Cryptocurrency Volatility Index: An Efficient Way to Predict the Future CVI

An Pham Ngoc Nguyen[1,2], Martin Crane[1,3], and Marija Bezbradica[1,3]

[1] School of Computing, Dublin City University, Ireland
[2] SFI Centre for Research Training in Artificial Intelligence, Ireland
[3] ADAPT Center for Digital Content Technology, Ireland

Abstract. The Cryptocurrency Volatility Index (CVI index) has been introduced to estimate the 30-day future volatility of the cryptocurrency market. In this article, we introduce a new Deep Neural Network with an attention mechanism to forecast future values of this index. We then look at the stability and performance of our proposed model against the benchmark models widely used for time series prediction. The results show that our proposed model performs well when compared to popular methods such as traditional Long Short Term Memory, Temporal Convolution Network, and other statistical methods like Simple Moving Average, Random Forest and Support Vector Regression. Furthermore, we show that the well-known Simple Moving Average method, while it has its own advantages, has the weak spot when dealing with time series with large fluctuations.

Keywords: cryptocurrencies · volatility · CVI · LSTM · attention mechanism

1 Introduction

The success of the cryptocurrency market can be seen through the constant increase in total market capitalization, going from 20 billion USD at the beginning of 2017 to over 3 trillion USD in 2021, and in the number of investors joining the community starting from 65 million in mid-2020 to over 300 million at the end of 2021. One reason for this expansion is due to the large gains that cryptocurrencies (especially Bitcoin) can bring to investors, thanks to dramatic fluctuations in prices [13]. Furthermore, unlike the stock market, the cryptocurrency market has much fewer restrictions, allowing investors to complete a transaction quickly and freely [37]. Unfortunately, the ease of trading in cryptocurrency makes it very vulnerable to external factors such as news of financial developments, the movement of other assets and even the statements from influencers [18]. As a consequence, compared to traditional markets such as stocks, bonds and commodities, the volatility of the cryptocurrency market tends to be extremely high [38]. This can lead to large fluctuations, for instance, a study conducted by Alexander et al. revealed that the losses of cryptocurrencies can reach 70 percent within one day [10]. Thus, the understanding of the volatility of the cryptocurrency

market is essential to reduce the investment risk as well as to open opportunities to predict market's movements and gain profits.

Due to the demand to measure the volatility of the cryptocurrency market, the Cryptocurrency Volatility Index (CVI index) has been launched. This is defined as a measure of the 30-day future fluctuation degree of the price of the entire cryptocurrency market using the Black-Scholes option pricing model. In this way, an index that fluctuates between 0 and 200 is developed, such that 200 will indicate the maximum level of implied volatility in the market whilst a value of zero indicates the lowest volatility [11]. This index is intended to prevent investors from putting themselves at risk by modifying their trading strategy in line with different values of CVI. The higher the CVI value is, the greater the risks are but also the greater the potential return is.

The usefulness of the CVI index is the main motivation behind this work. Specifically, our objective is to answer the following research question:

– Is it possible to predict with accuracy the future value of the CVI index using neural networks?

We examine data from 10 cryptocurrencies which act as input features. Furthermore, we use the CVI index time series as labels for our prediction task. All time series span from 10/11/2017 to 12/01/2022. The details of this dataset are described in section 3. Regarding the prediction model, we utilize Long Short Term Memory (LSTM), which is suitable for time series-related problems [17,19], to extract these input features, followed by a Multilayer Perceptron (MLP) which allows the synthesis of the information extracted from the previous stage, the outputs of this stage being the predicted CVI values. Moreover, instead of using only the output from the last LSTM cell, which might lead to the losses of valuable information from earlier input data points, we use all outputs from the entire sequence of LSTM cells that are weighted according to their level of importance. We use a technique named *Attention mechanism* [40] for this weighing and name this novel neural network method AT-LSTM-MLP.

