Implementation of Neural Networks in Time Series to Generate a Portfolio of Investment in Cryptocurrencies

José B. Hernández C.1,2, Jorge A. Flores S.3, and Jesús Láres4
1. Escuela de Matemáticas, Universidad Central de Venezuela, Los Chaguaramos, 1020, Caracas, Venezuela
2. Centro de Investigación en Matemáticas, A.C., Unidad Monterrey. Av. Alianza Centro 502, PIIT 66628, Apodaca, Nuevo León, México
3. EY Wavespace, Madrid, Spain
4. Parallel and Distributed Computing Center, Universidad Central de Venezuela, Los Chaguaramos, 1020, Caracas, Venezuela

Abstract: The present work aims to implement two types of neural networks and an analysis of a multivariate time series model of VAR type to predict the price of cryptocurrencies like Bitcoin, Dash, Ethereum, Litecoin, and Ripple. This subject has been popular in recent years due to the rapid price fluctuations and the immense amount of money involved in the cryptocurrencies market. Several technologies have been developed around cryptocurrencies, with Blockchain rising as the most popular. Blockchain has been implementing other information technology projects which have helped to open a wide variety of job positions in some industries. A “New Economy” is emerging and it is important to study its basis in order to establish the pillars that help us to understand its behavior and be ready for a new era.

Keywords: Neural networks, time series, cryptocurrencies, VAR models.

1. Introduction

At present, in the market of cryptocurrencies there are about 1575 coins, which are divided into two groups: Minable and Non-minable, the first ones being characterized by to have a finite amount of them and in the second group their amount may increase over time. The financial sector is currently undergoing great advances and at the same time transformations due to the demands of consumers. According to this and the trend to adapter of the new technologies into context the cryptocurrencies as a financial medium where the consumer uses the digital currency under the same sense of the physical currency but all through the Internet, where unlimited transactions and transfers are allowed [6].

Due to the fact that today is becoming more and more known the term cryptocurrency and at the same time the interest of many individuals have increased exponentially in this regard, it seems a pertinent idea to know the behavior of these to encourage those who still feel that it may be a risk to invest in these digital assets. According to this, one method of predicting behavior is through a time series. Given the behavior and the huge amount of data generated by cryptocurrencies, these can be a case study to analyze in the area of Data Science. Each time series will allow to perform the respective analysis, developing different methodologies in addition to the classical time series models. One of these methodologies is artificial neural networks.

Artificial neural networks are a family of algorithms that have formed the basis for the recent resurgence in the computational field called Deep Learning. The first work on neural networks began in the 50s and 60s of the 20th century. Recently, Deep Learning has experienced a resurgence of interest as it has achieved impressive cutting-edge results in specific tasks.
ranging from the classification of objects into images to fast and accurate automatic translation.

In this work we will use 5 types of cryptocurrencies that will be described in section 3. First we will adjust two types of neural networks, which will allow us to observe their behavior and predict their trend. We will also use VAR models to study the behavior of the time series of cryptocurrencies.

The rest of the article is organized as follows. In next section we will give a brief description of the MLP (Multilayer Perceptron) and LSTM (Long Short Term Memory) neural networks and also we will describe the VAR models (Autoregressive Vectors). In section 3 we will make the analysis of the cryptocurrencies and finally we will present the conclusions.

2. Preliminaries

2.1 Artificial Neural Networks

A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects: [1]

1) Knowledge is acquired by the network from its environment through a learning process.
2) Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

There are several models of neural networks, here we will describe two of them: MLP (Multilayer Perceptron) and LSTM (Long Short Term Memory).

