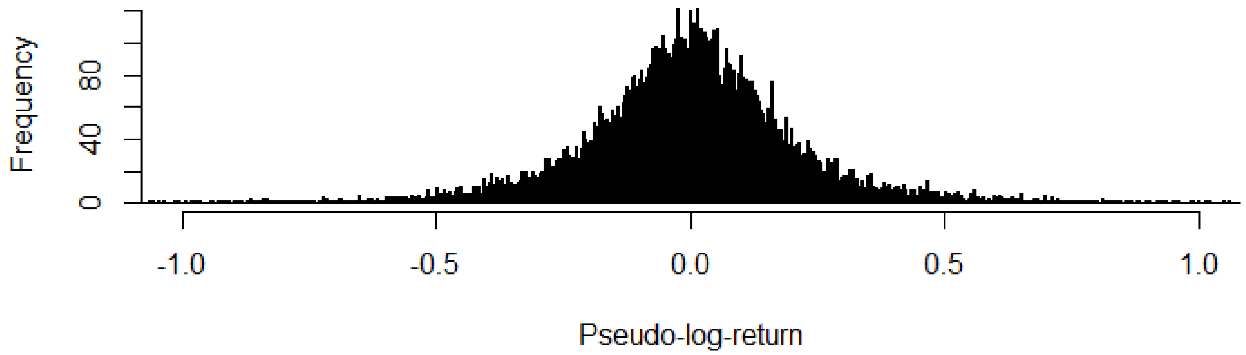


Figure **3-2**: One-minute Pseudo-log-returns Histogram.

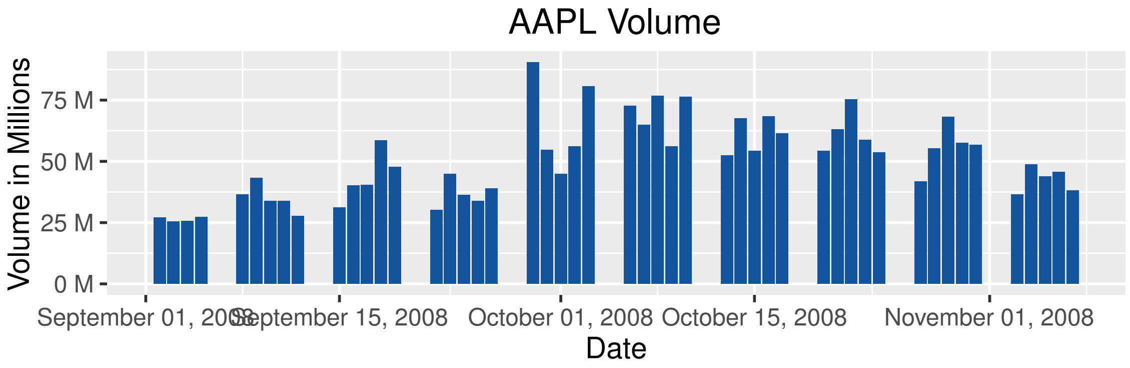


Figure **3-3**: Daily Trading Volume.

pseudo-log-returns histogram. It is some symmetric with mean of -0.002961% and standard deviation of 0.282765. The maximum pseudo-log-return is 7.952000%, the minimum one is -7.659000%, the first quartile is -0.112200% and the third one is 0.108500. Given these properties, the pseudo-log-returns distribution can be approximated to a normal distribution.

Reviewed literature suggests that any stock log-returns follows approximately a normal distribution with mean zero [10, 20, 27]. As expected, the pseudo-log-returns distribution behaves like the log-returns distribution. For this reason, the best variables that describe the market behaviour within a minute are the mean price and standard deviation of prices.

Figure **3-3** shows the daily trading volume. Besides, figure **3-4** shows the volume distribution. The 98.49519% of transactions are carried out by the less than $1,000$ shares with a mean equal to 165.5809 ones. The minimum value is 100 and the maximum one is $5,155,222$.

First, the data consistency was verified. All dates were in working days (not holidays and not weekends). All times were during normal hours of trading operation (between 9:30:00 am and 3:59:59 pm EST). All prices and volumes were positive. Therefore, it was not necessary to delete records.

All data were summarized with a one-minute detailed level. Three time series were constructed from trading prices: **Average Price**, **Standard Deviation of Prices** and

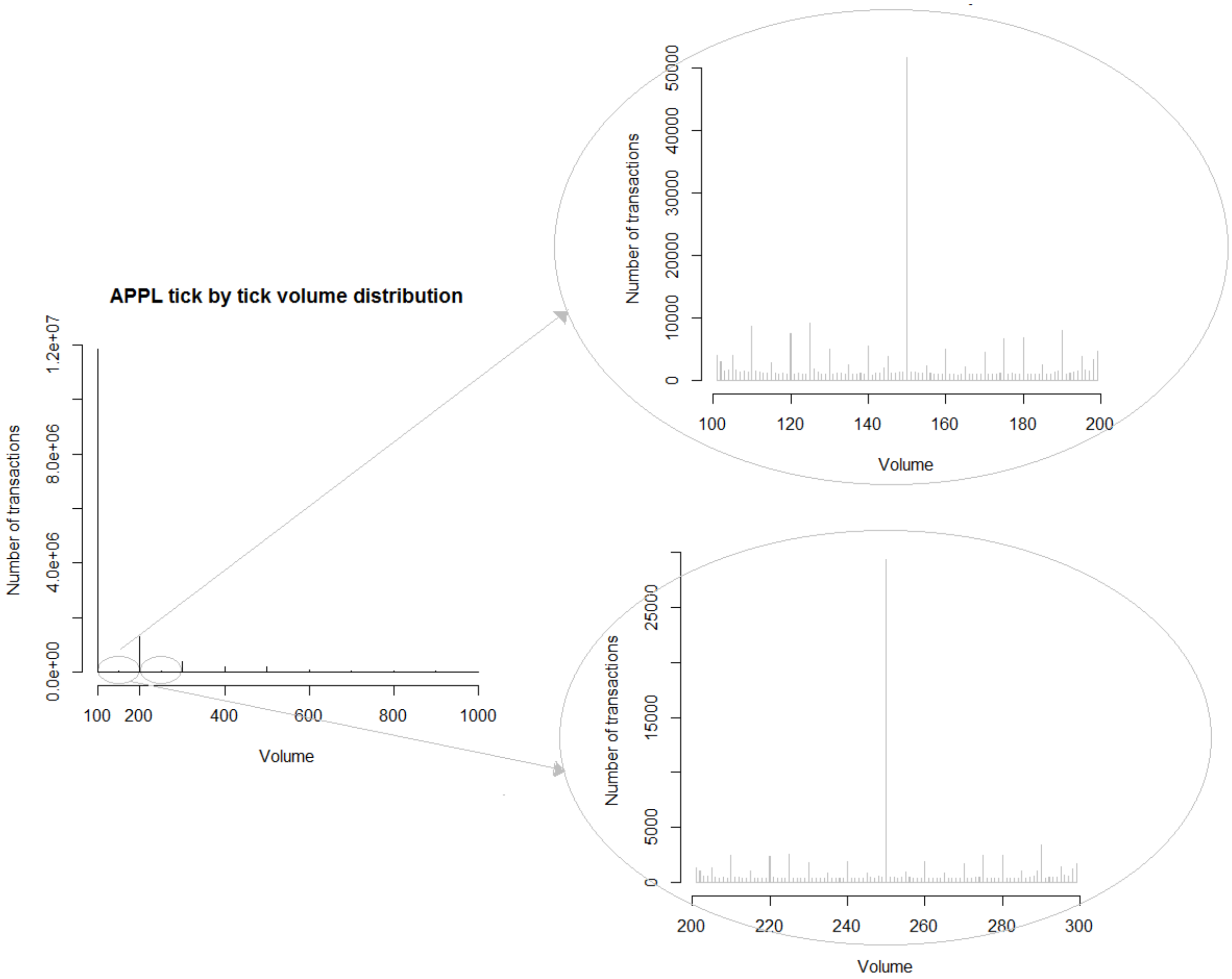


Figure **3-4**: Distribution of Volume.

Trend Indicator. Each series has 19110 records (49 trading days × 390 minutes per day).

Figure **3-5** shows the one-minute average price histogram. This distribution has a mean of 118.10 dollars and a standard deviation of 23.87065. The first quartile is 97.90 dollars and the second one is 108.40 dollars and the third one is 136.60 dollars.

Figure **3-6** shows the histogram of standard deviation of prices. This distribution has a mean of 0.10420 and a standard deviation of 0.07501747, a minimum value of 0.00865, a maximum one of 1.66100. The first quartile is 0.05826, the second one is 0.08495 and the third one is 0.12630.

Figure **3-7** shows the trend indicator histogram. This distribution has a mean of $-5.638 \cdot 10^{-6}$ and a standard deviation of $4.593 \cdot 10^{-4}$, a minimum value of $-2.827 \cdot 10^{-3}$, a maximum one of $3.032 \cdot 10^{-3}$. The first quartile is $-2.749 \cdot 10^{-4}$, the second one is $-4.399 \cdot 10^{-6}$ and the third one is $2.644 \cdot 10^{-4}$. This distribution has very small values, therefore it is very important to apply any normalization technique before using the DNN.

