An LSTM Approach to Predicting Ethereum's Price Trend Based on Blockchain Characteristics

Menno van Dijk
Tilburg University
m.a.c.vandijk@tilburguniversity.edu

Abstract

This paper investigates the possibility of predicting price movements of the world's third most valuable cryptocurrency by market capitalization, Ethereum. More specifically, the goal is to forecast the trend of Ethereum's price, up or down. It attempts to do so by employing multiple variations of LSTM neural networks, which are trained on a subset of features belonging to Ethereum's blockchain. These features are absent of any information pertaining to the historical price or market characteristics of Ethereum. The results show that it might be possible to predict trend movements with a greater than random chance using one of the proposed LSTM neural network architectures. More research is needed to thoroughly substantiate these claims.

1 Introduction

Predicting stock market prices using machine learning techniques has been investigated in the financial literature at great length, of which a summary can be found in Daflapurkar, Hule, Nirwan, Rao and Shaikh (2015). Contrary to the Efficient Market Hypothesis (Fama, 1965), this strain of literature states that it is possible to predict the price trend of the stock markets with a greater than random chance. However, while extensive literature exists about predicting price trends in these traditional financial markets, research into the same domain is relatively sparse when taken into the context of non-traditional financial markets. One of the most prominent non-traditional financial markets at the time of writing is the cryptocurrency market. While Bitcoin (Nakamoto, 2008) is the leading cryptocurrency in terms of market capitalization, Ethereum has grown to be the 3rd largest cryptocurrency in a short period of time[1]. Due to this fact, the literature with regards to predicting price trends of Ethereum is even more sparse compared to that of Bitcoin.

Nevertheless, evidence in favor of predictability of cryptocurrency price trends can be found. Bilanakos, Georgoula, Giaglis, Pournarakis and Sotiropoulos (2015) analyze the determinants of the price of Bitcoin by employing various machine learning techniques. They analyze Twitter sentiment surrounding Bitcoin by employing a Support Vector Machine and subsequently regress this sentiment measure on the price of bitcoin. Evidence of these regressions show that Twitter sentiment can be seen as a determinant for the price of Bitcoin. Furthermore, they analyze the effect of several characteristics of the Bitcoin blockchain (i.e. hash rate, total supply), including the amount of Google searches and Wikipedia views Bitcoin receives, on its respective price. The results show that all of these measures have a positive and statistically significant association with the price of Bitcoin. Additionally, Garcia and Schweitzer (2015) show that twitter sentiment on Bitcoin has a positive lagged effect on Bitcoin prices. These insights are put to the test in an out-of-sample trading scenario in which their strategy vastly outperforms the baseline strategy of buying Bitcoin at $t = 0$ and holding until the end of the sample period. The latter strategy is henceforth referred to as the "buy and hold" strategy and will subsequently be applied in this paper as a baseline measure to test for model performance.

An approach outside of using sentiment analysis and more in line with the approach taken in this paper is that of Caton, McNally and Roche (2018). Using both an RNN (Hopfield, 1982) and an LSTM (Hochreiter & Schmidhuber, 1997) neural network architecture, an attempt is made to predict the price trend of Bitcoin, using features related to historical prices as well as blockchain characteristics. More specifically, mining difficulty and hash rate of the Bitcoin blockchain are taken as a dependent variable for the model, as well as the opening price, daily high and daily low price of the day prior to the prediction. Lastly, two moving averages with different time windows are included to capture past price trends. After hyper-parameter optimization, results for the RNN architecture show that in 50.25% of cases the correct price trend can be predicted. The LSTM neural network increases the accuracy of predictions to 52.78%, which can be attributed to the fact that LSTMs are better able to capture longer-term dependencies. However, while the LSTM neural network architecture achieved a better performance in terms of accurately predicting price trends, the root mean squared error (RMSE) for predicting the actual return of Bitcoin's price was slightly worse, 8.07% compared to 7.15% for the RNN architecture, respectively. While these results are only marginally better than the ran-

[1] As of 30-12-2018.

dom chance of 50%, it does show how out-of-sample price trends can be predicted using recurrent artificial neural networks.
Another paper employing artificial neural networks is that of Pichl and Kaizoji (2017). Their paper shows how a feed-forward neural network with two fully interconnected hidden layers, each having 10 neurons, can capture most of the variation in the distribution of log returns for Bitcoin's price. The predictor used in this model was a simple moving-average of the past 10-day log returns. These results show that with a relatively simple neural network architecture, whilst simultaneously employing merely one easily obtainable predictor, it is possible to estimate how the price of Bitcoin tends to behave.