The remainder of this paper is organized as follows. Section 2 discusses related works. Section 3 introduces the dataset used in our experiments. Section 4 describes our AT-LSTM-MLP model and an outline of appropriate hyperparameters and specifications required to run experiments. Section 5 shows empirical results followed by an analysis learned from experiments. Section 6 gives conclusions for the article.

2 Related Works

Throughout the history of financial markets, there have been a number of prediction methods used to predict the future price and implied volatility of different assets such as stocks, bonds and cryptocurrencies.

Simple methods based on statistical learning frameworks have been found to show good performance in many studies, e.g. Simple Moving Average (SMA) [2], Support Vector Regression (SVR) [12] and Random Forest (RF) [31]. The

advantages of these statistical methods are that they are easy to implement, thus, the time complexity is significantly low and they tend to work well with different datasets. That is why these methods are often used as benchmark models to verify the efficiency of other methods even though there is a superiority from more recent methods [24,30].

Another commonly used method is GARCH, typically used to estimate the volatility of a time series such as stocks, bonds, market indices and recently, cryptocurrencies [8]. The first to use the GARCH model was Abdelhamid to estimate the volatility of different stocks circulating in the stock exchange of Casablanca at that time [14]. With time, there have been many variants of GARCH models proposed to optimize the prediction problem corresponding to a specific time series [5,21]. Until now, although the global financial market has changed considerably(i.e. more people invested, more investment options, etc), the performance of GARCH-type models still seems to be good. The study [29] showed that the volatility of world currencies, namely the GBP, CAD, AUD, CHF and the JPY is effectively predicted by using a GARCH-type model called IGARCH. On the other hand, CGARCH and TGARCH models work well with major cryptocurrencies such as Bitcoin, Litecoin and Ripple. All the time series were observed from October 2015 to November 2019. For a longer period from 2010 to 2020, GARCH and a simpler corresponding model ARCH have also been successfully used to form variance equations for highly-capitalized cryptocurrencies [6].

Neural Networks were first introduced in 1944 by McCullough and Pitts from the University of Chicago [26]. Since then, they have been applied to many different areas from healthcare services [16] to daily utility applications [41] and entertainment purposes [39]. In Finance, there is an increasing tendency to use RNNs to understand the operation of financial markets [1]. This is because they tend to work very well with time series data and are often integrated into prediction-related problems [19,34]. LSTM [15], a version of Recurrent Neural Networks, appears to be dominant because of its ability to recollect longer so that information in the architecture can travel deeper. In [28], Aditi et al. made a comparison between Polynomial Regression, RNN, LSTM and ARIMA [7] for Bitcoin price fluctuation prediction. To do the experiment, the authors used a dataset of a combination between daily close price, volume and social media-related information such as Google Trend index, tweets' emotions as well as the number of posts on Twitter containing the keyword "Bitcoin". Out of four methods, LSTM outperformed the others with an accuracy of approximately 83 percent. Elsewhere, the information about Bitcoin blockchain technology has also been exploited to predict the future price. Specifically, Suhwan et al. investigated the performance of different Deep Learning models for estimating Bitcoin prices one day ahead [20]. In this study, 30 features were collected with one of them being daily closing price of Bitcoin and the rest being about blockchain specifications such as the average block size, the relative measure of difficulty in finding a new block, the total number of blockchain wallets created, etc. The results showed that LSTM slightly outperformed Multi-layer perceptrons and a pre-trained model ResNet. Another common Neural Network architecture is

Convolution Neural Network (CNN) [4], widely applied to analyze image data. Recently, this type of Neural Network has been incorporated into financial research and has achieved certain successes. In particular, the authors in [3] used CNN as a part of their model to predict the future Bitcoin volatility and named it Temporal Convolution Network (TCN). Moreover, CNN can be incorporated into a LSTM model to improve the efficiency of the prediction. A method introduced in [25] for the prediction of Bitcoin, Ethereum and Ripple prices on the following hour combined three different Deep Learning models, including LSTM, Bidirectional LSTM and CNN, then used a statistical method such as Support Vector Regression or k-Nearest Neighbour to average the weighted output of each single model. It was confirmed that this practice helped improve the performance compared to individual models. In general, a large variety of methods for future volatility prediction exist in both traditional and cryptocurrency markets that use RNNs. All of these methods show an outperformance compared to GARCH-type models [24,17].