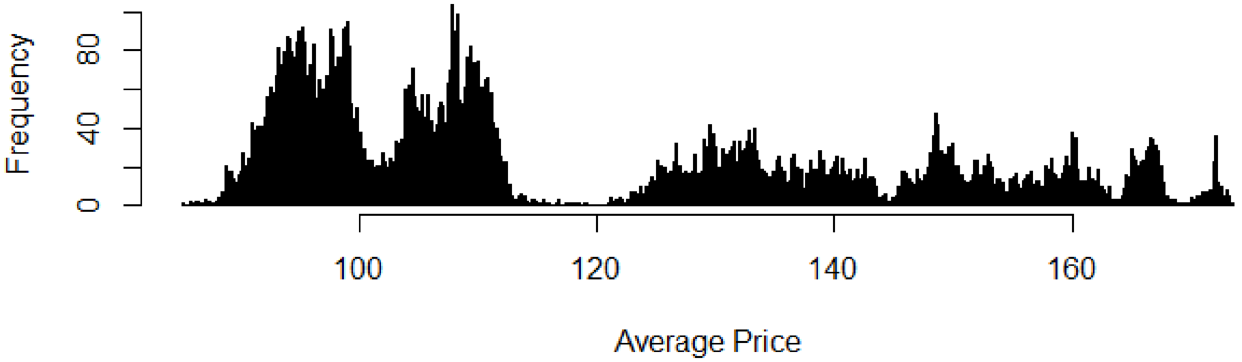


Figure **3-5**: Average Price Histogram.

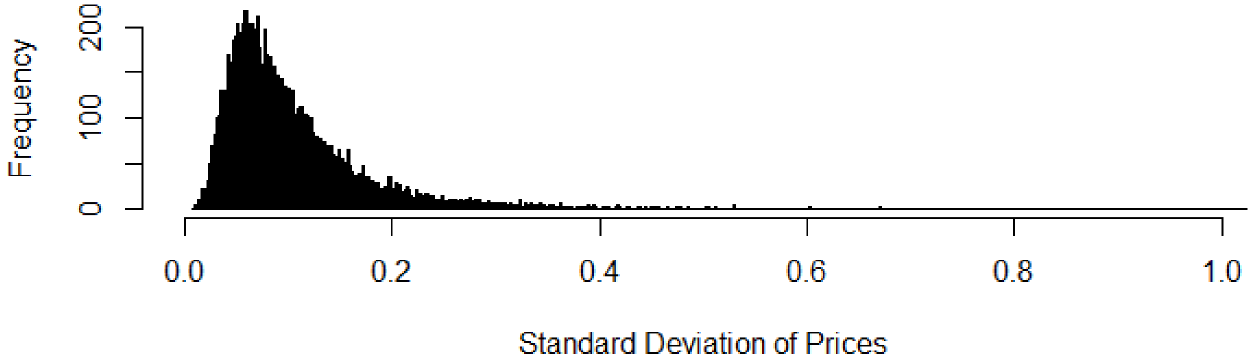


Figure **3-6**: Standard Deviation Histogram.

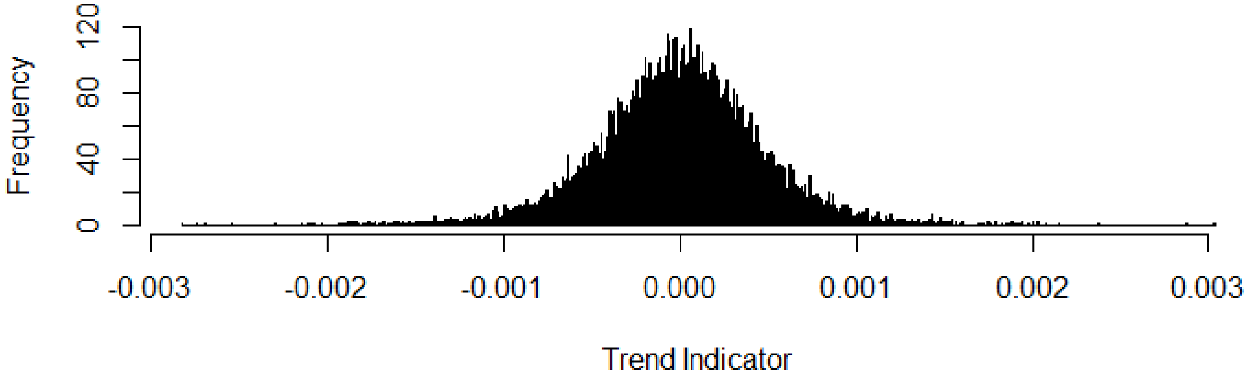


Figure **3-7**: Trend Indicator Histogram.

Chapter 4

Deep Neural Network Modelling

Figure **4-1** shows the DNN overview. Below the DNN is explained in detail.

4.1 Features Selection

In total four inputs-groups were chosen: Current Time, last n pseudo-log-returns, last n standard deviations of prices and last n trend indicators, where n is the window size. The current time group is composed of two inputs: **Current hour** and **Current minute**. The others groups are composed of n inputs for each one. In total the number of DNN inputs I is $3n + 2$. The following paragraphs describe each input group:

4.1.1 Current time:

The literature reviewed did not include time as an input. However, the hour and minute as integer values were chosen as two additional inputs, due to financial markets are affected by regimes that occurs repeatedly in certain minutes or hours during the trading day. This behaviour may be explained by the fact that both human and automatized (machines) traders have strategies that are running in synchronized periods.

To illustrate this affirmation, figure **4-2** shows the price variations that occurred at the first, third and sixth day. Approximately, in the minute 170, the stock was traded at the same opening price of corresponding day. Approximately, in the minute 250, the stock price fell 3 dollars relative to the opening price in these days. As these patterns, many more are repeated at certain time of day. In order to identify and to differentiate better these patterns, the current hour and current minute of day were added as additional inputs of DNN. These variables have 7 and 60 possible values ranging from 0 to 6 and from 0 to 59 respectively.

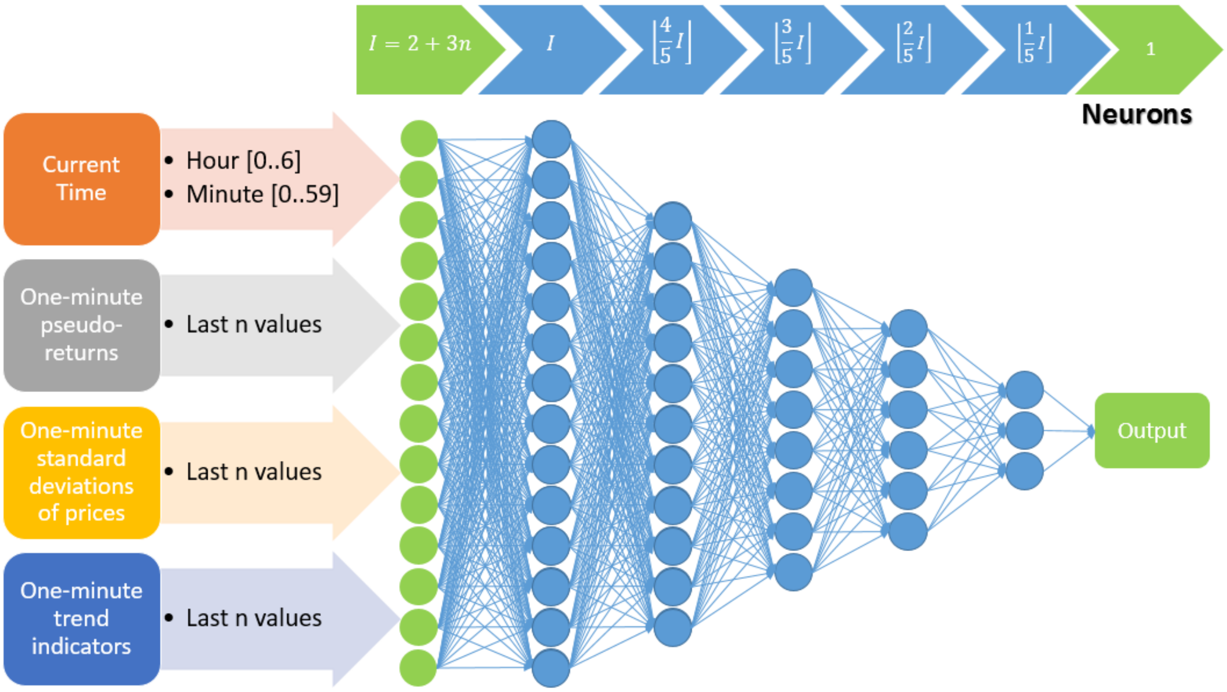

$I = 2 + 3n$
I
$\left\lfloor \frac{4}{5} I \right\rfloor$
$\left\lfloor \frac{3}{5} I \right\rfloor$
$\left\lfloor \frac{2}{5} I \right\rfloor$
$\left\lfloor \frac{1}{5} I \right\rfloor$
1
Neurons
Current Time
• Hour [0..6]
• Minute [0..59]
One-minute pseudo-returns
• Last n values
One-minute standard deviations of prices
• Last n values
One-minute trend indicators
• Last n values
Output

Figure **4-1**: DNN with five hidden layers

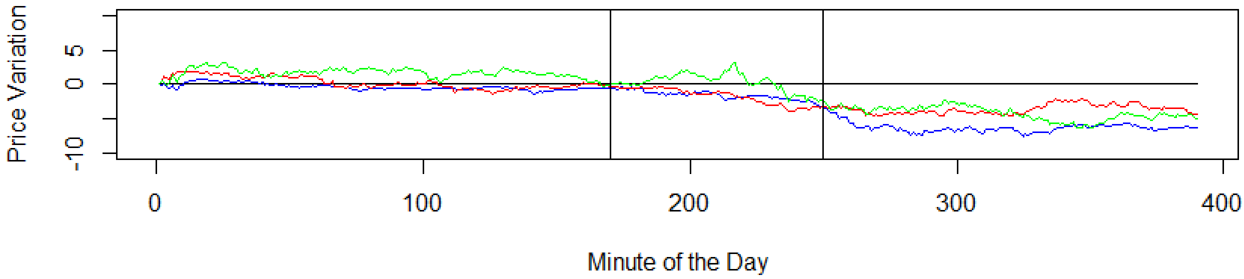


Figure **4-2**: The price variations that occurred at the first (blue line), third (red line) and sixth (green line) day

4.1.2 Last n pseudo-log-returns:

It is common to see works of neural networks used to forecast time series whose inputs are composed principally by the last untransformed observations. This is fine for several types of time series, but it is not advisable in financial time series forecasting. In any dataset and particularly in the one used in this work, if the nominal prices are used, it will be useless because a neural network will train with special conditions (prices fluctuates between 120 and 170 dollars) and then it will be tested against different market conditions (prices fluctuates between 90 and 120 dollars).