However, while all of the aforementioned literature concerns the predictability of Bitcoin's price, research into the same domain but concerning Ethereum is sparse. Nevertheless, one paper by Chen, Narwal and Schultz (2017) shows how various machine learning techniques perform when trying to predict Ethereum's price change. Their input into the model is a price series of the 5 days preceding the target prediction. Moreover, the target prediction was binary, +1 for a price that had risen on the target day and a -1 for a price that had dropped on the target day. Multiple methodologies were then employed, including Logistic Regression, Support Vector Machine, Random Forest, Naïve Bayes, Autoregressive Integrated Moving Average (ARIMA), Recurrent Neural Network and a plain feed-forward Neural Network. Every model showed that they were able to predict the price trend with a greater than random chance. It must be noted however that the RNN network, which included LSTM cells, performed amongst one of the worst when put to the task of predicting the directionality of price change within this study (with an accuracy of 52.43%). The best results were obtained by the ARIMA model, which had an accuracy score of 61.17%, substantially better than the random chance of 50%.

Subsequently, Aiello, Alessandretti, Baronchelli and Elbahrawy (2018) analyze the cryptocurrency market as a whole. They show that it is possible to generate returns in excess of their benchmark with various machine learning techniques. Their dataset consisted of daily prices for 1104 cryptocurrencies, between November 11, 2015 and April 24, 2018. Features used for prediction were merely related to historical data derived from the price of the respective cryptocurrency (i.e. market capitalization, standard deviation of price, etc.). Using these features, they employed three different models to predict the prices of all cryptocurrencies that were investigated. Model 1 and 2 were based on gradient-boosted decision trees and model 3 was based on an LSTM neural network architecture. In order to analyze the performance of the models, the predicted future prices were converted into signals for a trading scenario. If the price was predicted to go up by the model within a specific time window, a buy signal would be given, if on the contrary the price was predicted to go down then a sell signal would be given. Their baseline measurement was based around a simple moving average crossover strategy (see i.e., Gunasekarage & Power, 2001). Subsequently, the cumulative returns of these strategies were compared and showed that model 3 vastly outperformed all other models, with returns outperforming the second-best model by a factor of 3. This study indicates the possible strength of using LSTM neural networks for predicting price trends in cryptocurrencies.

Considering the fact that all of the above literature takes into account historical price information or sentiment to predict future price trends, this paper will take a novel approach. More specifically, this paper will investigate whether it is possible to predict future price trends by merely employing historical data regarding Ethereum's blockchain. Hence, the research question becomes: Is it possible to predict future price trends in Ethereum based on its blockchain characteristics with a greater than random chance?

In order to answer this question, the LSTM neural network architectures within this paper will be trained to predict the future return of Ethereum based solely on its past blockchain characteristics. The predicted return series will then be converted to a series of binary trading signals, +1 for a prediction that the price will rise on the target day and a -1 for a price that is predicted to go down. Furthermore, several error and performance metrics generated by the outcome of the predictions will be compared to baseline measurements. The error of predictions will be compared against the predictions laid forth by the Efficient Market Hypothesis, stating that there is a 50/50 chance that the price will move or down on a given day. The profitability of the trading strategy resulting from the predictions will be compared against the baseline of buying Ethereum on t = 0 and holding until the end of the sample period. Historical prices and blockchain characteristics for the Ethereum blockchain have been gathered for the period from 30 July 2015 to 21 May 2018. Comparing the results obtained by the proposed methodology in this paper to the baseline measurements, it can be seen that it might be possible to predict the price trend of Ethereum with a greater than random chance. These results should be taken with caution and more research is required to thoroughly substantiate these claims.