3 Dataset

In order to answer our research question, we have chosen 77 digital currencies that meet the following three criteria:

(i) Large market capitalization: The market capitalization is greater than 1 million USD.
(ii) Long historical time series: The number of trading days is greater than 1000.
(iii) Few missing values: The percentage of missing values is less than 10 percent.

Furthermore, we use the CVI Index, which we have introduced in the previous section acting as labels for our proposed prediction model. The dataset used in this work was collected from two different sources: for cryptocurrencies' historical data, we downloaded time series' closing prices on the open financial website Yahoo! Finance; for CVI Index, the data was collected from the website Investing.com. All time series are collected day by day.

However, the trading schedule of each cryptocurrency is different, resulting in inconsistency across the entire dataset. Due to the difference in trading dates, if we attempted to use all the historical time series, we might end up with an empty dataset. For this reason, we chose 10 out of 77 cryptocurrencies that are widely known and have the longest historical data, the ultimate input features comprise BTC, ETH , BCH, ADA, XRP, DOGE, LINK, LTC, XLM and ETC. Each of them has 1537 data points starting from 10/11/2017 until 12/1/2021. When it comes to our labels, CVI Index begins on 31/3/2019 and ends on the same day as the other time series, the total number of data points is 1019. Subsequently, after merging all available data together (10 cryptocurrency time series and CVI time series), we have a 1019 data point dataset that consists of 10 cryptocurrencies acting as input features and CVI index acting as labels.

Since the original input features were collected daily from closing prices, it is necessary to convert this type of data into daily volatility:
Step 1: Calculating daily return of each cryptocurrency by using daily closing prices: $y_t = log\left(\frac{Close_t}{Close_{t-1}}\right)$. Where y_t, $Close_t$ are daily return, close price at timestamp t, respectively.
Step 2: Calculating daily volatility from daily return, as suggested by Fernando et al [32]: $\sigma_t = \sqrt{\frac{1}{5}\Sigma_{i=0}^{4} y_{t-i}^2}$. Where σ_t is realized volatility at timestamp t.

4 Methodology

4.1 Attention Based-Deep Neural Network (AT-LSTM-MLP)

We create a novel Neural Network by combining 2 different types of Neural Network models, Long Short Term Memory [15] and Multilayer Perceptron (MLP) [23] and adding Attention Mechanism to weight the degree of importance for each LSTM cell. A diagram for this architecture is shown in Figure 1.

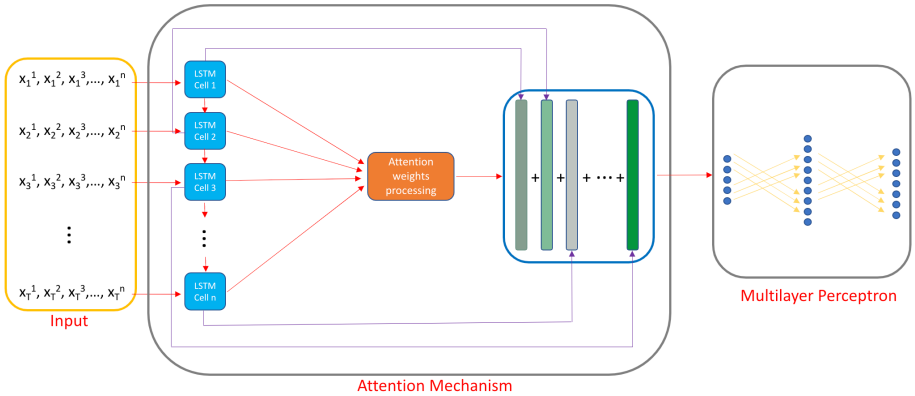

Fig. 1: AT-LSTM-MLP model. x_k refers to time series values at timestamp k, $k = (1, 2, \ldots, T)$, x^l refers to time series values of lth input feature, $l = (1, 2, \ldots, n)$. The model comprises 2 parts: Long Short Term Memory with Attention Mechanism and Multilayer Perceptron. The output of the model are future predicted CVI indices.

