**Machine Learning and Stock Prediction**

**Maria Eduarda P. Santana1, Julio L. R. Monteiro2, Arthur M. Rosa3, Paulo D’Angelo4**

**Abstract:** The rules behind market dynamics and stock market prediction are topics of intense debate. Fundamental and technical analysis have been used during the past decades to find patterns and to try to refute the Efficient Market Hypothesis. The current prediction techniques make use of more robust algorithms capable of learning with chaotic scenarios and making more accurate forecasts. This paper analyzes different machine learning techniques for stock market prediction and compares their efficiency with a random walk algorithm. The first technique, ARIMA, is a traditional time series analysis based on linear regression and graphical observation. The second technique, LSTM, is a deep learning algorithm based on continuous input data. Comparisons are made evaluating their performance using B3’s public market data for some Brazilian’s prominent stocks: PETR4, ITUB4 and BVMF3. The result indicates the high unpredictability of the market during the month of May 2017.

Keywords: financial market, market data, stock predicition, ARIMA, LSTM, B3, artificial intelligence, machine learning, deep learning.

1. **Introduction**

The extensive debate whether stock prices are random or chaotic was revolutionized by the age of artificial intelligence. In 1965, Eugene Fama presented the Efficient Market Hypothesis and suggested that current prices reflected all information available[1]. According to Fama, it is impossible to anticipate new political, social and economic information and, as a result, the market movement is unpredictable. This theory of randomness was followed by the idea that the market might be, instead, a complex, chaotic, and deterministic interaction between macro and microeconomic seasonal factors[2]. This synergy would result in a path that might seem random but, actually, it would follow a collection of rules. Nowadays, using machine learning algorithms and historical data, it is possible to detect remote non-linear patterns and question the unpredictability of the market. On the other hand, a random walk market is still a compelling hypothesis, specially for emerging volatile markets[3].

A chaotic market, unlike a random one, follows a set of cause-and-effect phenomena. Considering the atmosphere, it is hard to forecast the weather due to the large number of variables involved and because a small change in the initial structure might significantly affect the system – the so called butterfly effect[4]. Meteorologists, consequently, make educated guesses with the information they have available. That doesn’t mean it will suddenly start snowing in a warm weather. The previous state influences the next because it is a deterministic system ruled by gravity, chemical relationship of gases, physics of the wind, and other factors that band together. That is a chaotic system. The rules are hard to understant, but they exist.

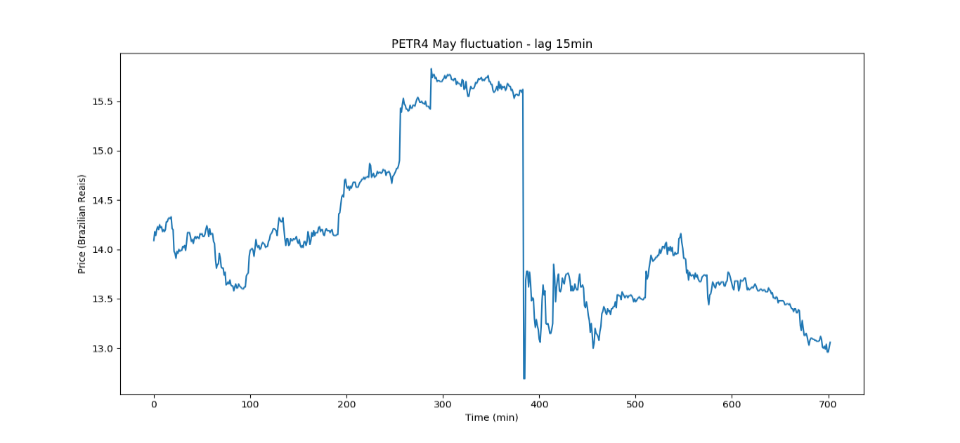
On the other hand, tossing a coin is random. There aren’t hidden rules interacting to determine if one gets head or tail, there are equal chances of getting any for every time the coin is flipped. It is like a snow, sun simultaneous probability that could be modeled by a random walk algorithm. Both chaotic and random systems sound unpredictable, but the chaos just seems so because it is not trivial how the hidden factors interact. That is when machine learning comes in, unfolds the mysterious layers of information and tries to predict what seems disordered.

Modern analysis suggests that the market might be chaotic, instead of random; which means an intelligent algorithm could understand the intricated relations, identify seasonal patterns and forecast future scenarios[5]. The purpose of this paper is to study this by comparing two machine learning methods with a random walk algorithm and discuss the results within the Brazilian market scenario.

The difference between machine learning and traditional programming is that, in machine learning, computers learn with a train dataset and create programs to predict new outputs. Consequentely, computers can learn with the past and alter its program without human intervention. There are two main machine learning techniques: supervised and unsupervised learning. In supervised learning, computers receive the input x, the targeted output y and they must find the function f for f(x) = y, which is called inductive learning. Meanwhile, in unsupervised learning, the machine only receives the input x, which is usually large and without clear lables, and must find patterns in the data[6]. Since in stock prediction the time (x) and price (y) direct coorelation is available, supervised learning is the technique explored in this paper.

Furthermore, machine learning can be divided into classification, regression, clustering and other classes of problems[7]. In classification, there are a number of predetermined categories for the output. For example, one could build and algorithm to predict if it is going to rain and the answer would be either “yes” or “no”. On the other hand, one could build a regression algorithm to measure the rainfall in inches and, in this case, the output would be a number instead of a set class of answers. That is, in classification problems the function being learned is discrete, whereas in regression the function is continuous[6].

The problem of stock forecasting, as the rain example, can be studied both ways. A classification algorithm would say if the stock would go “up” or “down”, and a regression algorithm would predict a real value for that stock. The machine learning techniques specifically analyzed in this paper are the Auto-Regressive Integrated Moving Average (ARIMA), a well-known forecast model, and the Long Short Term Memory model (LSTM), a recurrent neural network algorithm. The ARIMA model is commonly used for linear regression forecast and the LSTM algorithm was also built as a regression problem for comparison purposes.

1. **Data and preprocessing**

The data used in this paper is the public historical market data provided by B3, available on its website through a FTP server. The timeseries are related to trade history and each data file represents trades for stocks of equities market during a session day.