The paper is structured as follows. Section 2 will discuss the employed methodology. Section 3 will digress on the data, experiments, hyper-parameter tuning and the evaluation criteria used in this paper. Section 4 will display the results obtained. Section 5 will discuss opportunities for improvement and give guidelines for future research into the topic at hand. Section 6 concludes.

2 Methodology

Predicting the behavior of (financial) time-series is a notoriously difficult task for academics and companies alike, especially in light of the arguments put forth by the Efficient Market Hypothesis. However, the LSTM neural network architecture is capable of achieving state-of-the-art results when it comes to forecasting time-series (Berkowitz, Elkan & Lipton, 2015), as has also been shown in the aforementioned literature. Therefore, in order to answer the research question, this paper will employ LSTM neural network architectures.

More specifically, this paper will employ three different LSTM neural network architectures. The choice was made to compare three different architectures, as the exploratory stage of this research revealed it could be of interest to investigate how different architectures perform on the task at hand. The reasoning behind all hyper-parameter choices per models will be discussed in further detail in section 3.3 of this paper. All model architectures and their respective hyper-parameters are listed below.

Model 1: The first model consists of a single hidden LSTM layer with 256 neurons that goes into a single feedforward output layer with 1 neuron. The RMSprop optimizer with a learning rate of 0.0005 was utilized in order to train the model over a series of 10000 epochs with a batch size of 128. Lastly, all of the models make use of the mean squared error loss function as well as a linear activation function at the output layer.

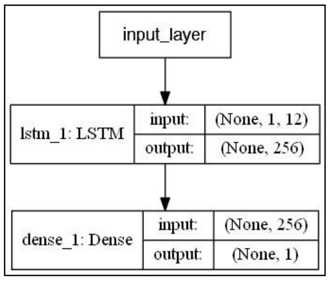

Figure 1: A visual representation of the LSTM neural network architecture of model 1 with specified input and output dimensions.

Model 2: The second model consists of a stacked LSTM neural network architecture with two hidden LSTM layers, each having 64 neurons, which eventually go into a single feedforward output layer with 1 neuron. This model made use of the RMSprop optimizer with a learning rate of 0.0005 and was trained over 2500 epochs with a batch size of 64.

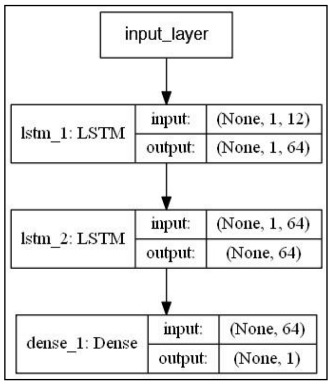

Figure 2: A visual representation of the LSTM Neural network architecture of model 2 with specified input and output dimensions.

Model 3: The third model is based of Karakoyun and Cibikdiken (2018) and has a more complex architecture. It consists of 3 hidden LSTM layers and 2 hidden fully connected feedforward layers. The first LSTM layer has 64 neurons, while the second and third have 128 neurons. The first fully connected feedforward layer has 8 neurons, while the second has 4. Between all layers but the second fully connected feedforward layer a ReLU activation function is being applied. Lastly, the RMSprop optimizer with a learning rate of 0.0005 was utilized to train this model over 7500 epochs with a batch size of 128.

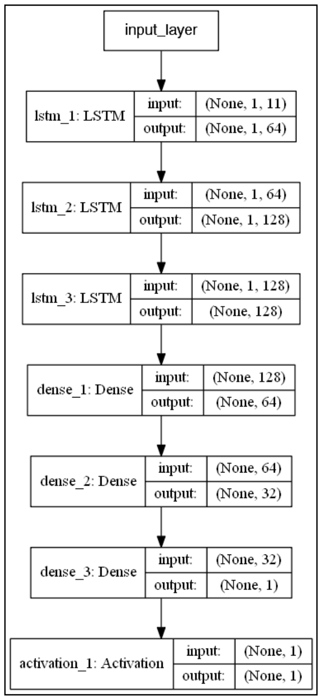

Figure 3: A visual representation of the LSTM neural network architecture of model 3 with specified input and output dimensions. Note that ReLU activation layers are not shown within this figure, this was done in order to keep the dimensions of the figure to an acceptable size.

