Predicting Bitcoin Prices with Recurrent Neural Networks

Josep Haines - 21033821, Reece Turner - 22036698, James Burt - 20016437
University of the West of England

Abstract

With Machine Learning (ML) and Finance intersecting in recent years, this study aims to bridge the gap between the subjects using ML models as the vehicle for accurately predicting Time Series data on Cryptocurrencies. The project evaluates Time Series Models such as Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and compare against others, to investigate how they perform on historical price data without being discriminated to a bias. Our Bitcoin dataset covers the last 4 years due to market conditions caused by COVID-19, recessions and the 2024 halving event. This makes it a perfect time to analyse and evaluate models by splitting the dataset into training, validation, and test sets. In doing so, we can investigate not only how the models learn and adapt to hyperparameters, but also evaluate each model's ability to generalise unseen data. Our study will measure the performance by activation functions and mathematical proofs to further optimise predictions.

This study uses supervised learning with performance bias checks, such as Mean Squared Error, to measure its applicability towards real world predictions with data coming from public Interfaces such as Yahoo Finance which is stored locally with its original integrity for convenience.

Keywords: LSTM, GRU, RNN, ANN, model, accuracy.

1 Introduction

While cryptocurrency offers lucrative returns, it's inherently volatile as the fluctuation of people's interest in the decentralized space shifts causing observations to become a challenge with human intervention. However, using machine learning we can adopt learning models to predict potential future forecasts of timeseries data price action to determine when investing is the most appropriate. Leveraging datasets with machine learning models can improve accumulative returns on the market whilst addressing the efficiency and accuracy metrics of our models. Integrating computational learning into our research would demonstrate the intelligence of machines to perform decision making upon the use of training data and with the use of LSTMs we can expand our domain knowledge about market predictions in the future to fine-tune accuracy if needed.

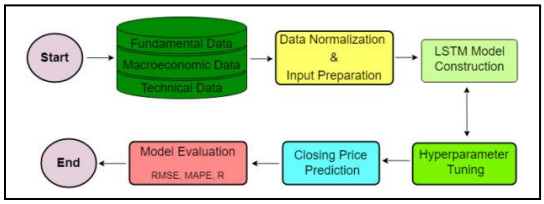
- We show how LSTMs can be used to perform market predictions with high accuracy.
- We propose GRUs and LSTMs effectiveness sequentially within the scope via similar traits and proficiencies.
- We demonstrate how the models bridge the gap of general purpose Artificial Neural Networks in relation to time series data.

2 Related Work

The connection between Machine Learning and Finance has become increasingly prominent in recent years. A significant area of research focuses on applying ML models to financial time series forecasting, particularly cryptocurrency prices due to their inherent volatility. This section explores related work and similar projects on Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, investigating their effectiveness in cryptocurrency price prediction.

A common practice of predicting cryptocurrency and stock exchanges in the past was using exclusively linear models to predict prices. Some algorithms have multiple kernels SVM's for example have 3, one being linear: support vector machines (SVM) (Hajek et al., 2023; Moula et al., 2017), suffer from some problems such as susceptibility to overfitting and do not fully exploit the potential of extracting high-level hidden patterns from cryptocurrency sequential data. (Chung et al., 2014).

Many studies that attempt to predict market or cryptocurrency trends with AI use normalisation to initially prepare the data for training. In our model we used a scaler to adjust our datapoint ranges from real world values to more abstracted data that can be trained with the model.



This image represents the usual step by step model of evaluating, training, and predicting data from the stock market (Nath Bhandari a et al., 2022)

2.1 Long Short-Term Memory Models

LSTMs were first introduced by Sepp Hochreiter and Jürgen Schmidhuber in a paper from 1997: Recurrent networks can in principle use their feedback connections to store representations of recent input events in form of activations (Hochreiter & Schmidhuber, 1997).

LSTMs have established themselves as a powerful tool for time series forecasting due to their ability to mitigate the long-term dependency problem - vanishing and exploding gradients, (P. Le, W. Zuidema, 2016). In our case connecting information over instances of timesteps. In the context of timeseries prediction, LSTMs have shown promising results in capturing complex patterns and generating accurate forecasts demonstrated by their creators but it's crucial to acknowledge that timeseries datasets are sequential compared to datasets with no order or features meaning LSTMs cannot shuffle data pseudo randomly as linearity, e.g. functions like tanh and sigmoid, compromises the datasets integrity therefore producing inaccurate model predictions. This being addressed, you can pass the current memory, C_t, into the next cell which creates a weighted running memory which allows the formation of another neuron "Dense" layer. (M. M. Patel, 2020)

LSTMs have shown evidence that they are 'far superior' (T. d. Oliveira Lima et al, 2021) than traditional methods of machine learning performing with higher accuracy and better efficiency at *higher timesteps* than prior methods especially when dealing with nonlinear data and when performing a hypothesis test on the results there was significant evidence to suggest significant differences between 'traditional' methods of machine learning and LSTMs (T. d. Oliveira Lima et al, 2021).