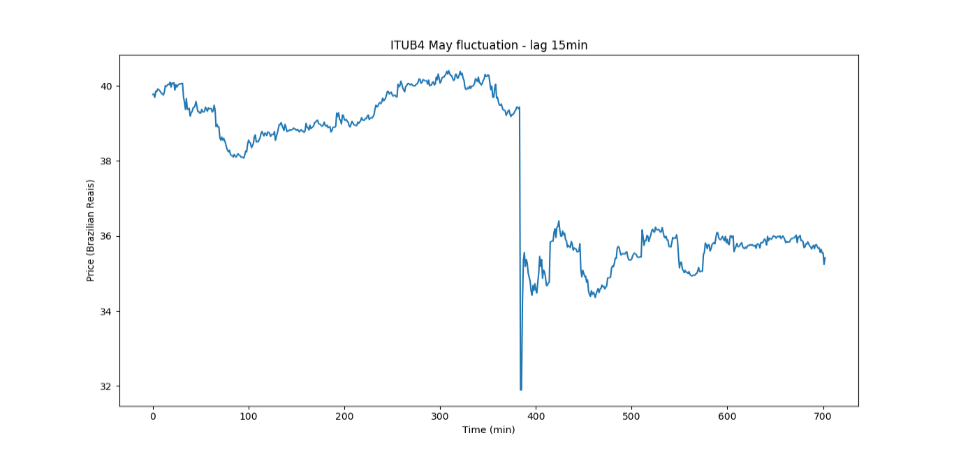
The data was preprocessed and reduced to a small set of stocks. The stocks chosen are significantly traded in the Brazilian market and their liquitidities verify how fast the algorithms can detect changes in momenta. PETR4 is Petrobras’ preference share, Brazil’s largest oil and gas company; ITUB4 is Itau Unibanco’s preference share, the largest bank in Latin America; and BVMF3 is B3’s ordinary share, the world’s fifth largest stock exchange in terms of value[9].

Figure 1 - PETR4 fluctuation during May 2017

Figure 2 – ITUB4 fluctuation during May 2017

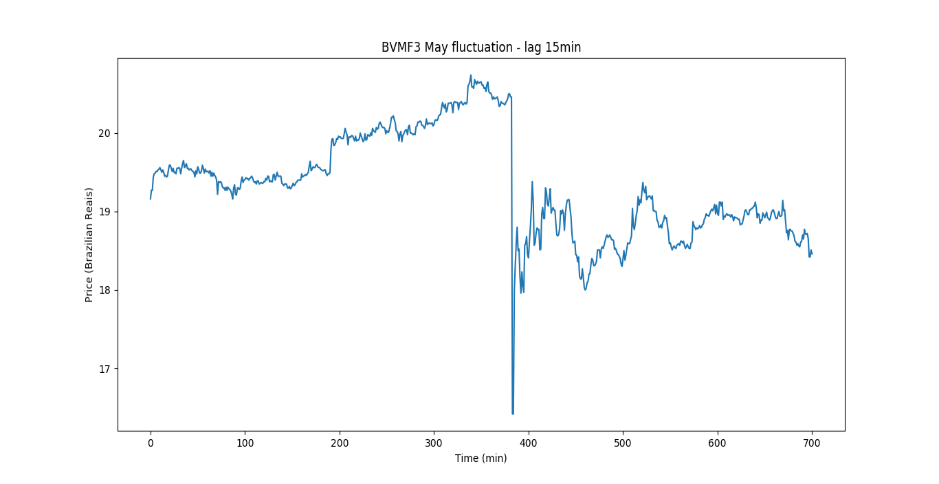
The trade prices were selected in fixed intervals of 15min – and then tested with 10min and 5min lags of observation – for the month of May 2017. Thus, about 700 points were inserted in each algorithm through a file containing the stock price indexes. Furthermore, the algorithm environment Anaconda[10] uses Python as its primary language, and Keras (running on Tensorflow) as the neural networks API.

Figure 3 – BVMF3 fluctuation during May 2017

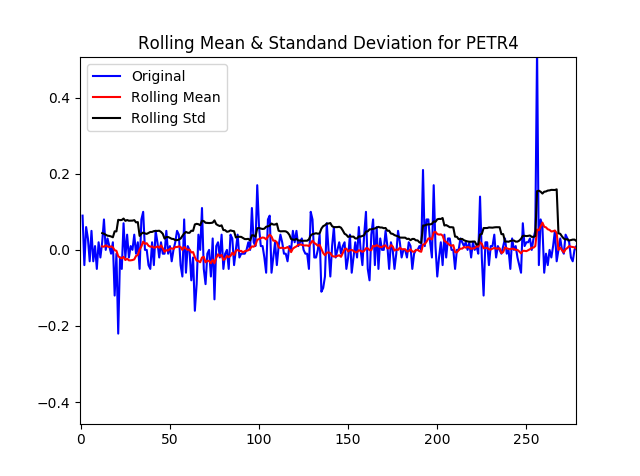
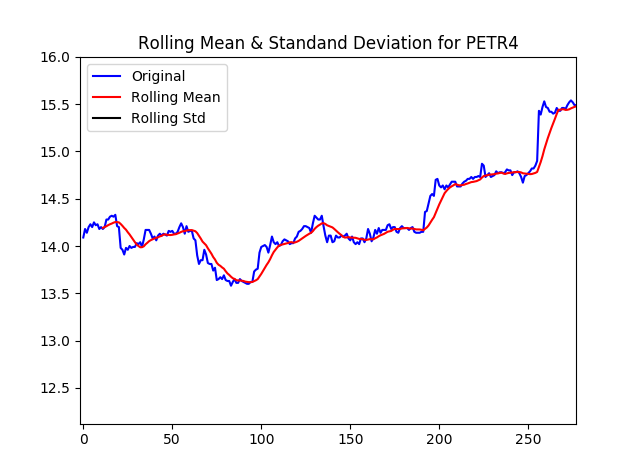
PETR4, ITUB4 and BVMF3 showed a similar pattern over time, indicating the coorelation of these stocks in the Brazilian market. Moreover, the up-down close correspondence revealed how external political and economical factors played an important role over internal corporate governance during May 2017.

The general downfall observed in the graphs happened during May 18th, when the main index Ibovespa (IBOV) fell 10.47% compared to the previous day’s closing price[11]. The night before, JBS’s president Joesley Batista had released an audio associating Brazil’s president Michel Temer to a scheme of bribe and corruption. The IBOV indicator includes the 50 most traded assets at B3 and, together, they represent over 80% of the stock value traded nacionally. Consequentely, the significant fall of the index due to the political turbulence resulted in a 30min circuit breaker to contain the high volatility of the market. As a result, this paper also aims to analyze how the different algorithms reacted to this sudden and unexpected market breakdown.