2.2 MLP Neural Networks

An MLP neural network consists of an input layer, an output layer, and one or more hidden layers. Information is always transmitted from the input layer to the output layer. Perceptrons are the basis of MLP neural networks. A perceptron takes several binary inputs and produces a single binary output. Rosenblatt [5] proposed a rule to compute the output. He introduced weight, which are real numbers that express the importance of the respective input to the output. The output of the neural network is determined as a function of whether the weighted sum of these weights is greater or less than a certain parameter of the neural network, i.e.

\[
\text{output} = \begin{cases} 
0, & \text{if } \sum_j w_j x_j \leq A, \\
1, & \text{if } \sum_j w_j x_j > A.
\end{cases}
\] (1)

where \(x_i\) represents the inputs, \(w\) represents the weights and \(A\) the parameter defined in the network. Perceptrons are used for decision making, where the value that has the most weight is taken into account to produce the output. By varying the weights and the network parameter, different decision making models are available.

The process in an MLP neural network is as follows: First, the inputs are taken, these enter into the perceptrons, producing an output that passes to the hidden layers receiving as net input the following

\[
\text{net}_j = \sum_{i=1}^{N_j} w_{ij} x_i + \theta_j,
\] (2)

where \(\theta_j\) is the threshold of the neuron.

Neurons in the occult layer apply nonlinear functions, known as activation functions, in order to transform signals. The result is as follows

\[
b_j = f(\text{net}_j),
\] (3)

where \(f\) is the activation function, \(\text{net}_j\) is the entry to the hidden layer and \(b_j\) is the value of the neuron output \(j\).

This value is transferred through the weights \(v_{kj}\) to the output layer

\[
\text{net}_k = \sum_{j=1}^{L_j} v_{kj} b_j + \theta_j.
\] (4)

In the output layer applies the same operation as in the previous layer, the neurons of this last layer provide the output \(y_k\) of the network that is given by

\[
y_k = f(\text{net}_k).
\] (5)

Then begins a stage of learning or training of the neural network, the objective is to minimize the error between the output obtained by the network and the desired output.

The error function, \(E_p\), for each \(p\) pattern, is defined by

\[
E_p = \frac{1}{2} \sum_{k=1}^{H} (d_{pk} - y_{pk})^2,
\] (6)
where $d_{pk}$ is the expected output for the output neuron $k$ and $y_{pk}$ is the output provided by the network to the presentation of the p pattern. As $E_p$ is a function of all the weights in the network, the $E_p$ gradient is a vector equal to the partial derivative of $E_p$ with respect to each of its weights. The gradient takes the direction that determines the fastest increase in the error, while the opposite direction determines the fastest decrease in the error. Therefore, the error can be reduced by adjusting each weight in the direction of the vector and can be expressed as follows

$$-\sum_{p=1}^{P} \frac{\partial E_p}{\partial W_{ij}}.$$  

2.3 LSTM Neural Networks

Recurrent Neuronal Networks (RNN) are networks with cycles that allow information to persist or connect to each other. These cycles allow information to advance from the previous step to the next step within the network. LSTMs (Long Short Term Memory) are a special type of RNN and are capable of learning long-term dependencies. These networks were introduced by Hochreiter and Schmidhuber [2].

An LSTM neural network is a network consisting of memory blocks and memory cells together with the gate units that contain them, that is, each LSTM unit is composed of a cell and three gates (input, output, forgetting). This structure allows the LSTM network to select which information is remembered and which is forgotten. Multiplicative input gate units are used to avoid negative effects that unrelated inputs can create. The input flow to the memory cell is controlled by the input gate, while the output gate controls the output sequence from the memory cell to other blocks of the LSTM network.

The oblivion gate in the memory block structure is controlled by a single-layer neural network. In an instant of time $t$, the components of the LSTM network are updated using the following equation

$$f_t = \sigma(W[x_t, h_{t-1}, C_{t-1}] + b_f),$$  

where $x_t$ is the input sequence, $h_{t-1}$ is the output of the previous block, $C_{t-1}$ is the memory of the previous LSTM block and $b_f$ is a polarization vector. $\sigma$ is the logistic sigmoid function and $W$ represents weight vectors for each input separately. The $\sigma$ function is applied to the previous memory block by multiplying elements and generates a number between 0 and 1 for each number in the state of the $C_{t-1}$ block, where 1 represents “keep this completely” and 0 represents “forget this completely”. The entrance gate is a section of the LSTM block where the new memory is created by a simple network with tanh activation function and the previous block. These operations are calculated using the following equations:

$$i_t = \sigma(W[x_t, h_{t-1}, C_{t-1}] + b_i)$$  

$$C_t = f_tC_{t-1} + i_t \tanh(W[x_t, h_{t-1}, C_{t-1}] + b_f)$$  

Finally, the probabilities of the current LSTM block $C_t$ are generated. To do this, first a layer is executed which contains an activation function in charge of deciding which parts of the state of the block will generate the output. Then, the state of the block is placed through the function tanh in order to obtain values between -1 and 1, later this value is multiplied by the output of the activation function. The outputs are calculated by means of

$$o_t = \sigma(W[x_t, h_{t-1}, C_t] + b_o)$$  

$$h_t = \tanh(C_t) o_t$$  

2.4 VAR Models

A time series $y_t$ follows a VAR (Autoregressive Vector) model of order $p$, VAR($p$), if

$$y_t = \sum_{i=1}^{p} A_i y_{t-i} + u_t,$$  

where $A_i$ are matrices of order $K \times K$, with $K$ the number of endogenous variables of the problem, for $i=1, ..., p$ and $u_t$ a process $K$-dimensional with $E(u_t)=0$ and invariant positive defined covariance matrix in time $E(u_t \cdot u_t^T) = \Sigma u$ (white noise) [4].

An important feature of a VAR($p$) process is its stability. This means that it generates a stationary time series with mean, variances and invariant covariance structure over time, given the appropriate initial values.
The above statement can be verified by evaluating the characteristic polynomial

\[ \text{det}(l_k - A_1 z - \cdots - A_p z^p) \neq 0, \text{for } |z| \leq 1 \]

If the solution to the above equation has a root for \( z=1 \), then some or all of the variables in the VAR\( (p) \) process are integrated of order one, i.e., \( I(1) \). In practice, the stability of an empirical process \( \text{VAR}(p) \) can be analyzed considering the calculation of the auto-values of the coefficient matrix. A \( \text{VAR}(p) \) process can be written as a \( \text{VAR}(1) \) process as follows

\[ \xi = A \xi_{t-1} + v_t, \]

where

\[ \xi = \begin{bmatrix} y_t \\ y_{t-p+1} \end{bmatrix}, \quad A = \begin{bmatrix} A_1 & A_2 & \cdots & A_{p-1} & A_p \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix}, \quad v_t = \begin{bmatrix} u_t \\ \vdots \\ 0 \end{bmatrix}, \]

so the dimensions of the vectors \( \xi_t \) and \( v_t \) are \((K_p-1)\) and the dimension of the array \( A \) is \((K_p\times K_p)\). If the module of the \( A \) eigenvalues are smaller than one, then the process \( \text{VAR}(p) \) is stable.

For a given sample of the endogenous variables \( y_1, \ldots, y_t \) and enough pre-sampling values \( y_{p+1}, \ldots, y_0 \), the coefficients of a \( \text{VAR}(p) \) process can be estimated efficiently by means of least squares applied separately to each of the equations. Finally, predictions for \( y \) from an empirical process \( \text{VAR}(p) \) can be generated recursively according to

\[ y_{T+h|T} = A_1 y_{T+h+1|T} + \cdots + A_p y_{T+h-p+1|T}, \quad (13) \]

where \( y_{T+j|T} = y_{T+j} \text{ for } j \leq 0 \). The forecast error covariance matrix is given as

\[ \text{Cov} \left( \begin{bmatrix} y_{T+1|T} \\ \vdots \\ y_{T+h|T} \end{bmatrix} \right) = \begin{bmatrix} I & 0 & \cdots & 0 \\ \phi_1 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{h-1} & \phi_{h-2} \cdots & I \end{bmatrix} \sum_n \begin{bmatrix} I & 0 & \cdots & 0 \\ \phi_1 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{h-1} & \phi_{h-2} \cdots & I \end{bmatrix}, \quad (14) \]