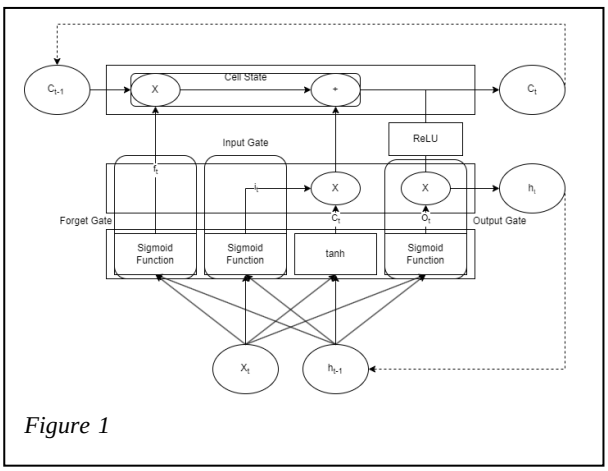


Figure 1

$$i_t = \sigma(x_t V_i + h_{t-1} W_i)$$
$$f_t = \sigma(x_t V_f + h_{t-1} W_f)$$
$$o_t = \sigma(x_t V_o + h_{t-1} W_o)$$
$$\overline{C_t} = \tanh(x_t V_g + h_{t-1} W_g)$$
$$C_t = \sigma(f_t \cdot C_{t-1} + i_t \cdot \overline{C_t})$$
$$h_t = \tanh(C_t) \cdot o_t$$

In **Figure 1**, x_t is the input, C_{t-1} is the output of the previous cell's memory, h_t is the output of the current cell, C_t is the memory of the current cell. W and V are the weights of the cell. We used sigmoid for the recurrent activation function as the data is stochastic, not linear, therefore it's a better approach to use a non-linear function, The ReLU activation function appears in **Figure 1** as ReLU introduces nonlinearity to help mitigate the vanishing gradient problem. The dashed lines on the diagram are to represent variables that are passed into the next LSTM cell.

2.1.1 Cell Gates

Forget Gate: The forget gate determines what information to discard from the cell's previous state h_{t-1}. It assigns weights between 0 and 1 to the previous cell state, with 0 indicating complete forgetting and 1 indicating full retention. It acts like a filter, assigning weights between 0 and 1 to each element in the previous state. A weight of 0 signifies the network "forgets" that specific piece of information entirely. A weight of 1 indicates the network retains that information completely. The forget gate decides what to forget from the information received through previous memory units (ArunKumar et al., 2022).

Input Gate: The input gate controls what new information is to store in the cell state. It considers the current input x_t and the previous cell state h_{t-1} to generate a candidate value for the new cell state.

Output Gate: The output gate determines what information from the current cell state to output. It considers the current cell state c_t and the previous cell state h_{t-1} to generate the output h_{t-1}.

These gates work together to ensure that only relevant information persists within the cell state over time. This enables LSTMs to effectively learn long-term dependencies within sequential data, making them particularly suitable for time series forecasting tasks like cryptocurrency price prediction.

2.1.2 Network Layers

Input Layer: For predicting the value of BTC, we pay special attention to data preprocessing in the input layer, to avoid overfitting in the estimation and optimization process, and assure correct selection of hyperparameters at the beginning of our tests. (Michańków et al., 2022). This layer is where pretrained cryptocurrency data is fed into the network. For some models this may include historical closing prices, trading volumes or other indicators that may be useful or relevant.

LSTM layer: The LSTM layer has multiple nodes a common value is 128 LSTM cell stacks one after another for each single layer. During training it will process each cell point by point at each step the forget gate analyses the previous cell state h_{t-1} and decides what information to discard (temporary price fluctuations). The input gate considers the current data point x_t and the previous cell state, determining what new information to learn (significant price movements or trends). The output gate controls what information from the current cell state is shared with the next layer h_t. It essentially focuses on the most relevant aspects of the price history for future predictions. (LSTM as an input, no additional hidden layers), and SGD (stochastic gradient descent algorithm) as optimizer. (Michańków et al., 2022).