1. **Stationarization**

In order to make forecasts, it is necessary to find the time invariant component of a series and project that to the future. If the market behavior were completely different every day, it would be impossible to predict the next scenario. However, if there is a constant behavior over time, there is a pattern to be explored. A stationarized series, on those lines, is a series with constant mean, constant variance and constant covariance of terms over time. ARIMA requires a stationarized series as an input and, to observe the correspondence, the same stationarized series was used for LSTM. If the initial series is non-stationary – which is usually the case for real world problems – it must be transformed using logging, deflating, differencing, decomposing or other stabilizing transformation[12].

Figure 4 – Comparison between a non-stationarized series (left) and a stationarized series (right)



The logging transformation is common and useful for up-trend series. However, since the stock indexes used in this paper don’t have a clear up trend over the month of May, differencing was applied without the logarithmic transformation. The best way to identify if the series is stationary after the transformation is using the Augmented Dickey-Fuller Test[13], which can be found in the Statsmodel Python library. The test verifies the null hypothesis, which, if accepted, means that the time series has a unit root and is non-stationary and, if rejected, means that the series does not have a time dependent structure and is stationary[14].

It is possible to verify this dependence through two main results of the adfuller function[REF]: the p-value and the test statistics. If the p-value is less than a specified chosen significance level (usually 5%) and if the test statistic is smaller than the critical values, it is possible to conclude it is stationary and insert the series in the ARIMA and LSTM algorithms. For this study, the timeseries for BVMF3, PETR4 and ITUB4 reached stationarity with one order of differencing (p-values < 0.001 and Test Statistics smaller than the critical values).

The partial timeseries of PETR4 before and after differencing are in Figure 4, enabling to observe the stabilizing transformation.

1. **ARIMA (Autoregressive Integrated Moving Average)**
2. Algorithm

ARIMA[8], also called Box-Jenkins method, is one of the most traditional machine learning models for forecasting time series. A time series is a sequence of points indexed in a certain time order. In this part of the study, stock prices were indexed with a 15min lag observation.

The model combines the Auto Regressive model (AR), which observes the relationship between previous numbers; integration (I), which uses differencing to make the time series stationary, and the Moving Average model (MA), which observes the dependency between previous residual errors. This elements are respectively sent to the function as the ARIMA (p,d,q) parameters.

*Parameters (p,d,q)*

There are mainly two ways to set the parameters. The first is the iteration method, in which the program tests a series of numbers and chooses the ones with best optimization. However, this approach when used alone is slow and requires a lot of machine processing. The second approach is the analysis of correlation (ACF) and partial correlation (PACF) graphs. There are a set of rules to interpret the graphs and, most times, slight variations should be tested to find the best set of parameters. The easiest way to determine ‘p’ (for Autoregressive) and ‘q’ (for Moving Average) is that ‘p’ is the lag value in which the PACF graph drops to zero for the first time, whereas ‘q’ is the lag value in which the ACF graph crosses the line of statistical significance\* for the first time[15]. A more detailed approach can be found in Duke University’s professor Robert Nau’s notes for Statistical forecasting[12]. In this paper, a hybrid approach was used. The graphic analyzis was realized and, then, was followed by some close iterations to guarantee the best set of parameters.

The order of differencing ‘d’ is rarely larger than 2 in financial algorithmsand overdifferencing can introduce unnecessary levels of dependency[16]. It is a situation in which the algorithm looks too much into the past and finds patterns that are not compatible with the maket’s fast dynamic. Therefore, this study tested a lag 1 order of differencing. After setting the parameters and testing the algorithm, it is usually a good practice to analyze the ACF plot for residual errors to certify that all trends were identified and the residuals are just white noise (non-significant values).

1. Implementation

In order to implement the algorithm, it is necessary to divide the dataset into train and test data. The model uses the train data to progressively learn a pattern and the test data to evaluate how much it has actually learned. The data was divided 65% for training and 35% for testing.

The optimum parameters (p,d,q) were slightly different for each stock, but the process for finding them was equivalent. In order to demonstrate the graphic analysis, PETR4 will be used as an example.

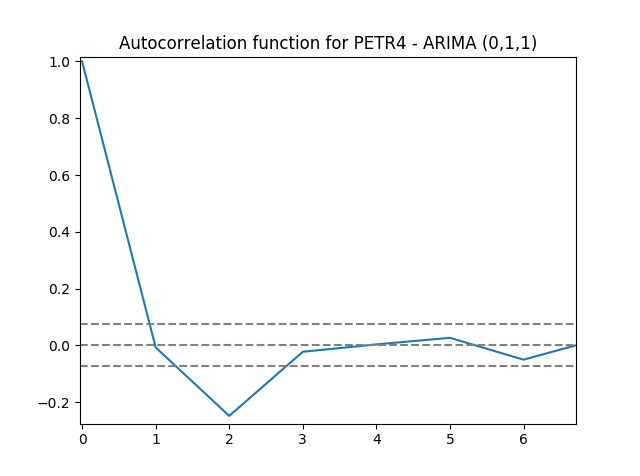
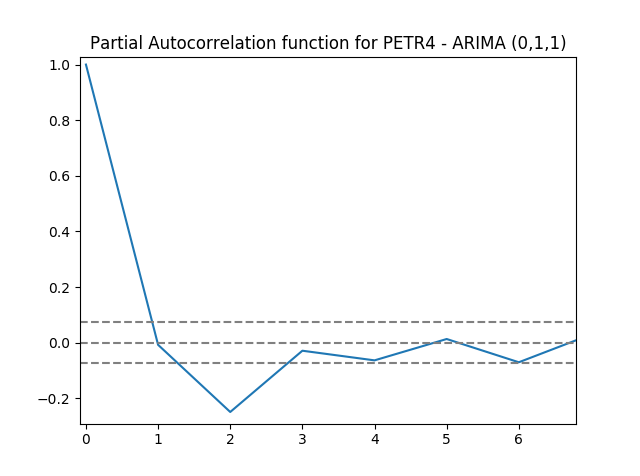
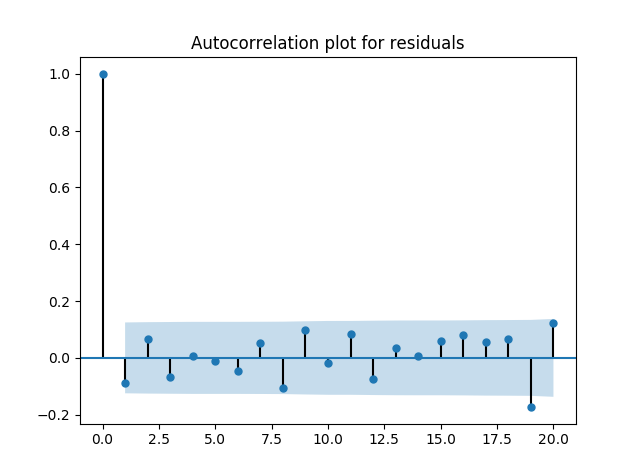