In other words, the neural network learns to identify many static patterns that will be not appearing at all. For example, a pattern like when the price is over 150 dollars, raises to 150.25 dollars and falls to 149.75 dollars, then it will change to 150.50 dollars, could be found, but this pattern never will occur because in the closest future the prices fluctuates between 90 and 120 dollars. However, if prices are transformed into differences or logarithmic returns, not only the data variance is stabilized, but also the time series acquire temporal independence. For example at the beginning of the selected period, a pattern, like when the price rises 25 cents and it falls 50 cents, then it will raise 75 cents, could be found and this pattern is more likely to occur in the future. Therefore, the last n one-minute pseudo-log-returns are inputs of DNN.

4.1.3 Last n standard deviations of prices:

The last n one-minute standard deviations of prices are DNN inputs.

4.1.4 Last n trend indicators:

The last n one-minute trend indicators are DNN inputs.

4.2 Output selection of the Deep Neural Network

The DNN forecasts the next one-minute pseudo-log-return. As it is shown on figure **4-3**, the average price (black line) is the variable that best describes market behaviour. The highest or lowest prices (blue lines) usually are found within a confidence range of average price, therefore the next highest and lowest prices can be estimated from a predicted average price. The closing price (red line) can be any value close to the average price; it sometimes coincides with the highest or lowest price. Unlike the average price, the highest, lowest and closing ones are exposed largely to noise or external market dynamics, for example, some traders listen a false rumour about bad news that will cause a sudden fall in the price, in order to reduce losses. As a result, they decide to sell at a lower price than the one traded before. This operation could be done at a certain second and it could affect numerically the highest, lowest or closing prices on the minute.

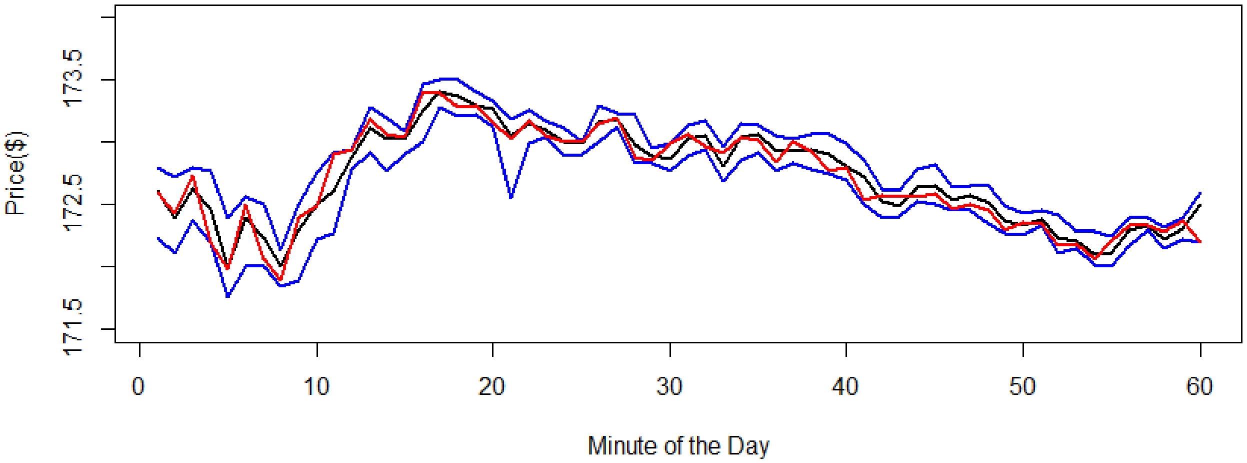


Figure **4-3**: First 60 Apple Stock Prices. Blue: High and Low. Red: Close. Black: Average.

Since the objective of this work is to learn the dynamics of the market to take ad-vantage of it eventually, the average price forecasts could be more profitable than the closing price forecast. With a good average price forecast, it is known that the stock is going to trade to that predicted value at any moment within the next minute. A real automated trading strategy should wait until the right time (for example, stock price reaches to price forecast) to open or to close their positions.

4.3 Deep Neural Network Architecture

The architecture was selected arbitrarily. It has one input layer, L hidden layers and one output layer. The number of neurons in each layer depends on the number of inputs I and the number of hidden layers L. In each hidden layer, the number of neurons decreases with a constant factor $\frac{1}{L}$. For example five with hidden layers: Each layer will have I, $\lceil\frac{4}{5}I\rceil$, $\lceil\frac{3}{5}I\rceil$, $\lceil\frac{2}{5}I\rceil$, $\lceil\frac{1}{5}I\rceil$ neurons respectively. For example with three hidden layers: Each layer will have I, $\lceil\frac{2}{3}I\rceil$, $\lceil\frac{1}{3}I\rceil$ respectively. Whereas the output layer always has one neuron. All neurons in the hidden layers use a $Tanh$ activation function, Whereas the output neuron which uses a $Linear$ activation function.

4.4 Deep Neural Network Training

The final dataset is made up of $19109 - n$ records. Each record contains $3n + 3$ numerical values ($3n + 2$ inputs and 1 output). The final dataset was divided into two parts: training data (the first 85% samples) and testing data (the remaining 15% samples). It should be noted that to construct each record, only information from the past is required. Therefore, there is not look-ahead bias and this DNN could be used for a real trading strategy.

For this work, H_2O, an open-source software for big-data analysis[21] was used. It implements algorithms at scale, such as deep learning [2], as well as featuring automatic versions of advanced optimization for DNN training. Additionally, it implements a adaptive learning rate algorithm, called ADADELTA [2], which is described in [32]. It was chosen in order to improve the learning process, due:

- It is a per-dimension adaptive learning rate method for gradient descent.
- It is not necessary to manually search parameters for gradient descent.
- It is robust to large gradients and noise.

4.5 Deep Neural Network Assessment

In order to assess the DNN performance, four statistics were chosen. Let E be the expected series and F be the series forecast:

1. **Mean Absolute Error:**

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|E_t - F_t| \tag{4-1}$$

2. **Mean Squared Error:**

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(E_t - F_t)^2 \tag{4-2}$$

3. **Mean Absolute Percentage Error:**

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\frac{|E_t - F_t|}{|E_t|} \tag{4-3}$$

4. **Directional Accuracy:** Percent of predicted directions that matches with the ideal differences time series. This measure is unaffected by outliers or variables scales.

$$DA = \frac{100}{n}\sum_{t=1}^{n}E_t \cdot F_t > 0 \tag{4-4}$$

Chapter 5

Proposed Strategy

The DNN predictions are used by the following high-frequency trading strategy: For each trading minute, it always buys(sells) a stock when the next predicted average price is above(below) the last closing price. When the price yields the predicted average price, it sells(buys) the stock in order to ensure the profit. If the price never yields the expected price, it sells(buys) the stock with the closing price of the minute, in order to close the position and potentially stop losing positions. Figure **5-1** shows the strategy flowchart. Below the algorithm is formally presented in pseudo-code:

```
for each trading minute do
  {At the beginning of the minute, to forecast the next one-minute average price}
  $I \leftarrow$ Current features vector.
  $predicted.pseudo.return \leftarrow DNN.forecast(I)$
  $predicted.average.price \leftarrow last.average.price \cdot e^{predicted.pseudo.return/100}$

  {To open a position}
  if $predicted.average.price > previous.closing.price$ then
    {To buy a stock at the current price}
    $current.operation \leftarrow BUY$
  else if $predicted.average.price < previous.closing.price$ then
    {To sell a stock at the current price}
    $current.operation \leftarrow SELL$
  else
    continue {To do nothing until next minute}
  end if

  {To close the position}
  while The current minute has not ended do
    $current.price \leftarrow$ Current stock price
    if $current.operation == BUY$ and $current.price \geq predicted.average.price$ then
```

```
        {To sell the stock at the current price and to earn the difference}
        $current.operation \leftarrow null$
        continue {To do nothing until next minute}
      end if

      if $current.operation == SELL$ and $current.price \leq predicted.average.price$ then
        {To buy the stock at the current price and to earn the difference}
        $current.operation \leftarrow null$
        continue {To do nothing until next minute}
      end if
    end while

    {To stop the losses, if the operation was not closed}
    if $current.operation! = null$ then
      if $current.operation == BUY$ then
        {To sell the stock at the current price}
      else if $current.operation == SELL$ then
        {To buy the stock at the current price}
      end if
      $current.operation \leftarrow null$
    end if
  end for
```