Our proposed architecture (AT-LSTM-MLP) takes advantage of the outstanding characteristics not only of LSTM but also Attention Mechanism by using the weighted outputs of all LSTM cells from the sequence. These LSTM outputs will go to a Multilayer Perceptron where the future CVI index is predicted at the end. The process can be described as follows:

Firstly, we determine the shape of input data, each row is one timestamp while each column is one input feature. In our defined formula, we set T as the number of timestamps and n as the number of input features. We use the matrix form below as an input of our model:

$$\begin{pmatrix} x_1^1 & x_1^2 & x_1^3 & \dots & x_1^n \\ x_2^1 & x_2^2 & x_2^3 & \dots & x_2^n \\ x_3^1 & x_3^2 & x_3^3 & \dots & x_3^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_T^1 & x_T^2 & x_T^3 & \dots & x_T^n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_T \end{pmatrix} \tag{1}$$

Secondly, we move this input forward to a sequence of LSTM cells where each single LSTM cell (f_t) takes one row of the input (x_t) sequentially with the number of LSTM cells equal to the number of rows. The output at this phase is a set of hidden states at each LSTM cell ($h_1, h_2, \dots, h_T$): $h_t = f_t\left(x_t, h_{t-1}\right)$. Since LSTM takes the previous hidden state as an input argument, we set as default that the original hidden state (h_o) is zero, meaning that there no information from the time series revealed to the first LSTM cell. The hidden states ($h_1, h_2, h_3, \dots, h_T$) then go through the self-attention mechanism. The output at this stage, denoted as s, is obtained by the following mathematical equations:

$$u_k = \tanh\left(W \times h_k\right) \tag{2a}$$

$$a_k = \frac{exp\left(score\left(u_k, u\right)\right)}{\Sigma_{t=1}^{T} exp\left(score\left(u_t, u\right)\right)} \tag{2b}$$

$$s = \Sigma_{i=1}^{T} a_i h_i \tag{2c}$$

where W and u are training parameters, $a_1, a_2, \dots, a_n$ are Softmax coefficients. The equation 2a can be interpreted as a representation of a Fully Connected layer once each hidden state is passed to this layer in order to be embedded in a new vector space. We define an alignment function named *score* to measure how important each embedded hidden state u_t is. Theoretically, *score* can be any function depending on the research questions. In this work, we choose a straightforward function $score\left(u_k, u\right) = u_k^T u$. This function will return a scalar value for each corresponding embedded hidden state. We use Softmax function to map the original values to a probability distribution where all the components add up to 1, the larger input components will correspond to larger probabilities. The last equation 2c is a weighted sum of all considered hidden states as well as the output of this attention mechanism.

Thirdly, the process continues moving to a Multilayer Perceptron with the predicted CVI index y_{CVI} obtained at the end: $y_{CVI} = MLP\left(s\right)$.

4.2 Training Parameters and Implementation of Our Proposed Model

Our aim is to predict one-day future CVI index using 10 input features, which are 10 cryptocurrency time series. We run our proposed model on Tesla K80

GPU with memory size of 12GB. We take the last 20 percent of the dataset for test set in order to evaluate the model's performance, while the remaining data is used for training. All trainable parameters we initialize randomly following normal distribution with mean of 0. We built the model and ran all mentioned methods using Pytorch [33]. The details for the model specifications are described in Table 1, the optimal value according to each specification is shown in bold. We note that Mean Absolute Error is chosen as the loss function for our model. This is because we got the lowest errors in the test set using this metric.