Figure 5 - ACF and PACF for PETR4

Plotting the autocorrelation and partial autocorrelation for PETR4 (Figure 5), the autocorrelation plot crosses the line of statistical significance[[1]](#footnote-1) between 1, and the partial autocorrelation plot drops to zero around 1. As a result, iterations for ARIMA (0,1,1), ARIMA (1,1,0) and ARIMA (1,1,1) were executed; the first one presenting a slightly better result.

In order to validate the algorithm, the ACF graph for residuals was plotted for the best set of parameters ARIMA (0,1,1) (Figure 6). As noticed, the residuals don’t have a statistical significant coorelation (they are all inside the blue area[[2]](#footnote-2)) and, consequentely, the algorithm is well parametrized.

Figure 6 - ACF for PETR4 residual errors

*.*

1. Root Mean Squared Error[[3]](#footnote-3)

The the root mean squared error was used as the measure of accuracy, representing the standard deviation of the prediction errors (difference between predicted values and test values).

A prodecure equivalent of the one described in parts A and B was used for the other stocks (ITUB4 and BVMF3) and the RMSE results are compiled in the Table 1 bellow.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **ARIMA (0,1,1)** | **ARIMA(1,1,0)** | **ARIMA(1,1,1)** |
| PETR4 | 0.04934 | 0.04935 | 0.04958 |
| ITUB4 | 0.10901 | 0.10882 | 0.11225 |
| BVMF3 | 0.07148 | 0.07112 | 0.07419 |

Table 1 - Comparison between ARIMA models for all assets

As observed, the RMSE are really similar for p = 0 - 1 and q = 0 - 1, which indicates that the way the previous stock values coorelate (Autoregressive part of the algorithm) is similar to the way their residual errors coorelate (Moving Average part of the algorithm).

1. Prediction plots

The prediction graphs were ploted using the best parameters for each stock (according to Table 1). The graph for PETR4 (Figure 7) is an example of the prediction for the first 100 price values using ARIMA (0,1,1) model.

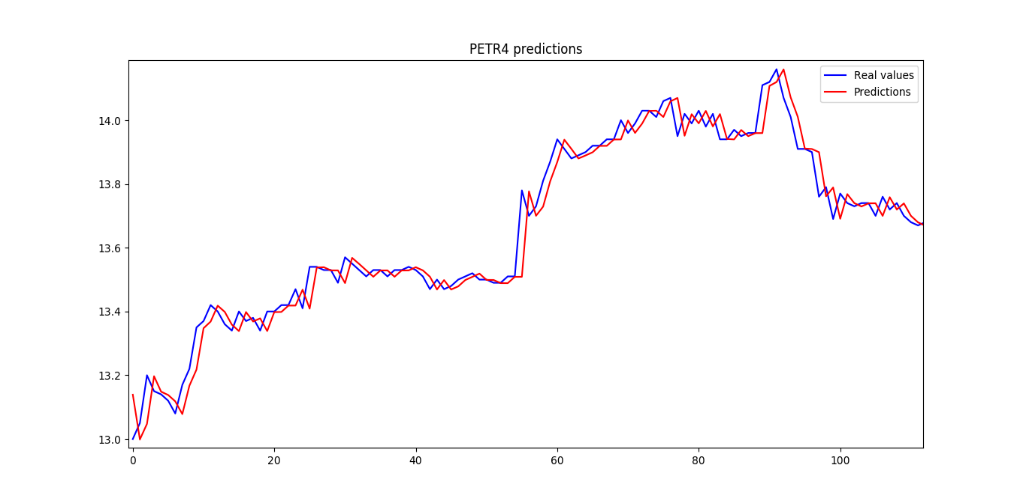


Figure 7 - PETR4 predictions for the month of May 2017

1. **LSTM (Long Short Term Memory)**
2. Algorithm

Artificial Neural Networks are mathematical models inspired by biological neural networks. They are composed of interconnected units called artificial neurons that process information from an input to an output layer. In between, there are hidden layers, where the weights of the algorithm are adjusted. The application of models with multiple hidden layers, as in this paper, is called deep learning. Nowadays, deep learning is the latest and most popular branch of artificial intelligence. The algorithms are used to process complex datasets and are intensely applied in the areas of speech and image recognition, computer vision and forecasting.

As keys to pattern identification, neural networks have been decisive to those who believe the market is not random, but chaotic. Thus, the neural network’s ability to capture remote patterns makes it ideal for modeling chaotic systems under apparently hidden rules[5]. However, stock prediction is based on historical data observation, and regular neural networks move forward and are unable to storage information that might be valuable to understand seasonality. As a result, the advance of a type of network with a “memory”, called Recurrent Neural Network (RNN), significantly improved timeseries prediction algorithms. The difference between a regular ANN and a RNN is the structure of the artificial neurons (Figure 8).