3 Experimental Method

The following section will digress on the data employed by this study, the experiments conducted in order to achieve the eventual results and how the final selection of hyper-parameters was established. Lastly, the error metrics corresponding to the price trend predictions, as well as the profitability metrics corresponding to the proposed trading strategy will be discussed.

3.1 Data

The data employed for training and validation was obtained from Kaggle[2], test data was obtained from Etherscan[3]. Training the model was done on 80% of the data, validation

[2] https://www.kaggle.com/sudalairajkumar/cryptocurrencypricehistory/data#ethereum_dataset.csv

[3] https://etherscan.io/charts

on 10% and testing on the remaining 10%. This split is not very common but due to the sparse availability of the data, it made sense to use more data for training purposes. Both datasets contain 17 features including the target, Ethereum's price. All of the data was date-stamped in chronological order. A total of 1055 observations were present in the datasets, for the period from 30 July 2015 to 21 June 2018. However, since the price of Ethereum was at zero during its inception date, including some of the following days, the decision was made to remove the first 13 rows of data. Furthermore, columns within the dataset pertaining to supply and market cap were removed as they are not inherently tied to Ethereum's blockchain, which therefore renders them outside the scope of this papers' research question. Moreover, the Unix timestamp column was removed since it proved of no additional value as a predictor. Lastly, the ENS registers column was removed. This register was a tool created by the instigators of Ethereum in order to convert a hexadecimal address to human readable text. This allowed users to send and receive Ethereum directly to or from a prespecified name instead of an address. While ENS registers were extremely popular right after the inception of Ethereum, they are virtually non-existent at the moment. The great disparity in number of registers and the volatility thereof made this column impractical for usability as a feature.

Additionally, in order to correct for the de facto block time halving that occurred on the 16th of October[4], all rows succeeding this date and in direct relationship with block time were multiplied by a factor 2. This was done since this change did not occur by market participants, but in light of a so-called "hard fork". Within a hard fork the developers of the currency create an upgraded version of the blockchain software and subsequently push it to all its users. Since neural networks are notoriously sensitive to the way inputs are supplied, this choice had to be made. Moreover, initial testing showed that the model was profoundly misinterpreting the predictive value belonging to this change.

Lastly, in light of the abovementioned, feature and target preprocessing was conducted. The target was converted from a price series to a return series by applying the following transformation: $R_t = (P_t - P_{t-1}) / P_t$. Doing this ensures that the data is detrended and normalized, which is important for use in time-series analysis (see i.e., Huang, Long, Norden & Peng, 2007). Features were preprocessed by standardizing them to have zero mean and unit variance. They were also lagged by one day so features of time t-$_1$ predict returns of time t.

3.2 Pipeline Procedure

Each experiment conducted within this paper went through the same pipeline. Firstly, all preprocessed features were reshaped to a 3D array with dimensions (*obs*, 1, 11), where *obs* is the number of observations and varies between the training, test and validation set, 1 is the time steps to predict ahead and 11 is the number of features used for prediction. Afterwards, each model was instantiated, compiled and fit with the optimal hyper-parameters. Return predictions were then made and compared to the return series from the validation or test set. Subsequently, error and performance metrics are returned to analyze the results of the model predictions.