4.3 Training Parameters and Implementation of Benchmark Models

The performance of our new method is verified by comparing with the following five different techniques:

1) Simple Moving Average [36]

We run the Simple Moving Average method for various window sizes according to the Autocorrelation plot. From this, we noticed 30 spikes outside of the blue area and thus statistically significant. Since a small change in the window sizes could lead to a small difference among the results, we do not use all possible window sizes. Instead, we choose window sizes = 2, 3, 4,. . ., 10, 15, 20, 25 and 30.

2) Support Vector Regression [27]

We use the Radial Basis Function kernel (RBF kernel) to find the optimal hyperplane for this regression problem because our data is clearly non-linear [35]. The regularization parameter is set to 1.0 to achieve a lower generalization error. To capture the shape of the data efficiently, we choose Gamma coefficient $\gamma = 0.1$, which is a parameter attached to the RBF kernel.

3) Random Forest [22]

We tested different possible values for the number of Decision Trees and found that 100 gives the best result. At each internal node of a Decision tree, we randomly choose three input features to consider when looking for the best split. The optimal feature will be chosen based on Variance Reduction technique. The minimum number of samples required to split an internal node is 2, the minimum number of samples at one leaf is 1. We use the average operation to synthesize the results of all Decision Trees.

4) Standard Long Short Term Memory [15]

This model comprises 20 LSTM cells with the size of hidden state being 1. We use the hidden state of the last LSTM cell as the output of the model. We train this model in 7000 epochs with a learning rate of 0.005. The other parameters used in this model are similar to our original model AT-LSTM-MLP.

5) Temporal Convolution Network [3]

From our original model, we replace the LSTM block with a TCN block. As each input of this model has 10 features and 20 timestamps, we set a kernel with

a length of 5 and the number of channels of 10 to slide on the input. We choose dilation = 2. We train this model in 5000 epochs with a learning rate of 0,001. The other parameters used in this model are similar to AT-LSTM-MLP.

5 Empirical Results and Analysis

Table 1: Specifications used in AT-LSTM-MLP, the best results are shown in bold.

Specifications	Values/Methods
Loss function	**Mean Absolute Error (MAE)** Mean Squared Error (MSE) Root Mean Square Error (RMSE) Mean Absolute Percentage Error (MAPE) Symmetric Mean Absolute Percentage Error(SMAPE)
Optimizer	Stochastic Gradient Decent (SGD) Nesterov Accelerated Gradient (NAG) Adagrad AdaDelta Adam **RMSProp**
#Time stamps	5, 10, 15, **20**, 25, 30, 35
LSTM's hidden state size	10, 20, 30, ... , **100**, 110, 120
#Fully Connected layers	1, 2, **3**, 4, 5
1st Fully Connected size	16, 32, **64**, 128, 256
2nd Fully Connected size	16, **32**, 64, 128, 256
3rd Fully Connected size	1
Batch size	16, **32**, 64
#Epochs	5000, **7000**, 10000
Learning rate	0.0001, 0.001, **0.005**, 0.1

We measure the error of the methods through three different metrics: MAE, RMSE, and SMAPE [9]. For methods when results vary at every single run, we run each model 10 times, then get the mean μ and standard deviation σ from all single results to give the final figure that is represented as $\mu \pm \sigma$. For static methods, such as Simple Moving Average and Support Vector Regression, we run them once and consider the result as the final result.

Table 2 shows the results of our proposed model and four other models used for comparison. AT-LSTM-MLP, which is our model, outperforms other methods with RMSE, MAE, and SMAPE of 2.24 ± 0.17, 1.62 ± 0.09, 1.76 ± 0.10, respectively. LSTM comes into second place with errors at more than double comparing to AT-LSTM-MLP. Whereas, the three remaining methods show poor results as the predicted values are too far from real values. These results give an answer to our research question that AT-LSTM-MLP can predict the future value of the CVI index well.