and the matrices \( \phi \) are the matrices of empirical coefficients of the representation of the Wold moving average of a stable process \( \text{VAR}(p) \) given by

\[ y_t = \phi_0 u_t + \phi_1 u_{t-1} + \phi_2 u_{t-2} + \cdots, \quad (15) \]

where \( \phi_0 = k \) and \( \phi_s \) can be calculated recursively according to

\[ \phi_s = \sum_{j=1}^{s} \phi_{s-j} A_j \text{ parameters \( = 1, 2, \ldots, \)}. \]

where \( A_j = 0 \) for \( j > p \). The \( \otimes \) operator in Eq. (14) is the Kronecker product, which is defined as follows:

**Definition 2.1** Given \( A = (a_{ij}) \in K^{m \times n} \) and \( B = (b) \in K^{p \times q} \), the Kronecker product is defined as

\[ A \otimes B = (a_{ij}B)_{ij} \]

where each block \( (a_{ij}B) \) is in \( K^{p \times q} \) and \( A \otimes B \in K^{pn \times mq} \). This product is made between matrices of any order and is not commutative.

### 3. Results and Discussion

The following is an analysis and implementation of the three models described above. For this we selected five currencies, which were: Bitcoin, Dash, Ethereum, Litecoin and Ripple. The data was obtained via web Scraping via the Google Chrome extension and extracted from the Coinmarketcap website (https://coinmarketcap.com/currencies/). The Fig. 1 presents a sample of the data obtained. The study period is from January 1, 2016 to March 11, 2018. The study was performed with the software R and Python.

We start by adjusting the MLP neural network to each coin in order to predict whether the coin price decreases or increases the next day. The Table 1 shows the model parameter values for each coin. As can be seen in the Table 1, high accuracy was achieved for almost all coins. For Bitcoin 91%, for Dash 87%, for Ethereum 90%, for Litecoin 88%, and the lowest was for Ripple 78%, which indicates positive results, since there is a high probability that the models will classify correctly, remembering that it is a binary classification, where “0” indicates that the price the next day decreases and “1” indicates that the price of the coin increases for the next day.

In the Table 2 we can observe the measurement of each model by classes, that is, “0” and “1”. This was done based on the confusion matrix of each model.
We can notice that in general the measurement of the models is positive obtaining values above 85% in almost all cases for “precision”, “recall”, and for “F1-score” which is the average of “precision” and “recall”.

**Table 1** MLP Neural Network parameters for each coin.

| Currencies | Hidden layers | Perceptron       | α     | Activ. func | Accuracy |
|------------|---------------|------------------|-------|-------------|----------|
| Bitcoin    | 1 2 3         | 1000 100 10      | 0.01  | logistics   | 0.91     |
| Dash       | 1 2 3         | 1000 100 10      | 0.001 | tanh        | 0.87     |
| Ethereum   | 1 2 3         | 100 100 100      | 0.1   | tanh        | 0.90     |
| Litecoin   | 1 2 3         | 100 100 100      | 0.001 | tanh        | 0.88     |
| Ripple     | 1 2 3         | 100 10 10        | 0.0001| tanh        | 0.78     |