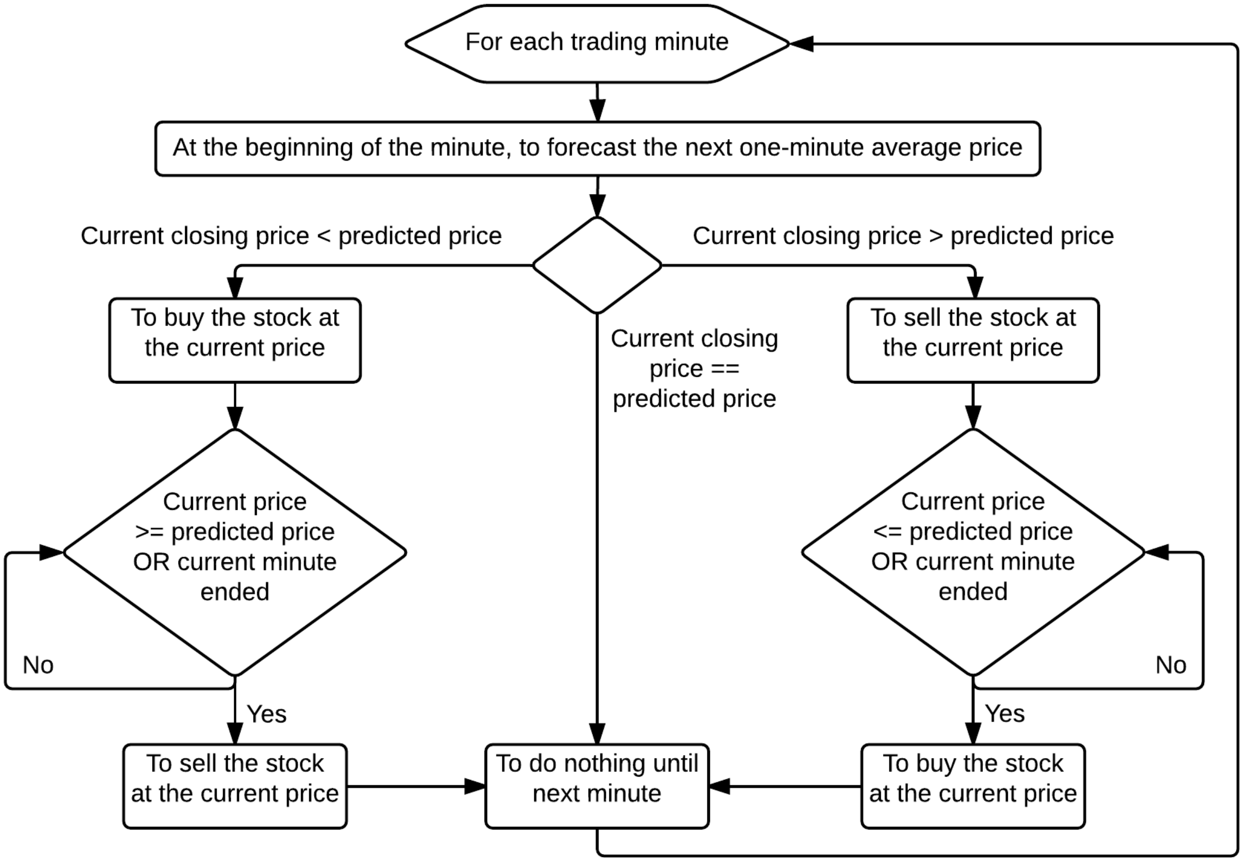

For each trading minute
At the beginning of the minute, to forecast the next one-minute average price
Current closing price < predicted price
Current closing price > predicted price
To buy the stock at the current price
Current closing price == predicted price
To sell the stock at the current price
Current price >= predicted price OR current minute ended
Current price <= predicted price OR current minute ended
No
Yes
Yes
No
To sell the stock at the current price
To do nothing until next minute
To buy the stock at the current price

Figure **5-1**: Strategy Flowchart

Chapter 6

Experiment

6.1 Short-term forecasting

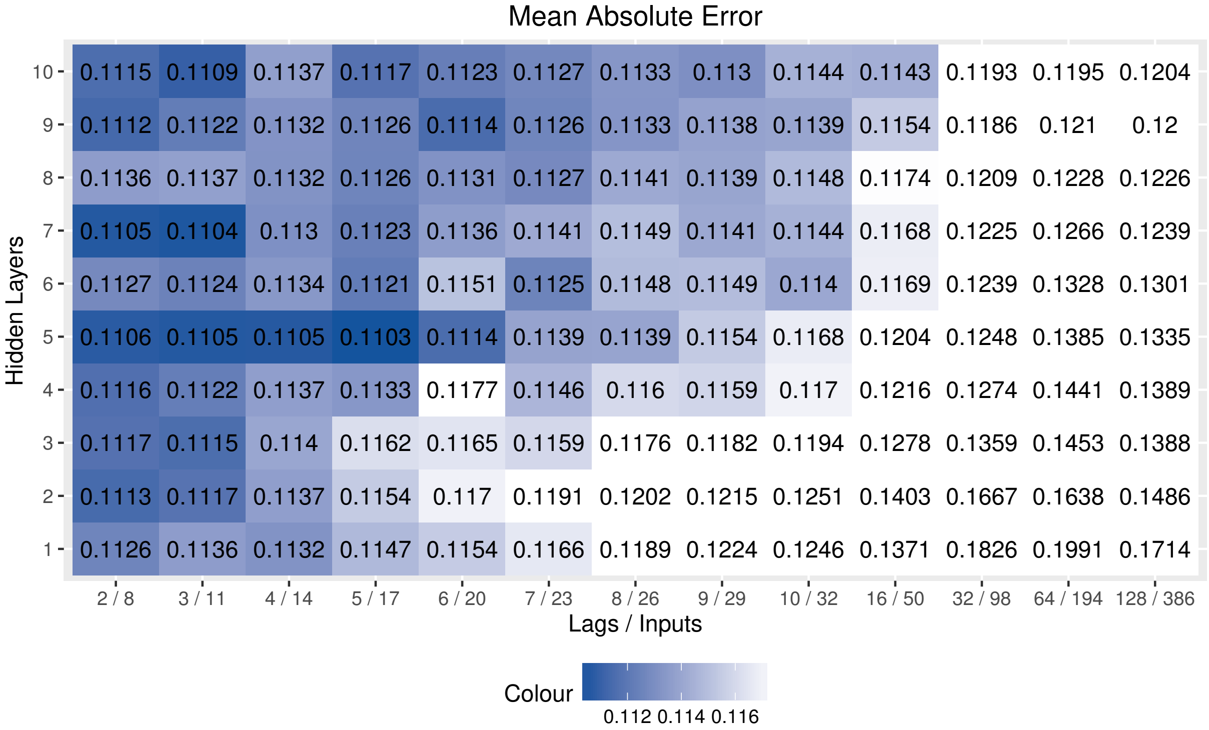
The DNNs were trained only with the training data during 50 epochs each one. The chosen ADADELTA parameters were $\rho = 0.9999$ and $\epsilon = 10^{-10}$. On the other hand, the DNNs were tested only with the testing data.

Figures **6-1**, **6-2**, **6-3** and **6-4** illustrate the average DNN performance using different sliding windows sizes (n lags) and number of hidden layers L. The number of neurons in each layer depends on the number of inputs $I = 3n + 2$. In each hidden layer, the number of neurons decreases with a constant factor $\frac{1}{L}$. For example five with hidden layers: Each layer will have I, $\lceil \frac{4}{5} I \rceil$, $\lceil \frac{3}{5} I \rceil$, $\lceil \frac{2}{5} I \rceil$, $\lceil \frac{1}{5} I \rceil$ neurons respectively. For example with three hidden layers: Each layer will have I, $\lceil \frac{2}{3} I \rceil$, $\lceil \frac{1}{3} I \rceil$ respectively. All DNNs have a single output layer with only one neuron. Furthermore, all neurons in the hidden layers use a $Tanh$ activation function, Whereas the output neuron which uses a $Linear$ activation function.

For each combination, ten different networks were trained. In each table, the best configurations were highlighted with a darker colour. As shown in the figures **6-1**, **6-2**, **6-3** and **6-4**, the best combinations are located leftward of x-axis and upward of y-axis, that is, the best yields are obtained by architectures that consider small sliding windows sizes and that are multi-layered.

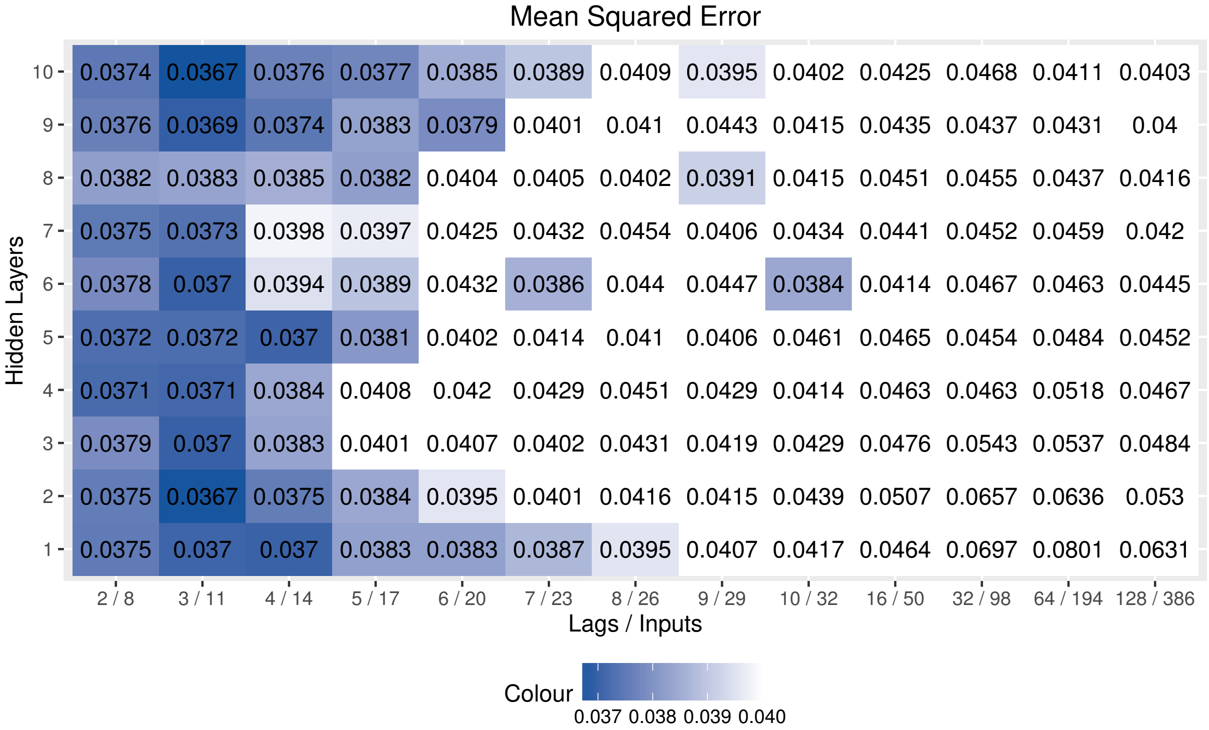
The best (lowest) MAE were achieved by architectures having between four and seven hidden layers and using a small sliding window size of less than six minutes. The best (lowest) MSE were achieved by almost all architectures using small sliding windows of four minutes or less, regardless of the number of hidden layers. The best (lowest) MAPE were achieved by architectures having four or more hidden layers. The best (highest) DA were achieved by architectures having four or more hidden layers. Also, when the number of layers increased, the DNNs achieved good MAPE and DA with larger sliding window sizes.