Hidden layer: These layers can be contained within the LSTM and Dense layers with the purpose to further process information extracted from the LSTM layer. Hidden layers can help predict hidden patterns and relationships.

Dense Layer: or fully connected layer is a single hidden layer that has a single cell and often uses an activation function such as 'ReLU' and optimizer 'RMSprop.' Sequence length of days (Michańków et al., 2022).

Output layer: depending on the usage the output layer may have different structure, if the goal is to predict future price movement the output layer might have multiple neurons to specific categories.

2.2 Gated Recurrent Networks

GRUs are another type of RNN architecture like LSTMs, but with a simpler gating mechanism. While LSTMs employ three gates (forget, input, and output), It has two gates, namely, an update gate and a reset gate. Together these two gates control the flow of information through the network. (Patel, 2020) This more abstracted structure makes GRUs computationally more efficient than LSTMs while maintaining the ability to learn long-term dependencies. When GRU's were first used they were introduced as another option to be used instead of LSTMs. They had shown promising results in many machine learning tasks, especially when input and/or output are of variable length (Chung et al., 2014). This paper was to prove how a GRU could be more efficient when given a specified length of data to sequence train different models.

$$u_t = \sigma(V_u\, x_t + W_u\, o_{t-1} + b_u)$$
$$r_t = \sigma(V_r\, x_t + W_r\, o\_(t-1) + b_r)$$
$$i_t = \tanh(V_o\, x_t + W_o(r_t \odot\ o_{t-1}) + b_o)$$
$$o_t = u_t \odot o_{t-1} + (1 - u_t) \odot i_t$$

This is where $x_t$is the input, o_t is the output, u_t is the update gate output, r_t is the reset gate output, $\odot$ is the Hadamard product and V, W and b are parameters and weights. The Hadamard product is a binary operation that receives two matrices of the same dimensions and returns a matrix with each element multiplied.

Update Gate: The update gate determines how much of the previous cell state h_{t-1} to keep and how much new information from the current input X_t to incorporate. It generates a value between 0 and 1, indicating the proportion of the previous state to keep. These equations represent the hidden state of the GRU: (ArunKumar et al., 2022)

$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t$$

Reset Gate: The reset gate controls the extent to which the network relies on past information. It assigns weights between 0 and 1. This gate has an activation referenced as r_t this can be computed using a sigmoid function applied to a linear transformation of the concatenation of x_t and h_{t-1}:

$$r_t = \sigma(V_r x_t + W_r o_{t-1} + b_r)$$

Overall, a GRU is a functional and useful type of RNN that could be used to train and predict our data. Despite it being a more simple and linear approach it could be a more efficient function so it would be worth implementing to at least compare to the LSTM. A GRU from the research may be better as a day-trading tool to predict shorter timeframes but may be less consistent and be more prone to overfitting than a traditional LSTM.

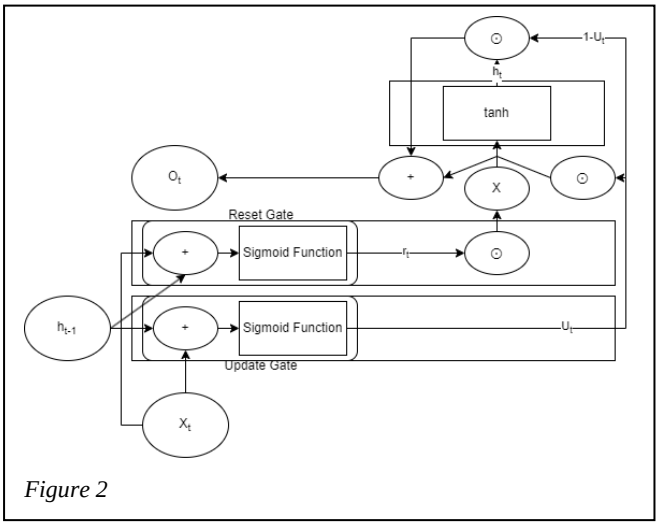


Figure 2

3 Experiments

Table 1: Parameters

Parameter	Values
N_TIMESTAMP	10
TEST_SIZE	0.15
UNITS	128
ACTIVATION_FUNCTION	relu
RNN_ACTIVATION_FUNCTION	sigmoid
RNN_OPTIMISER	adam
LOSS_ERROR	mse
EPOCH	32
PATIENCE	5
BATCH_SIZE	32

Table 2: Grid Search Parameters

Parameter	Values		
UNITS	32	64	128
ACTIVATION_FUNCTION	relu	tanh	-
RNN_ACTIVATION_FUNCTION	sigmoid	tanh	-
RNN_OPTIMISER	adam	rmsprop	-

A common way of experimenting and testing many possible parameters is a Grid Search where you can pass in many different parameter options, and it will test every combination and return the best result. The possible options tested are shown in **Table 2**.