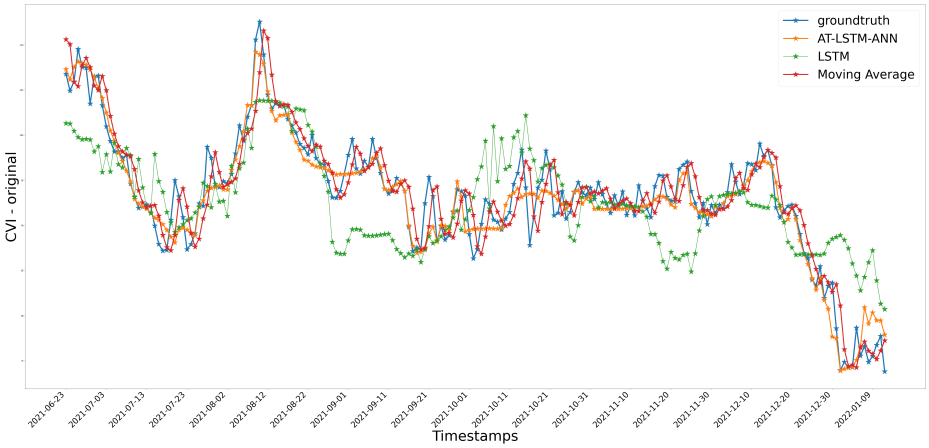

Fig. 2: Predicted values and real values, using our proposed method and four aforementioned benchmarks.

Table 2: Empirical results of 4 different methods measured in three metrics RMSE, MAE and SMAPE. The best results are shown in bold.

Models	RMSE	MAE	SMAPE
AT-LSTM-MLP	**2.24 ± 0.17**	**1.62 ± 0.09**	**1.76 ± 0.10**
LSTM	5.54 ± 0.85	4.11 ± 0.57	4.50 ± 0.64
SVR	12.69	10.42	11.84
RF	14.25 ± 0.17	12.20 ± 0.14	13.69 ± 0.36
TCN	9.20 ± 0.13	7.35 ± 0.11	8.10 ± 0.12

Table 3: Empirical results of Simple Moving Average with different window sizes. The best results are shown in bold.

Window size	2	3	4	5	6	7	8	9	10	15	20	25	30
RMSE	**2.63**	2.89	3.04	3.13	3.19	3.31	3.46	3.63	3.83	4.59	5.20	5.68	6.13
MAE	**2.02**	2.27	2.42	2.49	2.54	2.64	2.77	2.92	3.07	3.57	3.92	4.29	4.69
SMAPE	**2.16**	2.43	2.60	2.67	2.72	2.82	2.95	3.12	3.28	3.82	4.22	4.62	5.02

Table 3 show the results of SMA with different sizes of sliding window. In general, we can conclude that SMA works well with the data. However, the accuracy seems to fall off gradually as the sliding window size increases, the first sliding window sizes perform better than all other benchmarks with the exception of our method (AT-LSTM-MLP). SMA with sliding window size of 2 yields the best result of 2.02, 2.63, and 2.16 in MAE, RMSE, and SMAPE, respectively. This performance indicates the weakness of the SMA method. Particularly, SMA only works well with stable data, i.e. when the difference between time stamps within a sliding time window is small.

An illustration of predicted and real values using AT-LSTM-MLP and benchmark models is shown in Figure 2 (only methods with good performance are illustrated).

6 Conclusions

We have built a novel Deep Neural Network by combining Long Short Term Memory, Multilayer Perceptron and Attention mechanism, named AT-LSTM-MLP. Our goal is to predict the day-ahead CVI index using ten long-series cryptocurrencies with large market capitalization. By comparing our proposed method with five benchmark models in three different metrics, we show that AT-LSTM-MLP performs better than the others. Moreover, we also indicate the main weakness of SMA: it only works well with stable data, when the difference between time stamps within a sliding time window is small.