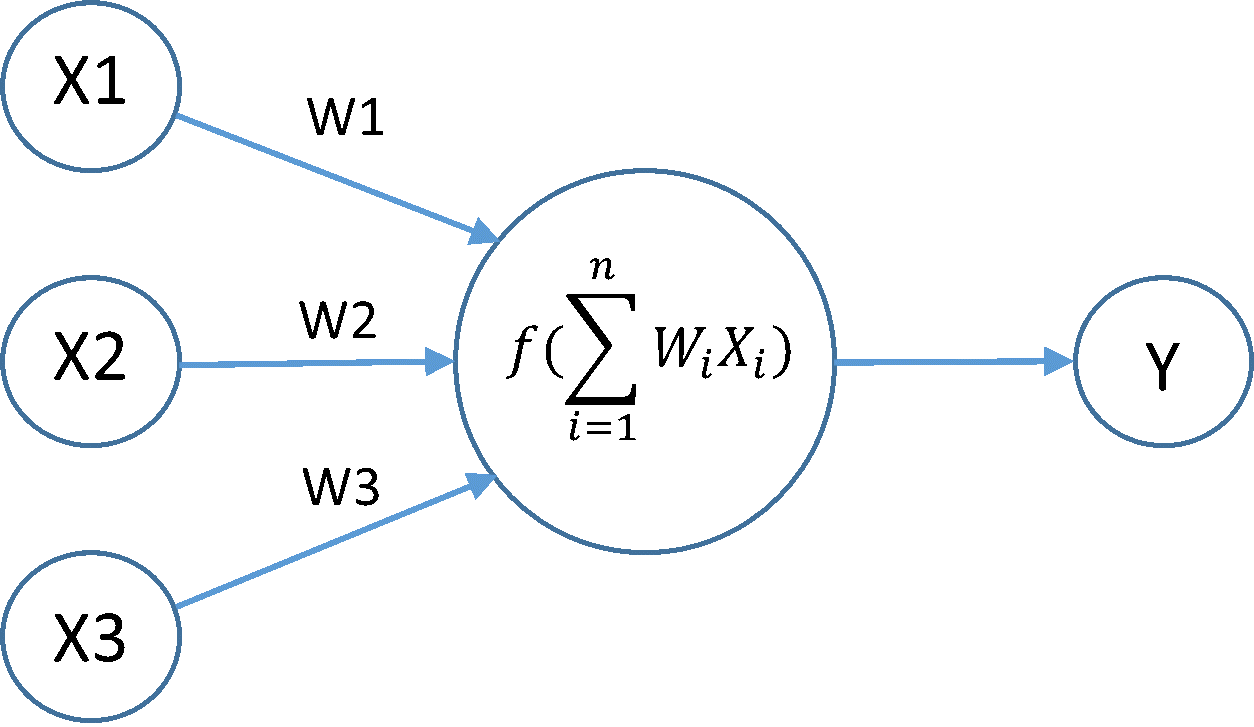
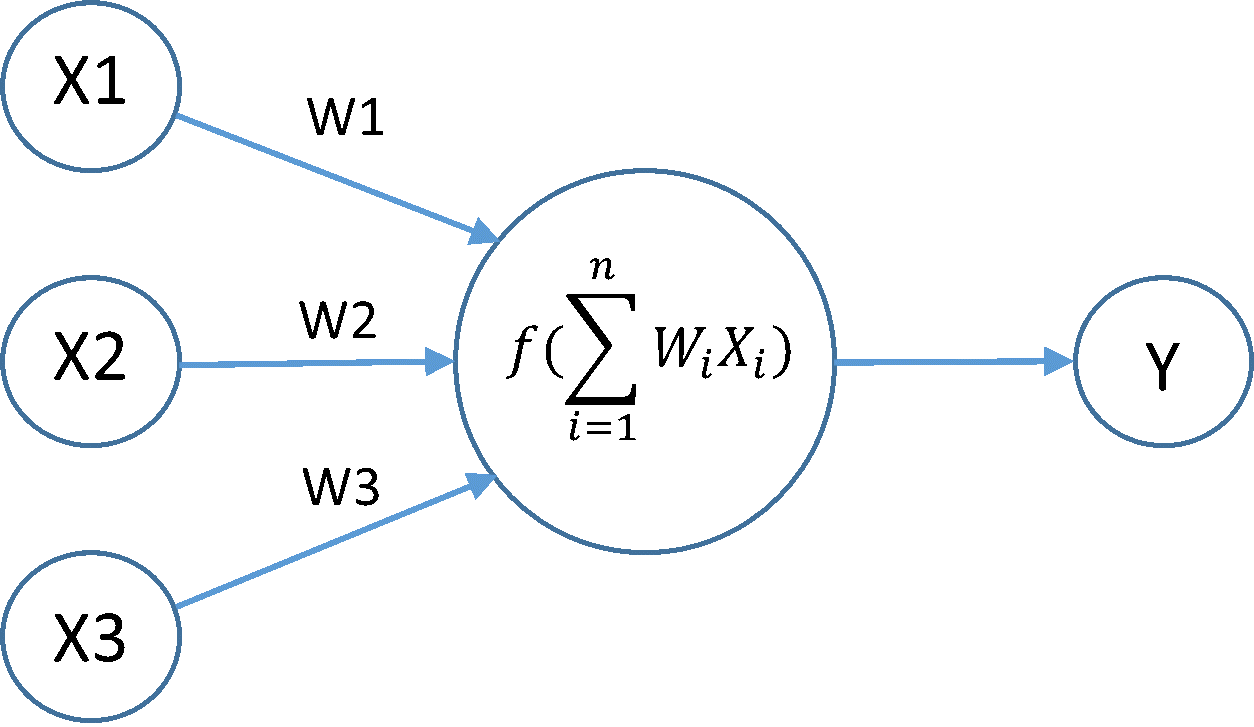


Figure 8 - ANN neuron (left) and RNN neuron (right). Source: Stack Overflow (website)

The neuron is a unit in which the inputs (x1, x2 and x3) are processed and the weights (w1, w2 and w3) determine the relevance of each input to the output[17]. In Figure 8 (left), the sum of the products of inputs and weights is inserted in an activation function *f* inside the neuron – that could be a linear function, a sigmoid, a hyperbolic tangent or other function suitable for the dataset. On the other hand, in an RNN, the neuron not only receives the new inputs x1,x2 and x3, but also receives the result of the previous iteration through the loop. This way, it allows information to persist.

These Recurrent Networks are specially used in problems with continuous inputs, once they can progressively process information without losing it layer after layer. However, the loop only allows the neuron to look at the immediate previous layer and, sometimes, there is relevant information further back. For such cases, there is a special kind of Recurrent Neural Network called Long Short Term Memory (LSTM), able to select and forget information throughout the layers using four selective internal gates.

A detailed description of the LSTM neuron (Figure 9) reveals the four gates - three with sigmoid functions and one with a hyperbolic tangent function - responsible to update the cell state and pass it forward[18]. The diagram is similar to RNN Figure 8 (right), exept that in Figure 8 there is only one activation function and the input-output is arranged left-right while, in Figure 9, it is down-up for clearness. Along these lines, the difference between a RNN model and a LSTM model resides in the number of internal gates and the fact that the LSTM sums the weight updates instead of multiplying them[[4]](#footnote-4). The algorithm built for this research was an LSTM model with two hidden layers.