We then proceed to adjust the LSTM neural network to forecast coin price values at specific time periods. The “close” variable was taken for each coin. The forecast period is 10 days for each coin. To make the adjustment, due to the high variability of the series, the “close” series were normalized. In the Table 3 the parameters and values obtained to adjust the LSTM neural network to each coin are shown. Columns 2 to 4 present the parameter values: Number of layers and neurons per layer, the activation function and the instance number for training (Epoch). Columns 5 and 6 show the loss function values for training and validation. In the graphs you can see the prediction every 10 days given by the network for the last 4 months (dashed lines) as well as the time series of closing prices (continuous line). We can notice in all cases even if the values are not exactly the same as the original series, the behavior in the same if it adjusts to the closing price, i.e., when the closing price increases the estimated by the network and the same for when the price decreases.
Finally we proceed to adjust the VAR models. For this we use the variables Open, High and Low as regression variables and as a Close dependent variable. The first step was to apply the function log to reduce the variability of the variables. The second step was to perform the Dickey-Fuller test to see if there is cointegration or not in the variables. In the Tables 5-9 you can see the results of them, as we can see all the variables for each coin are cointegrated of order 1. Once the cointegration tests are done, we proceed to calculate the optimal p for each coin, to finally adjust the model and make the prediction for 10 days. In the Table 4 are shown the p orders for each coin with the respective values of the Akaike information criterion (AIC), Hannan and Quinn information criterion (HQ), Schwarz Bayesian information criterion (SC) and the Akaike final prediction error (FPE). It should be noted that according to Lütkepohl [3] SC and HQ provide consistent estimates of the true order of the p delay, while AIC and FPE overestimate the order of the delay with positive probability. In the table we see that for Ethereum, Litecoin and Ripple, the AIC and FPE criteria indicate lag orders of 4, 6 and 17.
Table 4  Values of $p$ for the VAR model for each currency.

| Currency | AIC | HQ | SC | FPE |
|----------|-----|----|----|-----|
| Bitcoin  | 1   | 1  | 1  | 1   |
| Dash     | 2   | 1  | 1  | 2   |
| Ethereum | 4   | 1  | 1  | 4   |
| Litecoin | 6   | 1  | 1  | 6   |
| Ripple   | 17  | 2  | 1  | 17  |

respectively, while the SC and HQ criteria indicate $p = 1$ lag for almost all currencies, except for Ripple where the HQ indicates a $p = 2$ lag.

In the Fig. 3 we can see the predictions made with the VAR models for each crypto currency. In the upper left we see the forecast for Bitcoin, as we notice, it adjusts to the increase in the price of the currency observed in the previous week. The same can be said for Ethereum (middle left) and for Litecoin (middle right). On the other hand, for the Dash (top right) the prediction indicates a price increase even though previous records show a drop in the price of the crypto currency. At the bottom shows the forecast for the Ripple, there we can see that the forecast of the price of the crypto currency is down, following the general behavior observed in the previous week. All forecasts are to 10 days with a 95% confidence level.

Fig. 2  Closing price predictions with LSTM neural network for each coin: Bitcoin (top left), Dash (top right), Ethereum (middle left), Litecoin (middle right) and Ripple (bottom).
Implementation of Neural Networks in Time Series to Generate a Portfolio of Investment in Cryptocurrencies

Fig. 3  Closing price predictions with VAR models for each coin. Bitcoin (top left), Dash (top right), Ethereum (middle left), Litecoin (middle right) and Ripple (bottom).

Table 5  Dickey-Fuller Test for Bitcoin.

| Variable | Deterministic Term | Lags | Statistical | Critical Values |
|----------|--------------------|------|-------------|-----------------|
|          |                    |      |             | 1%  | 5%  | 10% |
| Open     | Constant, trend    | 1    | -2.227      | -3.96 | -3.41 | -3.12 |
| Δ Open   | Constant           | 1    | -19.691     | -3.43 | -2.86 | -2.57 |
| High     | Constant, trend    | 2    | -2.157      | -3.96 | -3.41 | -3.12 |
| Δ High   | Constant           | 1    | -19.096     | -3.43 | -2.86 | -2.57 |
| Low      | Constant, trend    | 2    | -2.316      | -3.96 | -3.41 | -3.12 |
| Δ Low    | Constant           | 1    | -20.068     | -3.43 | -2.86 | -2.57 |
| Close    | Constant, trend    | 1    | -2.230      | -3.96 | -3.41 | -3.12 |
| Δ Close  | Constant           | 1    | -19.871     | -3.43 | -2.86 | -2.57 |

4. Conclusions

With regard to the adjustment of the MLP neural network for each crypto currency we can note that good results were obtained in general. Taking into account this model we can mention that at the time of investing in crypto currencies the Bitcoin would be the first to consider, in fact, is the most quoted in the
Implementation of Neural Networks in Time Series to Generate a Portfolio of Investment in Cryptocurrencies

market. Then we would consider Ethereum, Litecoin and Dash. With this MLP model we would not consider the Ripple as the results were not optimal.