Comparing the figures, the best results were obtained by architectures having four to seven hidden layers, and sliding windows sizes of five or less minutes. Depending on training results, DNN performance may be better, but all networks converge with very similar and



Hidden Layers	2 / 8	3 / 11	4 / 14	5 / 17	6 / 20	7 / 23	8 / 26	9 / 29	10 / 32	16 / 50	32 / 98	64 / 194	128 / 386
10	0.1115	0.1109	0.1137	0.1117	0.1123	0.1127	0.1133	0.113	0.1144	0.1143	0.1193	0.1195	0.1204
9	0.1112	0.1122	0.1132	0.1126	0.1114	0.1126	0.1133	0.1138	0.1139	0.1154	0.1186	0.121	0.12
8	0.1136	0.1137	0.1132	0.1126	0.1131	0.1127	0.1141	0.1139	0.1148	0.1174	0.1209	0.1228	0.1226
7	0.1105	0.1104	0.113	0.1123	0.1136	0.1141	0.1149	0.1141	0.1144	0.1168	0.1225	0.1266	0.1239
6	0.1127	0.1124	0.1134	0.1121	0.1151	0.1125	0.1148	0.1149	0.114	0.1169	0.1239	0.1328	0.1301
5	0.1106	0.1105	0.1105	0.1103	0.1114	0.1139	0.1139	0.1154	0.1168	0.1204	0.1248	0.1385	0.1335
4	0.1116	0.1122	0.1137	0.1133	0.1177	0.1146	0.116	0.1159	0.117	0.1216	0.1274	0.1441	0.1389
3	0.1117	0.1115	0.114	0.1162	0.1165	0.1159	0.1176	0.1182	0.1194	0.1278	0.1359	0.1453	0.1388
2	0.1113	0.1117	0.1137	0.1154	0.117	0.1191	0.1202	0.1215	0.1251	0.1403	0.1667	0.1638	0.1486
1	0.1126	0.1136	0.1132	0.1147	0.1154	0.1166	0.1189	0.1224	0.1246	0.1371	0.1826	0.1991	0.1714

Figure **6-1**: DNN performance: MAE



Hidden Layers	2 / 8	3 / 11	4 / 14	5 / 17	6 / 20	7 / 23	8 / 26	9 / 29	10 / 32	16 / 50	32 / 98	64 / 194	128 / 386
10	0.0374	0.0367	0.0376	0.0377	0.0385	0.0389	0.0409	0.0395	0.0402	0.0425	0.0468	0.0411	0.0403
9	0.0376	0.0369	0.0374	0.0383	0.0379	0.0401	0.041	0.0443	0.0415	0.0435	0.0437	0.0431	0.04
8	0.0382	0.0383	0.0385	0.0382	0.0404	0.0405	0.0402	0.0391	0.0415	0.0451	0.0455	0.0437	0.0416
7	0.0375	0.0373	0.0398	0.0397	0.0425	0.0432	0.0454	0.0406	0.0434	0.0441	0.0452	0.0459	0.042
6	0.0378	0.037	0.0394	0.0389	0.0432	0.0386	0.044	0.0447	0.0384	0.0414	0.0467	0.0463	0.0445
5	0.0372	0.0372	0.037	0.0381	0.0402	0.0414	0.041	0.0406	0.0461	0.0465	0.0454	0.0484	0.0452
4	0.0371	0.0371	0.0384	0.0408	0.042	0.0429	0.0451	0.0429	0.0414	0.0463	0.0463	0.0518	0.0467
3	0.0379	0.037	0.0383	0.0401	0.0407	0.0402	0.0431	0.0419	0.0429	0.0476	0.0543	0.0537	0.0484
2	0.0375	0.0367	0.0375	0.0384	0.0395	0.0401	0.0416	0.0415	0.0439	0.0507	0.0657	0.0636	0.053
1	0.0375	0.037	0.037	0.0383	0.0383	0.0387	0.0395	0.0407	0.0417	0.0464	0.0697	0.0801	0.0631