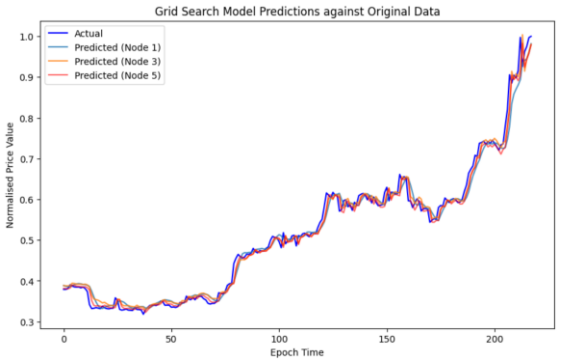


Figure 3

Figure 3 shows three nodes from within the Grid Search each with a different combination of parameters. As seen in this example, overall node 5's (red) predictions were the most accurate with the actual data, with nodes 1 and 3 performing worse. As node 1 (light blue) and 3 (orange) were performing similar this may suggest that they have similar parameters in the Grid Search and therefore also perform similar.

3.1 Methods

Datasets and Pre-processing

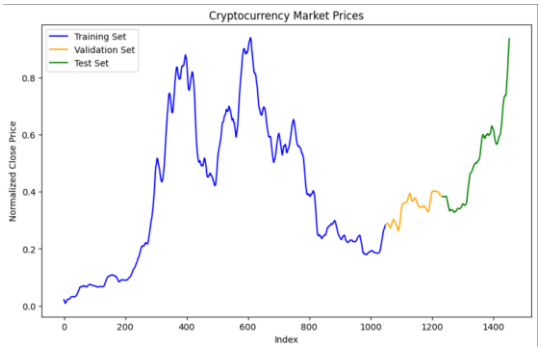


Figure 4

Dataset: Our chosen dataset was comprised of historical real-world data obtained from a RESTful API request from Yahoo Finance. The dataset features initially are Open, High, Low, Close and Volume (OHLCV) with respect to our chosen Cryptocurrency – Bitcoin.

With consideration of the features, our model presents a univariate configuration meaning a single feature is only required to create a model with. We decided to use only the close price as this is the final candlestick price before the next timestep therefore providing us with accurate Bitcoin price data. The other information is irrelevant to our LSTM or GRU as we are only interested in creating a model that can accuracy predict future close prices however it is possible to use these other features to create indicators or other models.

Preprocessing: For this step we took our univariate feature and proposed normalisation to decrease computational expensiveness reducing our time complexity for the LSTM to $O(1)$ according to (Hochreiter & Schmidhuber, 1997). Our model uses scaling of $[0, 1]$ range which you can view in the Y axis on **Figure 4**. This scaling also will benefit future predictions and model estimators such as Grid Search as the normalisation applies a combination of parameters in search for the best hyperparameters. From here our preprocessing involved taking the scaled close price

$$scaled\ close\ price = \frac{Close_i - \min(Close)}{\max(Close) - \min(Close)}$$

and creating a multi-dimensional array (reshaping), X, of n samples, t steps and f features; additionally, we also generated y being our Next Close day. This was calculated by:

$$y = scaled\ close\ price_{i+t}$$

Splitting Data: Finally, to finish preprocessing our dataset we introduced the train test split function from sci-kit learn. With a training set of *70%* and a test set of *15%*, we compute the remainder for validation. This is an important step because the models we used learn on the training set, being around the COVID-19 (2020) timeframe with said validation set designed to optimise learning extremes from the volatility. The test data created will be unseen until we demonstrate predictions.