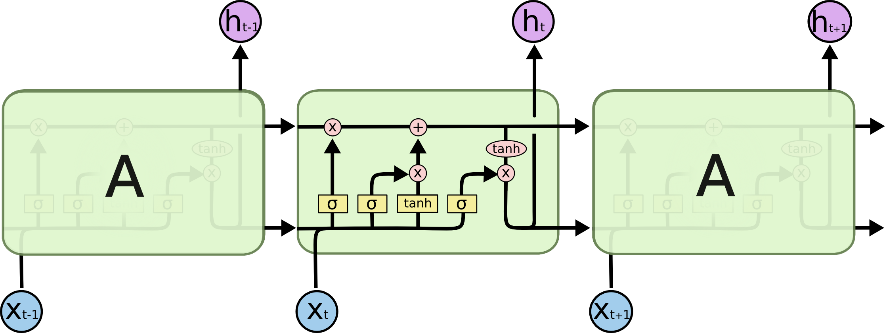


Figure 9 - LSTM neuron

1. Implementation

As the ARIMA model, it is necessary to divide the dataset into train and test data. For comparison purposes, the same dataset proportion of 65% for training and 35% for testing was used.The initial arrangement of weights is random and they are tunned as the algorithm is trained. Furthermore, the model is trained using backpropagation through time (BPTT), a gradient-based technique to minimize the cost function by moving backwards and readjusting the parameters[20].

One epoch represents the exposure of the algorithm to all training examples. However, the model doesn’t have to consume all the data at once (full batch learning) and update the parameters only at the end of each epoch. It is possible to use two methods to divide the training dataset:mini-batches or stochastic gradient descent.

When using mini-batches, the batch size is pre-set and the training examples are divided accordingly. For example, if there are 500 training examples and the batch size is 250, it will take 2 iterations to complete one cicle (epoch). Instead of updating the weights after the epoch is complete, the algorithm updates them after each iteration, making it possible to arrive at an optimum solution faster. Meanwhile, stochastic gradient descent is really similar, exept that it sets the batch size equal to one. Consequentely, the algorithm will iterate and update the weights every time it sees one new example.

Deciding which batch strategy to use is an important part of LSTM algorithms. Big batch sizes require more memory space, while small ones might suffer with ineccuracy if the algorithm is not programmed to maintain the cell state between batches[21]. Stochastic gradient descent method was used in this paper because, as ARIMA, the goal is to make one step forecasts - predict the next stock price after each point. Also, with the intention of maintaining memory after each iteration, the Keras implementation has *stateful* parameter of the LSTM function that was set to *True*, meaning the neuron state is preserved after each batch, instead of being reset.

In addition to the batch size, the number of epochs is, likewise, an important parameter to the LSTM model. After each epoch, the state of the cell is reset and the whole process is repeated for a set number of times. If this number is small, the algorithm might not learn enough from the train dataset before making predictions for the test dataset. However, if the number of epochs is too large, the model might suffer for overfitting – the algorithm memorizes all the noise from the train dataset and doesn’t perform well on new data[22]. The number of epochs used in this paper was 50, which, after some tests, proved to be a reasonable number for the 700 samples stock files.

Another feature that can also cause overfitting is the number of neurons in the hidden layers. There are a set of general rules explored in Jeff Heaton’s book “Introduction to Neural Networks for Java”[23] and the number of neurons is usually close to the size of the input and output. Still, as the author mentions, the rules are just starting points and it is important to iterate around those numbers. Since, for the case studied, each input and output have only one element at each iteration, small values were tested and 3 neurons for layer was the optimum parameter.

Because the initial conditions for the weights is random, the algorithm gives slightly different results each time it runs. In order to diminish this random component, the program was executed 10 times and an average of the RMSEs was used to compare with ARIMA’s result.

1. Root Mean Squared Error

The Root Mean Squared Error was also used to evaluate the accuracy of the algorithm. Some parameters were manually changed to analyse overfitting and the effect of the number of hidden layers.

|  |  |  |
| --- | --- | --- |
|  | **RMSE Test** | **RMSE Train** |
| PETR4 | 0.04933 | 0.00976 |
| ITUB4 | 0.11141 | 0.02689 |
| BVMF3 | 0.07356 | 0.01891 |

Table 2 - Comparison between test and training errors

The difference between the train and test errors can be explained by the origin of the data. The train dataset has been repeatedly examined by the algorithm during the training, whereas the test data has never been seeing by the model and requires its ability to generalize. As a result, the error is expected to be bigger. However, the test RMSE gives a better estimate of how the algorithm would perform in a real scenario. If the difference were even larger, it would be a sign of overfitting – the network would be performing too well on the train and poorly on the test section.

The number of hidden layers was also reduced to one in order to verity the relevance of the second hidden layer. In this case, the model is a simple LSTM and doesn’t involve deep learning. A table for Test RMSEs is presented bellow for one and for two hidden layers.

|  |  |  |
| --- | --- | --- |
|  | **2 Layers** | **1 Layer** |
| PETR4 | 0.04933 | 0.04949 |
| ITUB4 | 0.11141 | 0.11056 |
| BVMF3 | 0.07356 | 0.07440 |

Table 3 - Comparison of errors with different hidden layers

Even though the 2 hidden layer’s model showed slightly better results, the errors in general were really similar, suggesting that the addition of a second layer might not significantly change the output for a montly dataset. Perhaps, if the dataset were bigger, it would be possible to see the impact of adding a second hidden layer to detect hidden patterns.

1. Prediction plots

The prediction graphs were ploted using for each stock and the graph for PETR4 (Figure 10) is an example of the prediction for the first 100 price values and with the 2 hidden layers model.