The results from this study contribute to literature on the cryptocurrency market with some useful tools and information that aim to helping the investors in making decisions in investment. We believe that our method can be applied to other prediction tasks that involve time series because of its good performance.

Acknowledgements: The authors Martin Crane, Marija Bezbradica wish to acknowledge the support, in part, from the Science Foundation Ireland under Grant Agreement No. 13/RC/2106_P2 at the ADAPT SFI Research Centre at DCU. ADAPT, the SFI Research Centre for AI-Driven Digital Content Technology, is funded by the Science Foundation Ireland through the SFI Research Centres Programme. The author An P. N. Nguyen wishes to acknowledge the support from Dublin City University's Research Committee and research grants from Science Foundation Ireland Centre for Research Training in Artificial Intelligence under grant number 18/CRT/6223.

References

1. Mehreen Saeed: An introduction to recurrent neural networks and the math that powers them (2021)
2. Abu, Rosbi, Uzaki: Forecasting cryptocurrency price movement using moving average method: A case study of Bitcoin cash. J. Adv. Res. **7**(12), 609–614 (2019)

3. Akbiyik, E., Erkul, M., Kaempf, K., Vasiliauskaite, V., Antulov-Fantulin, N.: Ask "who", not "what": Bitcoin volatility forecasting with Twitter data. arXiv preprint arXiv:2110.14317 (2021)
4. Albawi, S., Mohammed, T.A., Al-Zawi, S.: Understanding of a convolutional neural network. In: 2017 Int. Conf. Eng. Technol. (ICET). pp. 1–6. IEEE (2017)
5. Ali, G.: Egarch, GJR-garch, Tgarch, Avgarch, Ngarch, Igarch and Aparch models for pathogens at marine recreational sites. J. Stat. Econom. Methods **2**(3), 57–73 (2013)
6. Almansour, B., Alshaterand, M., Almansour, A.: Performance of Arch and Garch models in forecasting cryptocurrency market volatility. Ind. Eng. Manag. Syst. **20**(2), 130–139 (2021)
7. Ariyo, A., Adewumi, A., Ayo, C.: Stock price prediction using the ARIMA model. In: 2014 UKSim-AMSS 16th Int. Conf. Comput. Modelling Simul. pp. 106–112. IEEE (2014)
8. Bhowmik, R., Wang, S.: Stock market volatility and return analysis: A systematic literature review. Entropy **22**(5) (2020)
9. Botchkarev, A.: Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. arXiv preprint arXiv:1809.03006 (2018)
10. Brauneis, A., Mestel, R.: Price discovery of cryptocurrencies: Bitcoin and beyond. Econ. Lett. **165**, 58–61 (2018)
11. Briefing, C.: What is the crypto volatility index (CVI)? (2021)
12. Dash, R.K., Nguyen, T., Sharma, A., Cengiz, K., Sharma, A.: Fine-tuned support vector regression model for stock predictions. Neural Comput. Appl. (2021)
13. Dong, F., Xu, Z., Zhang, Y.: Bubbly Bitcoin. Economic Theory pp. 1–43 (2021)
14. El Bouhadi, A.: Conditional volatility of most active shares of Casablanca stock exchange. Econometrica **50**(1) (1982)
15. Gravesi, A.: Long Short Term Memory. Supervised Sequence Labelling with Recurrent Neural Networks **385**, 37–45 (2012)
16. Han, C., Rundo, L., Murao, K., Noguchi, T.: MADGAN: unsupervised medical anomaly detection GAN using multiple adjacent brain mri slice reconstruction. BMC Bioinform. **22**(31) (2021)
17. Hu, Y., Ni, J., Wen, L.: A hybrid deep learning approach by integrating LSTM-ANN networks with Garch model for copper price volatility prediction. Physica A **557** (2020)
18. Huynh, T.: When Elon Musk changes his tone, does Bitcoin adjust its tune? Comput. Econ. (2022)
19. Jay, P., Kalariya, V., Parmar, P., Tanwar, S., Kumar, N., Alazab, M.: Stochastic neural networks for cryptocurrency price prediction. Blockchain-Enabled Trustworthy Systems **8** (2020)
20. Ji, S., Kim, J., Im, H.: A comparative study of Bitcoin price prediction using deep learning. Mathematics **7**(10), 898 (2019)
21. Katsiampa, P.: Volatility estimation for Bitcoin: A comparison of Garch models. Econ. Lett. **158**, 3–6 (2017)
22. Khaidem, L., Saha, S., Dey, S.R.: Predicting the direction of stock market prices using random forest. Appl. Math. Finance (2016)
23. Krogh, A.: What are artificial neural networks? Nat. Biotechnol. **26**(2), 195–197 (2008)
24. Liu, Y.: Novel volatility forecasting using deep learning–Long Short Term Memory Recurrent Neural Networks. Expert Syst. Appl. **132**, 99–109 (2019)