[4] See https://icodaily.net/ethereum-executes-hard-fork/

3.3 Hyper-Parameter Optimization

Optimization of the hyper-parameters for each model was done in two stages. Stage one consisted of finding the optimal optimizer, loss function, number of epochs and batch size while keeping the number of neurons in each hidden layer constant at 64. Although it would be best to optimize for all hyper-parameters simultaneously, separating the optimization process in multiple stages was an inconvenient necessity because of a sheer lack in computational power. Regardless, the first stage optimization process consisted of a grid search that iterated over the following parameters:

Epochs:	[500, 1000, 2500, 5000, 7500, 10000]
Batch size:	[64, 128]
Optimizers:	[Adam1, Adam2, RMSprop1, RMSprop2]
Loss function:	[MSE, MAE, MSLE]

Where Adam1 refers to the Adam optimizer with a learning rate of 0.001 and Adam2 refers to a learning rate of 0.0005, the same differential in learning rate holds for the RMSprop optimizer. Additionally, MSE refers to mean squared error and MAE refers to mean absolute error, MSLE refers to mean squared logarithmic error. Take note that the first stage optimization process was kept constant across all three models. The cumulative training time for the stage one optimization process was around 16 hours, executed on a GeForce GTX 960 graphics card. Eventually, for each of the three different architectures, the parameters with the lowest error metrics and highest performance metrics on the validation set were chosen for testing. Outcomes of the optimal hyper-parameters obtained from the first stage optimization process can be found in the methodology section of this paper.

After stage one optimization was finished, a second stage optimization process was conducted for each model individually. This stage optimized over the number of neurons present within the hidden layers in each model. Since the neural network architecture is incongruous model-to-model, the optimization procedure is incongruous as well. The following is a description as to how the optimization procedure of stage two was conducted for each model:

Model 1: Since there is only one hidden LSTM layer in the first model, parameter optimization for this model was relatively simple. It consisted of iterating over the following integer number of neurons:

Neurons:	[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]

The optimal number of neurons can be found in the methodology section of this paper.

Model 2: Considering the fact that model 2 has two hidden LSTM layers, parameter optimization is not much harder compared to the last model. The optimization procedure here consisted of a grid search over all possible combinations of the following integer number of neurons:

NeuronsL1:	[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
NeuronsL2:	[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]

Where L1 and L2 are in correspondence to the layer number provided in figure 2 of the methodology section. Similarly to model 1, the eventual choice of optimal parameters can be found in the methodology section.

Model 3: This model is substantially more complicated than model 1 and 2, thus requiring additional thought in order to optimize the parameters with taking into account the available computational resources. In accordance to Cibikdiken and Karakoyun (2018) the second and third layer are linked, meaning that the neurons within these hidden layers will always stay at the same level. This substantially reduces the strain put on the available computational resources, but might lead to less favorable results. Taking into account the aforementioned, optimization in this case consisted of a grid search over the following integer number of neurons:

NeuronsL1:	[4, 8, 16, 32, 64, 128, 256]
NeuronsL2/3:	[4, 8, 16, 32, 64, 128, 256]
NeuronsD1:	[8, 16, 32, 64]
NeuronsD2:	[2, 4, 8, 16]

Where L1 and L2/3 refer to all three LSTM hidden layers, while D1 and D2 refer to both fully interconnected feedforward layers. This is also portrayed in figure 3 of the methodology section. As with stage one, the number of neurons that scored best on both the error and performance metrics on the validation set were chosen for final testing. Total cumulative training time for stage two optimization was around 12 hours.

3.3 Error and Performance Metrics

In order to evaluate the accuracy of the predictions of price trend, a comparison will be made against the benchmark of predicting at random in the test set. Since the prediction of price trend is a binary classification task, the following metrics to determine the accuracy of predictions were employed:

Precision:	$TP / (TP + FP)$
Recall:	$TP / (TP + FN)$
F1-Score:	$(2 \times P \times R) / (P + R)$

Where TP is true positives, FP is false positives, FN is false negatives, P is precision and R is recall.

Subsequently, performance of the trading strategy based on each of the three models is evaluated, and compared to the benchmark strategy. In order to do so, the annualized Sharpe ratio (a risk-return metric) and the maximum drawdown (a risk metric) will be employed, as follows:

Sharpe: $(\sqrt{D} \times \mu_r) / \sigma_r$
Drawdown max: $(T - M) / M$

Where D represents the number of trading days in a year used for annualization, μ_r is the mean return of the trading strategy, σ_r is the standard deviation of the trading strategy, T is the trough of the trading strategy within the trading timeframe and M is the maximum value achieved by the trading strategy within the trading timeframe. Alongside these performance metrics, the overall cumulative compounded returns of the trading strategy and the buy and hold strategy will be compared. Note that D = 365.25 in this paper, as opposed to the value of 252 that is common in the traditional financial literature. This is the case since cryptocurrencies are exchanged throughout every day.