Figure **6-2**: DNN performance: MSE

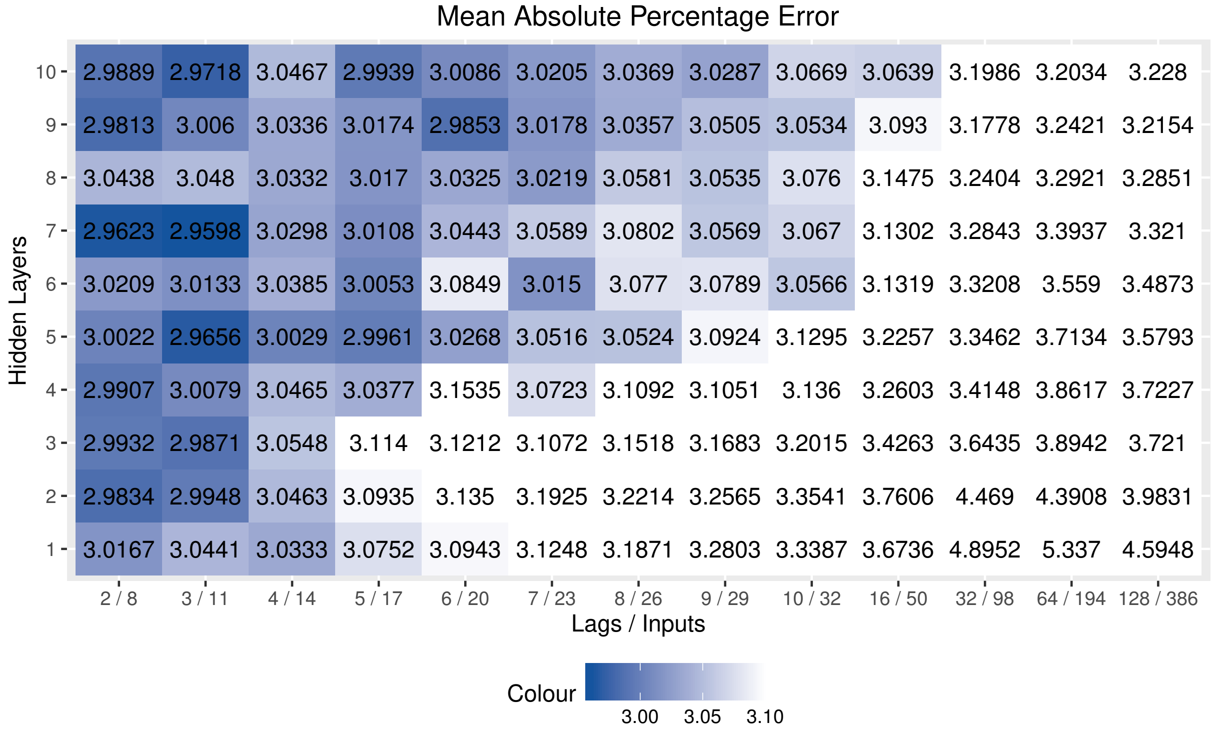


Figure **6-3**: DNN performance: MAPE

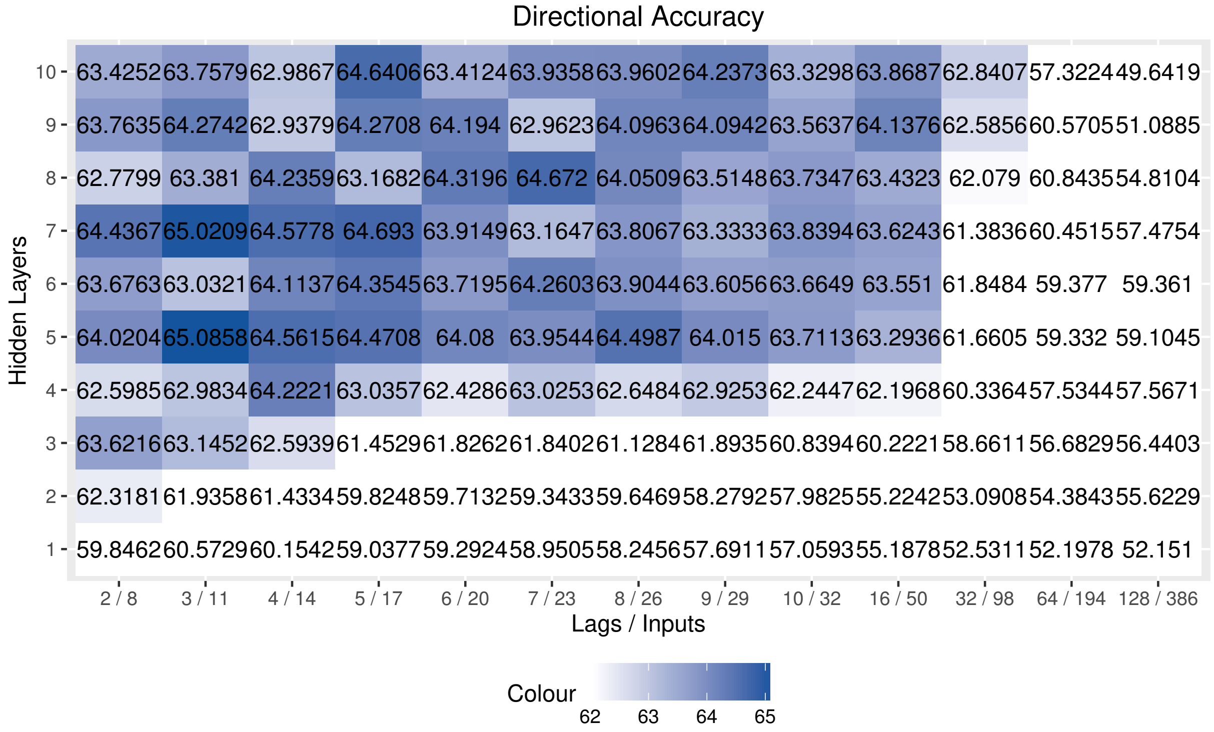


Figure **6-4**: DNN performance: DA

homogeneous results.

On average, the DNNs achieved approximately between 0.11 and 0.20 of MAE, between 0.03 and 0.08 of MSE, between 2.95% and 5.33% of MAPE and between 49.64% and 65.08% of DA. The DNNs are able to predict these sudden rises or falls in price. This information may be useful for any trading strategy.

6.2 Strategy simulation

For the effectiveness of the proposed strategy, it is required a DNN that has a good performance identifying the price direction. For this reason, the architecture with the best directional accuracy (DA) was selected: A DNN that uses a sliding window size of 3 minutes, and that have five hidden layers, 14 inputs and 1 output; Each hidden layer has 14, 12, 9, 6 and 3 neurons respectively.

Figure **6-5** shows the strategy performance during a trading simulation over the testing data. The simulation did not consider transaction costs and was performed with the selected DNN. Buying and selling the equivalent of a dollar in shares, the strategy accumulated approximately $0.55 in 8 trading days. It made 2866 trades, of which 2302 (%80.32) ones were closed with profits, 551 (%19.23) ones with losses, and 13 (%0.45) ones with no gain or no loss.

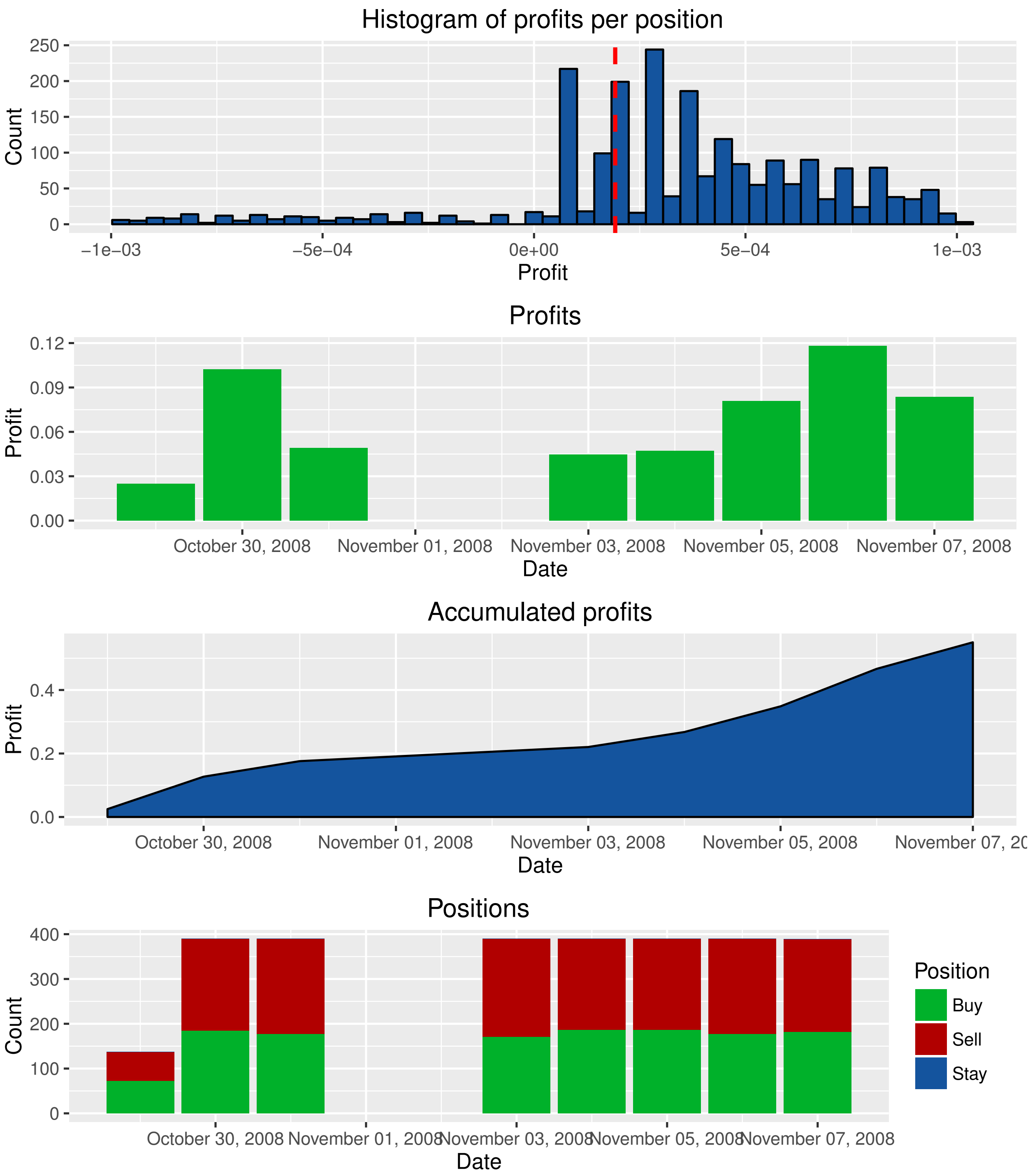


Figure **6-5**: Trading Strategy Performance

Chapter 7

Strategy Validation

The dataset used in the experiment can be a little dramatic, since it belongs to a period in financial crisis. For this reason it was decided to select two recent datasets and apply the same strategy using the same network architecture and same data split.

It is important to remember that this high-frequency trading strategy is only applicable to high liquidity stocks, because it requires to open and close positions in a time interval equal or less than one minute. Therefore, from the TAQ database of the NYSE [14], all trade prices for Apple ordinary stock (ticker: AAPL - dataset: A) and Google ordinary stock (ticker: GOOG - dataset: B) were downloaded from the August 10th, 2015 to August 8th, 2016. Figures **7-1** and **7-2** show price and volume behaviour of dataset A. And figures **7-3** and **7-4** show price and volume behaviour of dataset B.

A DNN with the selected architecture (5 hidden layers and a sliding window size of 3 minutes) was trained with the dataset A. The data set was split in the same way: The first 85% as training data and the remaining 15% as testing set. Figure **7-5** shows the strategy performance with the dataset A.

The simulation was performed with the same conditions of the previous experiment (only with the testing data, and the strategy buys or sells the equivalent of a dollar per trade).