Training Procedure

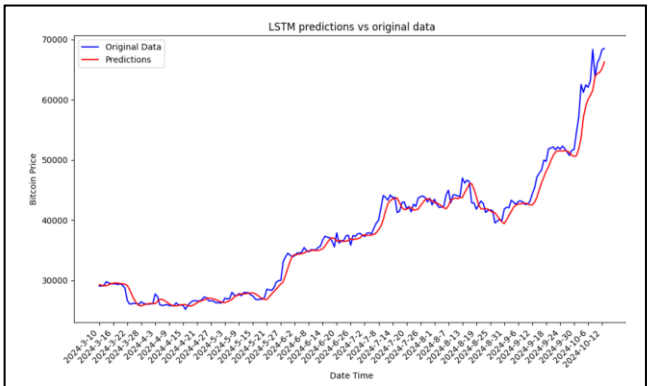
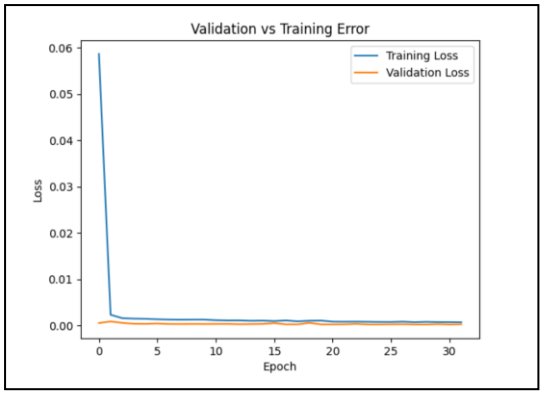


Figure 3

Figure 5 represents our test data against our $X\ test$ prediction – this data was predicted then the scaler was inversed hence the price data on Y axis and datetime on X axis. **Figure 5** configuration was the initial test to see how plausible LSTMs are for our project which turned out to be a viable base for future predictions. Our model incorporates the parameters mentioned in **Table 1** where *EPOCH* was set to an excessive *32* epoch. While the number is quite large for task our model takes advantage of TensorFlow/Kera's "Early Stopping" object. This object is a callback method passed as a parameter in the fit function, designed to prevent over or underfitting after a specified amount of *patience*. Typically, a large patience makes the time complexity worse due to excessive computation however setting to *5* we found it was the most appropriate for exhaustive searching. The result of doing this produced the graph below:



Moreover, our constants passed as parameters in **Table 1** used *128* units which was recommended in a paper by (Pimenidis, E. et al, 2022). Upon information and belief (3.2), our LSTM model outperforms previous models mentioned due to our holistic set of current hyperparameters which demonstrates a "training loss" of *0.0006728420848958194* and a "validation loss" of *0.0001981903478736058* towards the convergence point.

3.2 Results and Discussion

Table 3: Grid Search Mean Squared Error

MSE	Values	Activation	Optimiser	Recur. Act.	Units
Predicted Node 0	0.0001 9165	relu	adam	sigmoid	32
Predicted Node 2	0.0001 8613	relu	adam	sigmoid	128
Predicted Node 3	0.0001 6952	relu	adam	tanh	32
Predicted Node 4	0.0001 2722	relu	adam	tanh	64
Predicted Node 17	0.0001 0715	tanh	adam	tanh	128

Table 3 is composed of the model prediction losses over a range of epochs, with easy stopping in mind. The displayed values are from performing a Grid Search where the best output becomes the weight for the new instance. When the search runs, we only append the best (lowest) Mean Squared Error (MSE) for that instance to an array then optimises as part of the next iteration in the exhaustive search.

$$MSE = 1/n \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

To make it a fair test we also run the same notebook on two computers and both machines produces a variety of values where the best result discovered was by *Node 17* being *0.00010715* which is a 54% accuracy with a 46% decrease in loss from the original model without grid search demonstrating the inclusion of validation set with parameter changes is in fact reducing error rate.

Moreover, between *Node 3* and *Node 4* the model has the same parameters except for the units, which is increased to *64* from *32*. It seems that this run shows a *75%* accuracy with a *25%* loss. Conversely, *Node 1* and *Node 2* delta is *97%* accuracy with 3% loss but has double the units compared to *Node 3* and *Node 4* – significant improvements were reduced from this point regardless of the activation function and unit size.

The statistics show that the original configuration found a reduction in loss by 21% therefore by the time our search reached *Node 4*, which was further optimized by Grid Search to produce a suitable decay difference of ~56%, with an overall average mean loss of *0.000156334*. The data clearly shows our model as proven its proficiencies in being able to observe complex data patterns without producing invalid local optimums using the gating mechanisms. This can be seen on **Figure 3** where the loss rate is relatively high between the manual model prediction and the original data whereas towards the end of the predictions the spread is tighter for *Node 3*.