4 Results

In table 1 the error metrics for the baseline measure of predicting at random, as well as the error metrics for each trained model are displayed. It can be seen that two out of three models performed relatively well on the test set. While the predictions are better than the random chance of 50%, they are very fragile to changes in the hyper-parameters. In order to test for robustness, very modest changes in the number of epochs and batch size were conducted, which lead to completely different results. This could however also be attributed to the fact that it can be hard to create reproducible results when training deep neural networks on a GPU[5], due to the stochastic nature of initialization. Nevertheless, these results look mildly promising.

Metric / Model	Precision	Recall	F1-Score
Baseline	0.50	0.50	0.50
Model 1	0.5173	0.5169	0.5169
Model 2	0.4153	0.4157	0.4153
Model 3	0.5132	0.5286	0.52079

Table 1: Error metrics of predictions on the test set for all models and the baseline. The baseline strategy of randomly guessing yields 0.50 on precision, recall and f1-score by definition. Higher numbers are better.

Table 2 shows the results of the trading strategy that is based on the predictions from the trained networks. While it seems obvious to assume that a high accuracy of price trend prediction would lead to high performance metrics, this is not always the case. When the model is highly accurate in predicting price movements that are negligible, but highly inaccurate when predicting price movements that are extreme outliers, there is a high chance the trading strategy will still lose money. However, in the case of this paper it can be seen that high predictability leads to relatively good performance. This might be attributed to the fact that the price of Ethereum was not moving extremely volatile during the out-of-sample testing period. It can also be noted that model 3 seemed to perform the best in terms of all performance metrics. Investigating how further optimization of model 3 would increase performance on the test set could therefore lead to interesting results.

However, when applying the proposed profitable trading strategies (model 1 and 3) in a real-world trading scenario, it should be noted that slippage and fees will greatly affect performance. Slippage occurs when the expected price of a trade is different than the price it is actually executed at. The results of slippage can be detrimental to a trading strategy and it is hard to correct for in a simulated environment. Moreover, virtually all cryptocurrency exchanges charge trading fees based on the number of total transactions and the size thereof. While factoring in the fees paid by a trading strategy is possible to simulate, these would have to be individually tailored to fit the fee structure of every distinct exchange.

Metric / Model	(Excess) Return	Sharpe Ratio	Drawdown Max
Baseline	-10.9264%	-0.4539	-62.0064%
Model 1	(-4.4975%)	-0.6877	-56.2268%
Model 2	(-8.6222%)	-0.9213	-52.4002%
Model 3	(31.0905%)	0.9175	-28.6381%

Table 2: Performance metrics of the trading strategy based on prior predictions for all models. The buy and hold strategy is taken as the baseline. Higher numbers are better.

5 Discussion and Future Work

LSTM neural networks are evidently profoundly effective at capturing long-term dependencies (Greff, Koutnik, Schmidhuber, Srivastava & Steunebrink, 2017). However, when the data being analyzed exhibits patterns of extreme volatility it seems that achieving good performance becomes more troublesome. While the results obtained in this paper are relatively in line with past research in terms of predictive accuracy, it must be noted that they are very fragile. During hyper-parameter optimization it became evident that even very small changes in the batch size or epochs set for training gave vastly different outcomes. This indicates that, while the results of model 3 looks promising, it should be approached with severe caution. Moreover, this paper has assumed no effects of slippage or transaction fees for testing its trading strategies. Hence, the author of this paper would advice against trading in a real-world scenario based on the methodology proposed in this paper.