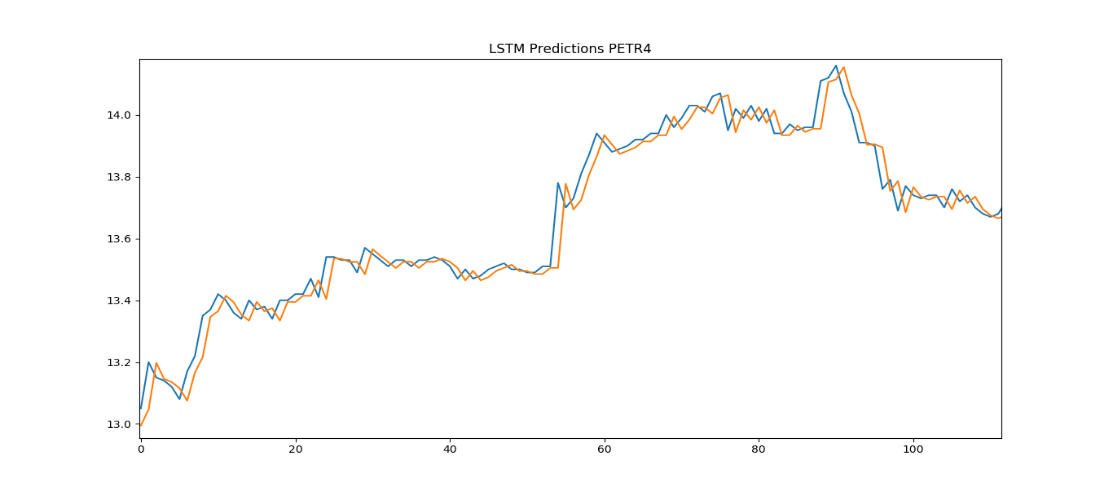


Figure 10 - LSTM PETR4 predictions for May 2017

1. **Comparative analysis of algorithms**

As observed in tables 1,2 and 3, the RMSEs were really similar for both algorithms, LSTM or ARIMA performing slightly better depending on the asset. An analysis of the simple error in cents using Whisker’s plots[24] was made to compare the algorithms’ performances. Moreover, a table was added to compare them with a Random Walk algorithm and test the unpredictability of the market for this specific dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| **PETR4** | ARIMA | LSTM | Random Walk |
| Mean | 0.032137 | 0.033582 | 0.032152 |
| STD | 0.037512 | 0.036570 | 0.037526 |
| Min | 0.000173 | 0.000836 | 0.000114 |
| Max | 0.271152 | 0.279962 | 0.271137 |

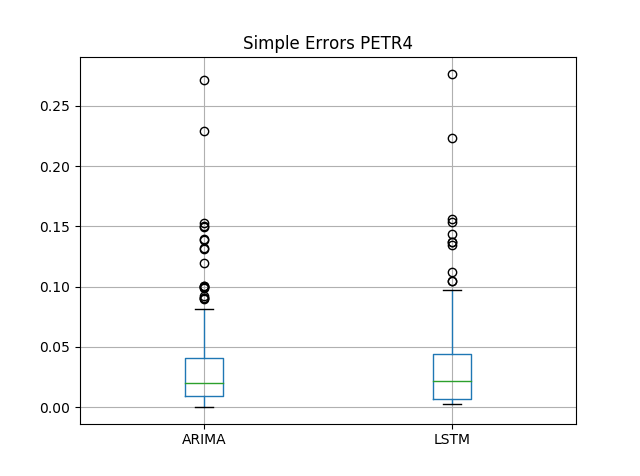


Table 4 - PETR4 errors for each model

Figure 11- Simple Errors PETR4

|  |  |  |  |
| --- | --- | --- | --- |
| **ITUB4** | ARIMA | LSTM | Random Walk |
| Mean | 0.070288 | 0.073057 | 0.070199 |
| STD | 0.082618 | 0.082522 | 0.082526 |
| Min | 0.000104 | 0.001414 | 0.000973 |
| Max | 0.728268 | 0.737137 | 0.728490 |

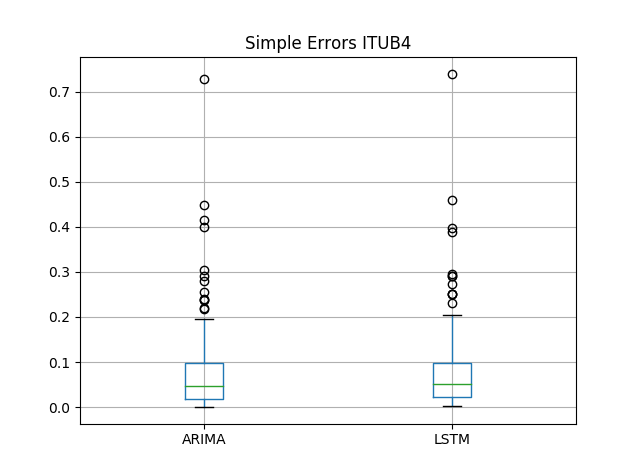
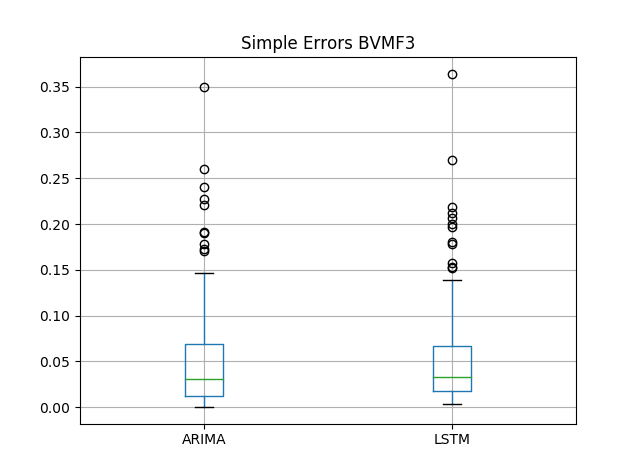


Table 5 - ITUB4 errors for each model

Figure 12- Simple Errors ITUB4



|  |  |  |  |
| --- | --- | --- | --- |
| **BVMF3** | ARIMA | LSTM | Random Walk |
| Mean | 0.049130 | 0.052281 | 0.048930 |
| STD | 0.051637 | 0.051110 | 0.051528 |
| Min | 0.000253 | 0.002413 | 0.000066 |
| Max | 0.349344 | 0.368737 | 0.350923 |

Table 6 – BVMF3 errors for each model

Figure 13- Simple Errors BVMF3

A Random Walk Algorithm can be built using ARIMA(0,1,0), a model in which there is no autocorrelation between the terms and between the residual errors. The next observation is equal to the previous one with a random step up or down[25]. This method substantiate the Efficient Market Hypothesis, once it says the movement of the market is random and cannot be predicted. As observed, two of the three assets analysed had better results for the random walk algorithm than for the machine learning models.

1. **Conclusion**

The results of this research suggests that machine learning can be applied in financial market and show good results, but the market was still highly unpredictable in the studied scenario. It is important to consider that the assets being analysed are part of a volatile emergent market in a month of political turbulence. Consequentely, they might indeed, follow a random walk or be hard to predict with one month of data.