Table 6  Dickey-fuller test for dash.

| Variable | Deterministic Term | Lags | Statistical | Critical Values |
|----------|--------------------|------|-------------|----------------|
|          |                    |      | 1%          | 5%             | 10%            |
| Open     | Constant, trend    | 1    | -1.796      | -3.96          | -3.41          | -3.12          |
| Δ Open   | Constant           | 1    | -21.069     | -3.43          | -2.86          | -2.57          |
| High     | Constant, trend    | 1    | -1.897      | -3.96          | -3.41          | -3.12          |
| Δ High   | Constant           | 1    | -21.169     | -3.43          | -2.86          | -2.57          |
| Low      | Constant, trend    | 1    | -1.853      | -3.96          | -3.41          | -3.12          |
| Δ Low    | Constant           | 1    | -20.793     | -3.43          | -2.86          | -2.57          |
| Close    | Constant, trend    | 1    | -1.756      | -3.96          | -3.41          | -3.12          |
| Δ Close  | Constant           | 1    | -21.314     | -3.43          | -2.86          | -2.57          |

Table 7  Dickey-fuller test for Ethereum.

| Variable | Deterministic Term | Lags | Statistical | Critical Values |
|----------|--------------------|------|-------------|----------------|
|          |                    |      | 1%          | 5%             | 10%            |
| Open     | Constant, trend    | 1    | -1.706      | -3.96          | -3.41          | -3.12          |
| Δ Open   | Constant           | 2    | -14.952     | -3.43          | -2.86          | -2.57          |
| High     | Constant, trend    | 1    | -1.749      | -3.96          | -3.41          | -3.12          |
| Δ High   | Constant           | 1    | -17.799     | -3.43          | -2.86          | -2.57          |
| Low      | Constant, trend    | 2    | -1.725      | -3.96          | -3.41          | -3.12          |
| Δ Low    | Constant           | 1    | -20.447     | -3.43          | -2.86          | -2.57          |
| Close    | Constant, trend    | 1    | -1.691      | -3.96          | -3.41          | -3.12          |
| Δ Close  | Constant           | 2    | -14.932     | -3.43          | -2.86          | -2.57          |

Table 8  Dickey-fuller test for Litecoin.

| Variable | Deterministic Term | Lags | Statistical | Critical Values |
|----------|--------------------|------|-------------|----------------|
|          |                    |      | 1%          | 5%             | 10%            |
| Open     | Constant, trend    | 1    | -1.840      | -3.96          | -3.41          | -3.12          |
| Δ Open   | Constant           | 1    | -20.417     | -3.43          | -2.86          | -2.57          |
| High     | Constant, trend    | 1    | -1.804      | -3.96          | -3.41          | -3.12          |
| Δ High   | Constant           | 1    | -19.713     | -3.43          | -2.86          | -2.57          |
| Low      | Constant, trend    | 1    | -1.832      | -3.96          | -3.41          | -3.12          |
| Δ Low    | Constant           | 2    | -18.174     | -3.43          | -2.86          | -2.57          |
| Close    | Constant, trend    | 1    | -1.824      | -3.96          | -3.41          | -3.12          |
| Δ Close  | Constant           | 1    | -20.287     | -3.43          | -2.86          | -2.57          |

Table 9  Dickey-fuller test for ripple.