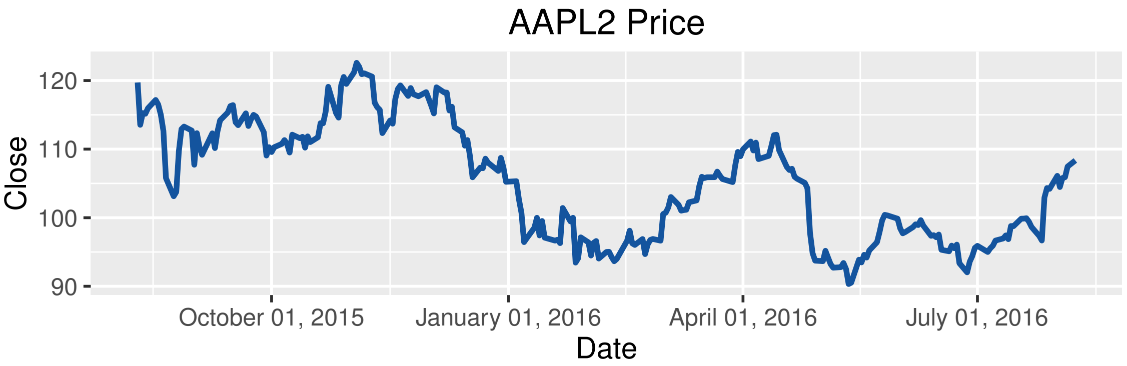


Figure **7-1**: Dataset A: AAPL Price

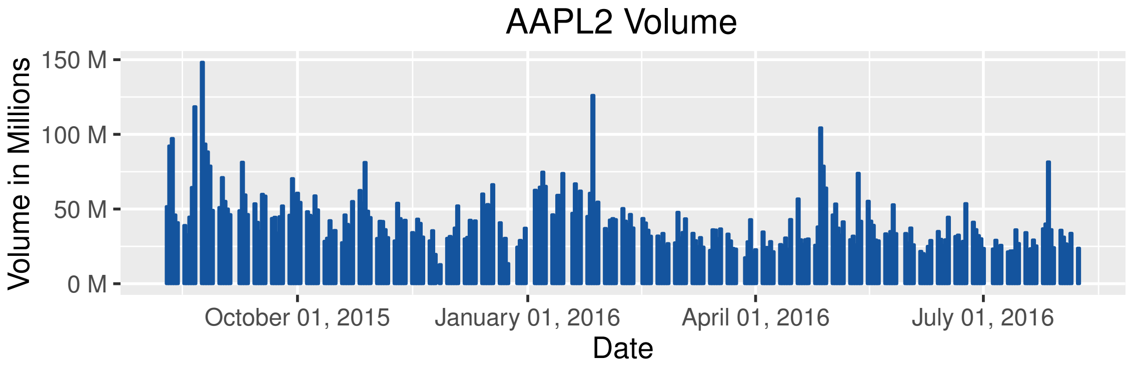


Figure **7-2**: Dataset A: AAPL Volume

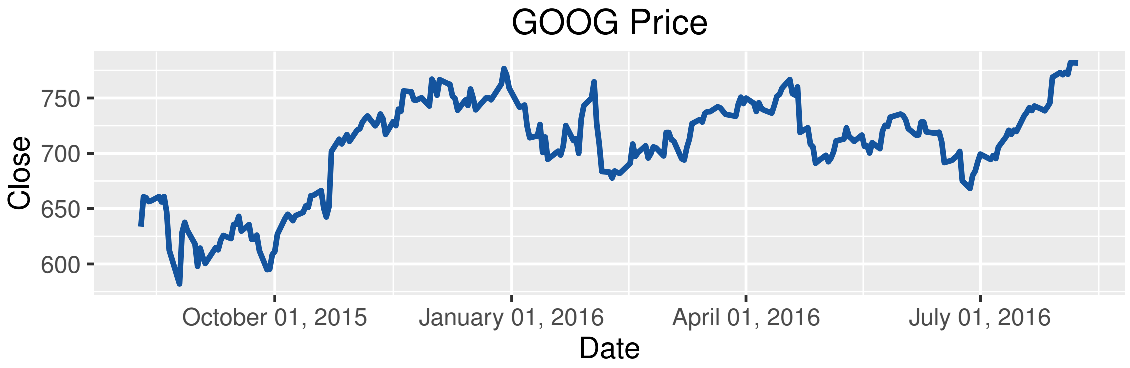


Figure **7-3**: Dataset B: GOOG Price

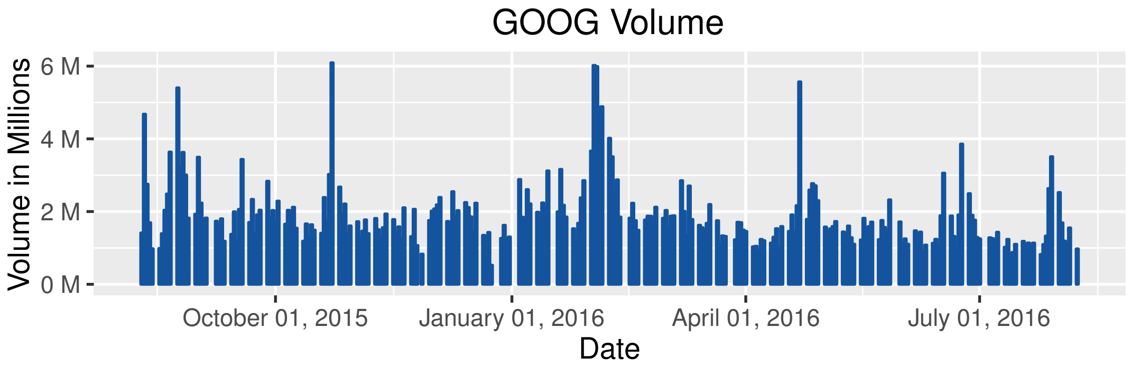


Figure **7-4**: Dataset B: GOOG Volume

The strategy accumulated approximately \$1.15 in 38 trading days. It made 14713 trades, of which 11941 (%81.16) ones were closed with profits, 2655 (%18.04) ones with losses, and 117 (%0.8) ones with no gain or no loss.

A DNN with the selected architecture (5 hidden layers and a sliding window size of 3 minutes) was trained with the dataset B. The data set was split in the same way: The first 85% as training data and the remaining 15% as testing set. Figure **7-6** shows the strategy performance with the dataset B.

The simulation was performed with the same conditions of the previous experiment (only with the testing data, and the strategy buys or sells the equivalent of a dollar per trade). The strategy accumulated approximately \$2.87 in 38 trading days. It made 14691 trades, of which 13065 (%88.93) ones were closed with profits, 1601 (%10.90) ones with losses, and 25 (%0.17) ones with no gain or no loss.