This research’s results can be also used to compare ARIMA and LSTM. The RMSEs were really similar for both algorithms and, although ARIMA is faster and demands less machine processing, it requires a careful parametrization with a lot of graphic analysis. Meanwhile, LSTM adjusts the weights and biases by itself and just requires the epochs, batch sizes and general parameters that can be mostly found by iteration.

Some changes that could be made for future analysis is the increase of the dataset, including more months or years, to allow the algorithms to find remote patterns. Moreover, the price could be coorelated to other parameters as momentum, volatility, other currencies and indexes, instead of only the asset’s historical data. Furthermore, the LSTM parameters could be changed (number of epochs increased) or a moving simulation could be used. In other words, the system could be trained with January data, tested on February data and used to predict March data. Then, afterwards, the window would move and the model would be trained with February, tested with March and used to predict April[5].

It was observed how machine learning can be used to predict stock prices and how the Brazilian financial maket behaves in the random vs chaotic dilema. As most emergent countries, the behavior is inconstant and more data might be necessary to account for these fluctuations[3]. Many financial institutions are tuning to techniques based on machine learning to be able to cope with the amazing quantity of data produced nowadays. The challenge, now, is to find the best algorithms and organize the data to fit with specific economic and political scenarios. The data and code used for this project is stored in the github: *(link)*.

1. **References**

[1] LO, Andrew W. Efficient markets hypothesis. 2007.

[2] HSIEH, David A. Chaos and nonlinear dynamics: application to financial markets. **The journal of finance**, v. 46, n. 5, p. 1839-1877, 1991.

[3] ZEREN, Feyyaz; KONUK, Filiz. Testing the random walk hypothesis for emerging markets: evidence from linear and non-linear unit root tests. **Romanian Economic and Business Review**, v. 8, n. 4, p. 61, 2013.

[4] LORENZ, Edward. The butterfly effect. **WORLD SCIENTIFIC SERIES ON NONLINEAR SCIENCE SERIES A**, v. 39, p. 91-94, 2000.

[5] LAWRENCE, Ramon. Using neural networks to forecast stock market prices. **University of Manitoba**, 1997.

[6] BROWNLEE, JASON. Basic Concepts in Machine Learning - Machine Learning Mastery. Machine Learning Mastery. Available at: <http://machinelearningmastery.com/basic-concepts-in-machine-learning/>. Accessed: 1  Aug.  2017.

[7] MADGE, Saahil. Predicting Stock Price Direction using Support Vector Machines. **Independent Work Report Spring**, 2015.

[8] PAI, Ping-Feng; LIN, Chih-Sheng. A hybrid ARIMA and support vector machines model in stock price forecasting. **Omega**, v. 33, n. 6, p. 497-505, 2005.

[9] ALVES, LISE. Merger in Brazil Creates 5th Largest Stock Market in the World | The Rio Times | Brazil News. The Rio Times. Available at: <http://riotimesonline.com/brazil-news/rio-business/merger-in-brazil-creates-5th-largest-stock-market-in-the-world/>. Accessed: 1  Aug.  2017.

[10] Boschetti, Alberto, and Luca Massaron. *Python data science essentials*. Packt Publishing Ltd, 2016.

[11] TREVIZAN, KARINA. Após ter pregão interrompido, Bovespa retoma atividade e opera em forte queda. G1. Available at: <http://g1.globo.com/economia/mercados/noticia/bovespa-180517.ghtml>. Accessed: 1  Aug.  2017.

[12] NAU, ROBERT. Statistical forecasting: notes on regression and time series analysis. people.duke.edu. Available at: <http://people.duke.edu/~rnau/411home.htm>. Accessed: 1  ago.  2017.

[13] MUSHTAQ, Rizwan. Augmented Dickey Fuller Test. 2011.

[14] BROWNLEE, JASON. How to Check if Time Series Data is Stationary with Python. Machine Learning Mastery. Available at: <http://machinelearningmastery.com/time-series-data-stationary-python/>. Accessed: 1  Aug.  2017.

[15] Forecast a time series with ARIMA in Python. datascience.ibm.com. Available at: <https://datascience.ibm.com/exchange/public/entry/view/815137c868b916821dec777bdc23013c>. Accessed: 1  Aug.  2017.

[16] Building ARIMA models. PennState Eberly College of Science. Available at: <https://onlinecourses.science.psu.edu/stat510/node/25>. Accessed: 1  Aug.  2017.

[17] NIELSEN, Michael A. Neural Networks and Deep Learning. Determination Press, 2015.

[18] OLAH, CHISTOPHER. Understanding LSTM Networks. Available at: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Accessed: 1  Aug.  2017.

[19] HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. **Neural computation**, v. 9, n. 8, p. 1735-1780, 1997.

[20] GUO, Jiang. Backpropagation through time. **Unpubl. ms., Harbin Institute of Technology**, 2013.

[21] Keras Documentation. keras.io. Available at: <https://keras.io/getting-started/faq/#what-does-sample-batch-epoch-mean>. Accessed: 1  Aug.  2017.

[22] BABYAK, Michael A. What you see may not be what you get: a brief, nontechnical introduction to overfitting in regression-type models. **Psychosomatic medicine**, v. 66, n. 3, p. 411-421, 2004.

[23] HEATON, JEFF. Introduction to Neural Networks for Java. 2. ed. [s.l.]: Heaton Research, Inc, 2008.

[24] TUKEY, JOHN W. Exploratory data analysis. Reading, Mass. [u.a.]: Addison-Wesley, [s.d.].

[25] HALLS-MOORE, MICHAEL. White Noise and Random Walks in Time Series Analysis. Quantstart.com. Available at: <https://www.quantstart.com/articles/White-Noise-and-Random-Walks-in-Time-Series-Analysis>. Accessed: 1  Aug.  2017.

1. *The statistical significance is represented by the y = 1.96 line which, by the Central Limit Theorem, is the number used for 95% confidence intervals.* [↑](#footnote-ref-1)
2. *A different ACF layout was used to highlight the statistical non-significant area* [↑](#footnote-ref-2)
3. *For ARIMA, this error may also be referenced as Estimated White Noise Standard Deviation.* [↑](#footnote-ref-3)
4. *\*It is a way to solve the vanishing gradient problem common in regular Recurrent Neural Networks[19].* [↑](#footnote-ref-4)