| Variable | Deterministic Term | Lags | Statistical | Critical Values |
|----------|--------------------|------|-------------|----------------|
|          |                    |      | 1%          | 5%             | 10%            |
| Open     | Constant, trend    | 2    | -1.750      | -3.96          | -3.41          | -3.12          |
| Δ Open   | Constant           | 2    | -13.785     | -3.43          | -2.86          | -2.57          |
| High     | Constant, trend    | 1    | -1.801      | -3.96          | -3.41          | -3.12          |
| Δ High   | Constant           | 2    | -14.720     | -3.43          | -2.86          | -2.57          |
| Low      | Constant, trend    | 2    | -1.705      | -3.96          | -3.41          | -3.12          |
| Δ Low    | Constant           | 1    | -17.279     | -3.43          | -2.86          | -2.57          |
| Close    | Constant, trend    | 2    | -1.753      | -3.96          | -3.41          | -3.12          |
| Δ Close  | Constant           | 2    | -13.766     | -3.43          | -2.86          | -2.57          |
Implementation of Neural Networks in Time Series to Generate a Portfolio of Investment in Cryptocurrencies

Fig. 4 Diagram of fit and residuals for Close variable of Bitcoin from VAR Model. Fit value of Close (top) residuals (middle), ACF of residuals (bottom left) and PACF of residuals (bottom right).

Fig. 5 Diagram of fit and residuals for Close variable of Dash from VAR Model. Fit value of Close (top) residuals (middle), ACF of residuals (bottom left) and PACF of residuals (bottom right).

However, for the LSTM neural network, the desired results were not achieved. One of the factors that influenced this process was that we worked with a low amount of data, this is due to the fact that an equal period of time was used for the five currencies where their data did not have missing values.

Finally, positive results were obtained for the VAR time series model. In this case when reversing the order of choice would be Bitcoin, Litecoin and Ethereum. For the Dash, the model is not conclusive.
and for the Ripple shows low price which would make us lose our investment. With the VAR model we can notice that in four of the five currencies the prediction of the model was that the price would increase, making a contrast with the other models and with the real value this did not happen in the same way.

It is important to point out that this time the VAR model was not designed to predict irregularities or cover external factors that may occur in the cryptocurrency market. As future recommendations, it is proposed to train neural network models with large amounts of data in order to obtain better results. On the other hand, to implement other models of series of time involved in the area of the econometrics as the models ARIMA or GARCH.

Fig. 6  Diagram of fit and residuals for Close variable of Ethereum from VAR Model. Fit value of Close (top) residuals (middle), ACF of residuals (bottom left) and PACF of residuals (bottom right).

Fig. 7  Diagram of fit and residuals for Close variable of Litecoin from VAR Model. Fit value of Close (top) residuals (middle), ACF of residuals (bottom left) and PACF of residuals (bottom right).
Implementation of Neural Networks in Time Series to Generate a Portfolio of Investment in Cryptocurrencies

Fig. 8  Diagram of fit and residuals for Close variable of Ripple from VAR Model. Fit value of Close (top) residuals (middle), ACF of residuals (bottom left) and PACF of residuals (bottom right).

References

[1] Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation* (2nd ed.). Pearson Prentice Hall, Patparganj, India, pp. 23-71.

[2] Hochreiter, S., and Schmidhuber, J. (1997). “Long Short-Term Memory.” *Neural Computation* 9 (8): 1735-1780.

[3] Lütkepohl H. (2005). *New Introduction to Multiple Time Series Analysis*, Springer-Verlag, Berlin, pp. 69-192.

[4] Paff, B. (2008). “VAR, SVAR and SVEC Models: Implementation within R Package Vars.” *Journal of Statistical Software, Foundation for Open Access Statistics* 27 (4): 1-32.

[5] R. Rosenblatt (1958). “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” *Psychological Review* 65 (6): 386-408.

[6] Sánchez A., and Terán, O. E. (2018). “Criptomonedas como oportunidad de negocio de microempresas del sector turístico en la zona sur oriente del estado de México.” *Revista global de negocios* 6 (1): 93-104.