4 Conclusion

This study explored the potential of LSTM's for predicting cryptocurrency prices. The LSTM we implemented has

demonstrated promising results in capturing patterns and trends within the Bitcoin cryptocurrency price data that we used to train. From the test and validation data it was able to accurately predict results without knowing the price data from each timestep during the walk-forward process.

GRU as this is another prominent RNN that is popularly used which we could have implemented. Despite it being computationally less expensive it is more efficient when used for sequential data, but not long-term, so it seemed unnecessary to implement a less optimised tool for our scenario. Overall, we found from our research that the LSTM model would use our dataset in the most optimal way than other models like the GRU and ARIMA (Auto-Regressive Integrated Moving Average), however they could've been investigated for more short-term predictions and perhaps a classification model could be used univariately to hyperplane data via binary classification rather than regression.

By continuously developing and applying our solutions to these types of problems we refine our understanding and application of machine learning models for not just the financial domain, but also the best way to optimise specific algorithms for any real-world datasets appropriately. We have solved the problem of predicting Bitcoin prices with a relatively robust and reliable tool.

5 References

Hochreiter, S. and Schmidhuber, J. (1997) *Long short-term memory | MIT Press Journals & Magazine | IEEE Xplore*. Available at: https://ieeexplore.ieee.org/abstract/document/6795963/ (Accessed: 29 April 2024).

Chung, J. et al. (2014) *Empirical evaluation of gated recurrent neural networks on sequence modelling, arXiv.org.* Available at: https://arxiv.org/abs/1412.3555 (Accessed: 20 April 2024).

Patel, M.M. (2020) *Journal of Information Security and Applications*, Journal of Information Security and Applications ScienceDirect.com by Elsevier. Available at: https://www.sciencedirect.com/journal/journal-of-information-security-and-applications (Accessed: 20 April 2024).

T. d. Oliveira Lima, M. Colaco, K. H. de J Prado, I. Dias de J. and F. R. de Oliveira (2021) *A Big Data Experiment to Evaluate the Effectiveness of Traditional Machine Learning Techniques Against LSTM Neural Networks in the Hotels Clients Opinion Mining.* | IEE Available at: https://ieeexplore.ieee.org/abstract/document/9671939 (Accessed: 29 April 2024).

E. Pimenidis, P. Angelov, C. Jayne, A. Papaleonidas, M. Aydin (2022) *Artificial Neural Networks and Machine Learning – ICANN 2022* Available at: https://link.springer.com/book/10.1007/978-3-031-15934-3 (Accessed: 27 April 2024)

Ahmed Bouteska, Mohammad Zoynul Abedin, Petr Hajek, Kunpeng Yuan (2024) *Cryptocurrency price forecasting – A comparative analysis of Ensemble Learning and Deep Learning Methods, International Review of Financial Analysis.* Available at: https://www.sciencedirect.com/science/article/pii/S1057521923005719 (Accessed: 29 April 2024).

Arun Kumar, K.E. et al. (2022) *Comparative analysis of gated recurrent units (GRU), long short-term memory (LSTM) cells, autoregressive integrated moving average (ARIMA), Seasonal Autoregressive Integrated moving average (SARIMA) for forecasting COVID-19 trends*, Alexandria Engineering Journal. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9453185/ (Accessed: 20 April 2024).

Michańków, J., Sakowski, P. and Ślepaczuk, R. (2022) *LSTM in algorithmic investment strategies on BTC and S&P500 index, Sensors (Basel, Switzerland)*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8839390/ (Accessed: 21 April 2024).

Mohil Maheshkumar Patel, Sudeep Tanwar, Rajesh Gupta, Neeraj Kumar. (2020) *A Deep Learning-based Cryptocurrency Price Prediction Scheme for Financial Institutions* Available at: https://www.sciencedirect.com/science/article/abs/pii/S2214212620307535 (Accessed: 24 April 2024).

Phong Le, Willem Zuidema (2016) *Quantifying the vanishing gradient and long distance dependency problem in recursive neural networks and recursive LSTMs* Available at: https://arxiv.org/abs/1603.00423 (Accessed: 28 April 2024)

Hum Nath Bhandari a et al. (2022) *Predicting stock market index using LSTM, Machine Learning with Applications.* Available at: https://www.sciencedirect.com/science/article/pii/S2666827022000378 (Accessed: 01 May 2024).