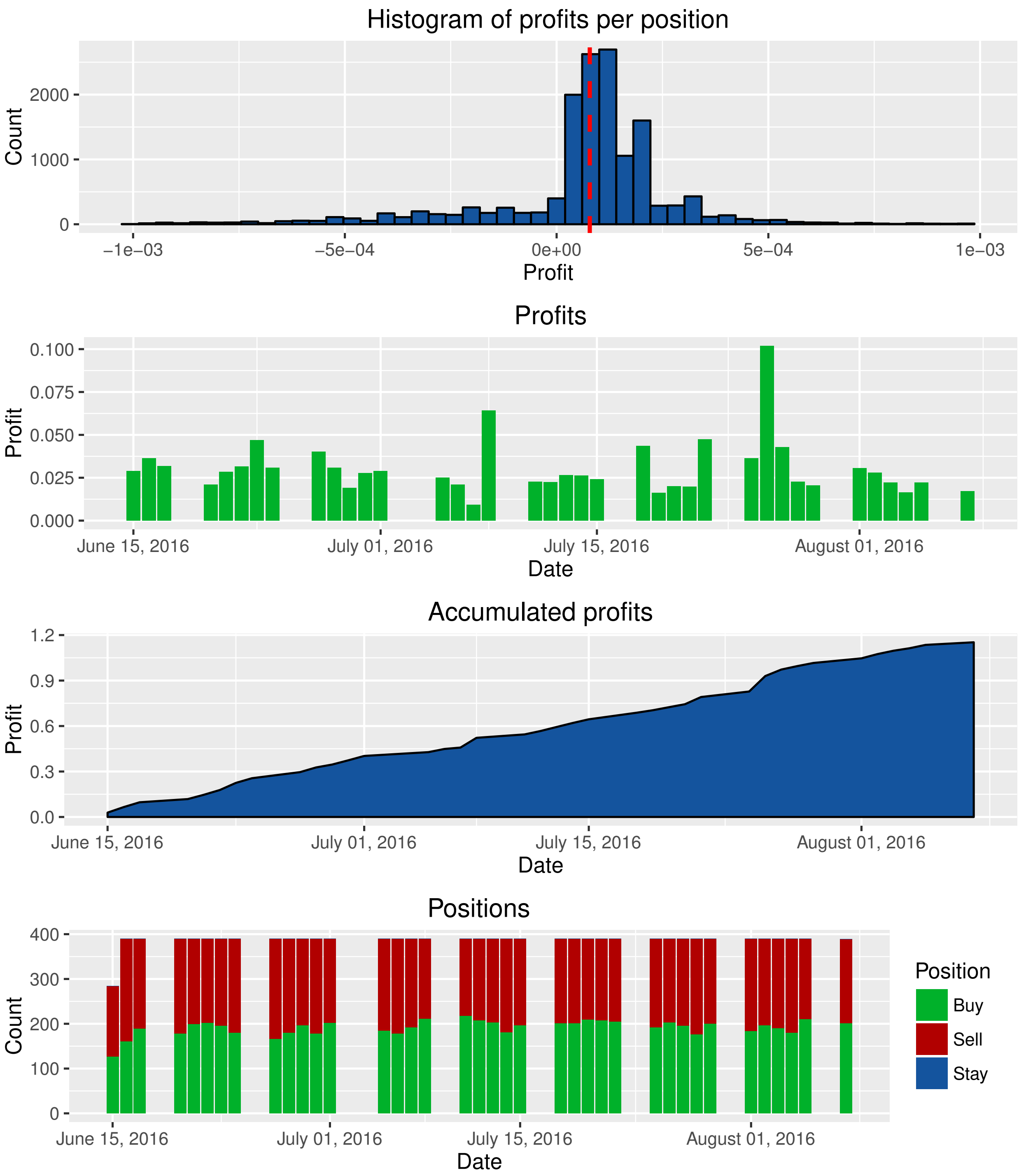


Figure **7-5**: Dataset A (AAPL): Trading Strategy Performance

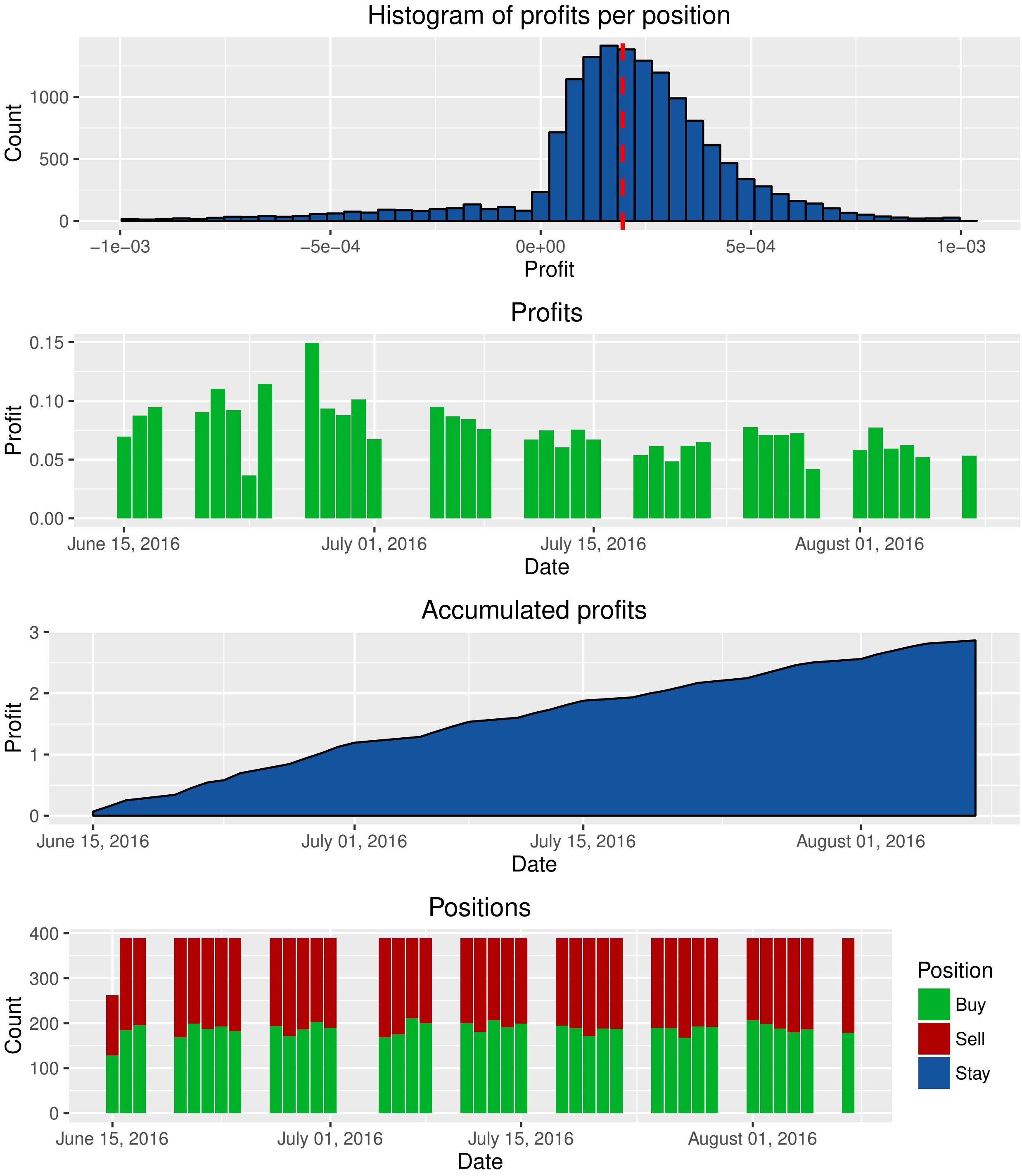


Figure **7-6**: Dataset B (GOOG): Trading Strategy Performance

Chapter 8

Conclusions

Although the strategy turns out to be interesting and yields a good performance, it must be refined in order to implement in a real environment, for example, it could analyse whether it closes its position in the next minute or it keeps it open in order to decrease transaction costs. Another improvement would be to include historical volume data or external news to the DNN inputs.

Traders collectively repeat the behaviour of the traders that preceded them [29]. Those patterns can be learned by a DNN. The proposed strategy replicates the concept of predicting prices for short periods. Furthermore, adding time as a DNN input allows it to differentiate atypical events and repetitive patterns in market dynamics. Moreover, small data windows sizes are able to explain future prices in a simpler way.

Overall, the DNNs can learn the market dynamic with a reasonable precision and accuracy. Within the deep learning arena, the DNN is the simplest model, as a result, a possible research opportunity could be to evaluate the performance of the strategy using other DL model such as Deep Recurrent Neural Networks, Deep Belief Networks, Convolutional Deep Belief Networks, Deep Coding Networks, among others.

Bibliography

[1] Arnold, L ; Rebecchi, S ; Chevallier, S ; Paugam-Moisy, H: An Introduction to Deep Learning. En: *ESANN* (2011)

[2] Arora, Anisha ; Candel, Arno ; Lanford, Jessica ; LeDell, Erin ; Parmar, Viraj. *Deep Learning with H2O*. 2015

[3] Bengio, Yoshua: Learning Deep Architectures for AI. En: *Foundations and Trends® in Machine Learning* 2 (2009), 1, Nr. 1, p. 1–127. – ISSN 1935–8237

[4] Chao, Jing ; Shen, Furao ; Zhao, Jinxi: Forecasting exchange rate with deep belief networks. En: *The 2011 International Joint Conference on Neural Networks*, IEEE, 7 2011. – ISBN 978–1–4244–9635–8, p. 1259–1266

[5] Dalto, M: Deep neural networks for time series prediction with applications in ultra-short-term wind forecasting. En: *Rn (Θ1)*

[6] De Gooijer, Jan G. ; Hyndman, Rob J.: 25 years of time series forecasting. En: *International Journal of Forecasting* 22 (2006), 1, Nr. 3, p. 443–473. – ISSN 01692070

[7] Ding, X ; Zhang, Y ; Liu, T ; Duan, J: Deep learning for event-driven stock prediction. En: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (ICJAI)* (2015)

[8] Dow Jones & Company, Inc. *Stock Scan: World's Blue Chips*. 2014

[9] Gallo, C ; Letizia, C ; Stasio, G: Artificial Neural Networks in Financial Modelling. (2006)

[10] Härdle, W ; Kleinow, T ; Stahl, G: Applied quantitative finance: theory and computational tools. (2013)

[11] Haykin, SS: *Neural networks and learning machines*. 2009

[12] Hinton, Geoffrey E. ; Osindero, Simon ; Teh, Yee-Whye: A fast learning algorithm for deep belief nets. En: *Neural computation* 18 (2006), 7, Nr. 7, p. 1527–54. – ISSN 0899–7667

[13] Hornik, K: Approximation capabilities of multilayer feedforward networks. En: *Neural networks* (1991)

[14] Intercontinental Exchange Inc. *TAQ NYSE Trades.* 2016

[15] Kaastra, Iebeling ; Boyd, Milton: Designing a neural network for forecasting financial and economic time series. En: *Neurocomputing* 10 (1996), 4, Nr. 3, p. 215–236. – ISSN 09252312

[16] Krollner, B ; Vanstone, B ; Finnie, G: Financial time series forecasting with machine learning techniques: A survey. (2010)

[17] Larochelle, H ; Bengio, Y: Exploring strategies for training deep neural networks. En: *The Journal of Machine Learning Research* (2009), p. 1–40

[18] Li, Xiaodong ; Huang, Xiaodi ; Deng, Xiaotie ; Zhu, Shanfeng: Enhancing Quantitative Intra-day Stock Return Prediction by Integrating Both Market News and Stock Prices Information. En: *Neurocomput.* 142 (2014), p. 228–238. – ISSN 0925–2312

[19] Marszałek, A. ; Burczyński, T.: Modeling and forecasting financial time series with ordered fuzzy candlesticks. En: *Information Sciences* 273 (2014), 7, p. 144–155. – ISSN 00200255

[20] Mills, TC ; Markellos, RN: The econometric modelling of financial time series. (2008)

[21] Nusca, Andrew ; Hackett, Robert ; Gupta, Shalene: Arno Candel, physicist and hacker, 0xdata. En: *Meet Fortune's 2014 Big Data All-Stars* (2014)

[22] Preethi, G ; Santhi, B: STOCK MARKET FORECASTING TECHNIQUES: A SURVEY. En: *Journal of Theoretical & Applied Information Technology* (2012)

[23] Riedmiller, M. ; Braun, H.: A direct adaptive method for faster backpropagation learning: the RPROP algorithm. En: *IEEE International Conference on Neural Networks*, IEEE, 1993. – ISBN 0–7803–0999–5, p. 586–591

[24] Schmidhuber, Jürgen: Deep learning in neural networks: An overview. En: *Neural Networks* 61 (2014), 10, p. 85–117. – ISSN 08936080

[25] Sureshkumar, KK ; Elango, NM: Performance analysis of stock price prediction using artificial neural network. En: *Global journal of computer science and Technology* 12 (2012)

[26] Takeuchi, L ; Lee, YYA: Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks.

[27] Tsay, Ruey S.: *Analysis of financial time series*. Vol. 543. John Wiley & Sons, 2005

[28] White: Economic prediction using neural networks: the case of IBM daily stock returns. En: *IEEE International Conference on Neural Networks*, IEEE, 1988. – ISBN 0–7803–0999–5, p. 451–458

[29] Wilder, J W.: *New Concepts in Technical Trading Systems*. Trend Research, 1978. – ISBN 9780894590276

[30] Yeh, SH ; Wang, CJ ; Tsai, MF: Corporate Default Prediction via Deep Learning. (2014)

[31] Yoshihara, Akira ; Fujikawa, Kazuki ; Seki, Kazuhiro ; Uehara, Kuniaki: PRICAI 2014: Trends in Artificial Intelligence: 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014. Proceedings. Cham : Springer International Publishing, 2014. – ISBN 978–3–319–13560–1, Kapitel Predicting, p. 759–769

[32] Zeiler, Matthew D.: ADADELTA: An Adaptive Learning Rate Method. (2012), 12, p. 6

[33] Zekic, M: Neural network applications in stock market predictions-a methodology analysis. En: *Proceedings of the 9th International Conference on Information and Intelligent Systems* (1998)