25. Livieris, I., Pintelas, E., Stavroyiannis, S., Pintelas, P.: Ensemble deep learning models for forecasting cryptocurrency time-series. Algorithms **13**(5) (2020)
26. McCulloch, W., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biol. **5**(4), 115–133 (1943)
27. Meesad, P., Rasel, R.I.: Predicting stock market price using support vector regression. 2013 Intl. Conf. Informatics, Electronics & Vision (ICIEV) pp. 1–6 (2013)
28. Mittal, A., Dhiman, V., Singh, A., Prakash, C.: Short-term Bitcoin price fluctuation prediction using social media and web search data. In: 2019 Twelfth International Conference on Contemporary Computing (IC3). pp. 1–6. IEEE (2019)
29. Naimy, Haddad, Fernández-Avilés, Khoury, E.: The predictive capacity of Garch-type models in measuring the volatility of crypto and world currencies. PLoS One **16**(1) (2021)
30. Nguyen, H.V., Naeem, A., Wichitaksorn, N., Pears, R.: A smart system for short-term price prediction using time series models. Comput. Electr. Eng. **76**, 339–352 (2019)
31. Park, J.S., Sung Cho, H., Sung Lee, J., Chung, K.I., Kim, J.M., Kim, D.J.: Forecasting daily stock trends using random forest optimization. 2019 Int. Conf. ICT Converg. (ICTC) pp. 1152–1155 (2019)
32. Pérez-Cruz, F., Afonso-Rodriguez, J.A., Giner, J.: Estimating garch models using support vector machines. Quantitative Finance **3**(3), 163 (2003)
33. PyTorch: PyTorch Tutorials. `https://pytorch.org/tutorials/` (2022)
34. Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., Cottrell, G.: A dual-stage attention-based recurrent neural network for time series prediction. Int. Joint Conf. Artif. Intell. (IJCAI) (2017)
35. Ramedani, Z., Omid, M., Keyhani, A., Shamshirband, S., Khoshnevisan, B.: Potential of radial basis function based support vector regression for global solar radiation prediction. Renew. Sust. Energ. Rev. **39**, 1005–1011 (2014)
36. Raudys, A., Pabarskaite, Z.: Optimising the smoothness and accuracy of moving average for stock price data. Technol. Econ. Dev. Econ. **24**(3), 984–1003 (2018)
37. Reiff, N.: What are the advantages of paying with Bitcoin? (2021)
38. Rogojanu, A., Badea, L.: The issue of competing currencies. Theor. Appl. Econ. **590**(1), 103–114 (2014)
39. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815 (2017)
40. Tixier, A.: Notes on deep learning for NLP. Computer Science Department (DaSciM team), École Polytechnique, Palaiseau, France (2018)
41. Zhang, X., Zhou, Y., Wang, J., Lu, X.: Personal interest attention graph neural networks for session-based recommendation. Entropy **23**(11) (2021)