Future work could explore the outcomes of a single-staged hyper-parameter optimization procedure. Due to the limited computational power available within this study it

[5] https://docs.nvidia.com/deeplearning/sdk/cudnn-developer-guide/index.html#reproducibility

was not feasible to simultaneously optimize over the number of epochs, batch size, optimizer, loss function as well as the neurons per layer. With the availability of more computational resources it would pose for an interesting question to see how the results will vary. Furthermore, it would be of interest to see how the trading strategy proposed by this paper would play out in a real-world trading scenario. As mentioned before, issues like slippage and transaction fees are not explored in this paper. They could however greatly impact the profitability of said trading strategy. Lastly, the performance obtained within this paper could be compared against different benchmarks, such as the moving average crossover strategy. Doing so would provide for an additional check of the robustness of performance.

5 Conclusion

In conclusion, this paper employed three different LSTM neural network architectures to predict the price trend of Ethereum using features related to its blockchain. The results were evaluated in two-fold. Firstly, the predictive power of the various LSTM architectures was compared against the baseline of random chance, as proposed by the Efficient Market Hypothesis. Secondly, a trading strategy derived from these predictions was employed out-of-sample. The results show how it can be possible to predict the price trend of Ethereum with a greater than random chance, although this conclusion should be taken with caution. Additionally, the trading strategy resulting from these predictions proved to be more profitable than the baseline strategy of buying and holding. It should be noted here that the cumulative returns of the baseline strategy during the out-of-sample period were not above zero. More research into the same domain is required to thoroughly substantiate the claims put forth by this paper.

References

Alessandretti, L., Aiello, L. M., Baronchelli, A., & Elbahrawy, A. (2018). Machine Learning the Cryptocurrency Market. *Available at SSRN 3183792*.

Berkowitz, J., Elkan, C., & Lipton, Z. C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Bilanakos, C., Georgoula, I., Giaglis, G. M., Pournarakis, D., & Sotiropoulos, D. N. (2015). Using Time-Series and Sentiment Analysis to Detect the Determinants of Bitcoin Prices. *MCIS 2015 Proceedings*.

Caton, S., McNally, S., & Roche, J. (2018). Predicting the Price of Bitcoin using Machine Learning. *26th Euromicro Internation Conference on Parallel, Distributed and Network-based processing*, 339-343.

Chen, M., Narwal, N., & Schultz, M. (2017). Predicting Price Changes in Ethereum. Retrieved from http://cs229.stanford.edu/proj2017/final-reports/5244039.pdf

Cibikdiken, A. O., & Karakoyun, E. S. (2018). Comparison of ARIMA Time Series Model and LSTM Deep Learning Algorithm for Bitcoin Price Forecasting. *Proceedings of MAC 2018 in Prague*, 171-180.

Greff, K., Koutnik, J., Schmidhuber, J., Srivastava R. K., & Steunebrink, B. R. (2017). LSTM: A search space odyssey. *Transactions on neural networks and learning systems*, 28(10), 2222-2232.

Daflapurkar, P. M., Hule, S., Nirwan, E., Rao, A., Shaikh, H. (2015). Survey: Stock Market Prediction Using Statistical Computational Methodologies and Artificial Neural Networks. *International Research Journal of Engineering and Technology*, 2(8), 99-102.

Fama, E. F. (1965). The behavior of stock-market prices. *The Journal of Business*, 38(1), 34-105.

Garcia, D., & Schweitzer, F. (2015). Social signals and algorithmic trading of Bitcoin. *Royal Society open science*, 2(9), 150288.

Gunasekarage, A., & Power, D. M. (2001). The profitability of moving average trading rules in South Asian stock markets. *Emerging Markets Review*, 2(1), 17-33.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8), 2554-2558.

Huang, N. E., Long, S. R., Peng, C. K., & Wu, Z. (2007). On the trend, detrending, and variability of nonlinear and nonstationary time series. *Proceedings of the National Academy of Sciences*, *104*(38), 14889-14894.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

Pichl, L., & Kaizoji, T. (2017). Volatility Analysis of Bitcoin Price Time Series, *Quantitative Finance and Economics*, 1(4), 474-485.