A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting

Peter T Yamak
Beijing University of Technology
100 Pingleyuan, Chaoyang District
Beijing 100124 P.R. China
peteryamak@emails.bjut.edu.cn

Li Yujian†
School of Artificial Intelligence
Guilin University of Electronic Technology
Guilin 541004, Guangxi
liyujian@guet.edu.cn

Pius K Gadosey
Beijing University of Technology
100 Pingleyuan, Chaoyang District
Beijing 100124 P.R China
kwaogad@emails.bjut.edu.cn

ABSTRACT

A critical area of machine learning is Time Series forecasting, as various forecasting problems contain a time component. A series of observations taken chronologically in time is known as a Time Series. In this research, however, we aim to compare three different machine learning models in making a time series forecast. We are going to use the Bitcoin's price dataset as our time series data set and make predictions accordingly. The results show that the ARIMA model gave better results than the deep learning-based regression models. ARIMA gives the best results at 2.76% and 302.53 for MAPE and RMSE respectively. The Gated Recurrent Unit (GRU) however performed better than the Long Short-term Memory (LSTM), with 3.97% and 381.34 of MAPE and RMSE respectively.

CSS CONCEPTS

• Computing methodologies → Machine learning → Machine learning approaches → Neural networks • Mathematics of computing → Probability and statistics → Statistical paradigms → Time series analysis

KEYWORDS

Time Series, Bitcoin, ARIMA, LSTM, GRU

ACM Reference format:

Peter T Yamak, Li Yujian and Pius K Gadosey. 2019. A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting. In *Proceedings of 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence (ACAI' 19). Sanya, China, 7 pages.* https://doi.org/10.1145/3377713.3377722

†Corresponding author: liyujian@guet.edu.cn

ACAI '19, December 20–22, 2019, Sanya, China

ACM ISBN 978-1-4503-7261-9/19/12…$15.00
https://doi.org/10.1145/3377713.3377722

1 Introduction

A critical area of machine learning is Time Series forecasting, as various forecasting problems contain a time component. It is one of the most applied data science technique in business, supply chain management, and production.

A series of observations taken chronologically in time is known as a Time Series. A time-series dataset is different from other datasets, as it adds a time element. This added element is both a limitation and a structure that offers an additional source of information.

In this research, however, we aim to compare three different machine learning models in making a time series forecast. We are going to use the Bitcoin's price dataset as our time series data set and make predictions accordingly. We are going to use a one-day interval exchange rate in American Dollars (USD) from 28th November 2014 to 5th June 2019.

Bitcoin is the foremost distributed cryptocurrency, while other digital currencies are created by either cloning or adjusting the mechanism of Bitcoin [1]. Transactions are safe and authentic when they are controlled by cryptography stored in Blockchain. These transactions are stored in a decentralized network [2]. In 2017, Bitcoin was measured by market capitalization as a leading force in the cryptocurrency market, its accounts occupying 72% of the total cryptocurrency market. As at December 2017, the prices of Bitcoin had increased from 1,000 dollars to 16, 000 dollars. This rampant changes in rates make it highly challenging to predict Bitcoin prices

2 Related Work

In this section, we define the background and significant previous works that motivated our work.

2.1 Autoregressive Integrated Moving Average (ARIMA)

The Auto-Regressive Integrated Moving Average model is used in fitting time series data, to help in getting a better understanding of the data or to make predictions [3]. Autoregression uses

observations from previous time steps to forecast the value for the next time step, using the regression equation.

The Integrated uses differencing of raw observations to make the series stationary. Differencing is the method for transforming a non-stationary time series into a stationary one. The process is done by subtracting an observation value from a previous observation value. The process is repeated until a stable series is achieved.

The Moving Average calculates the simple average in a particular time frame and divides it with the total number of time frames taken. A pure Auto-Regressive model is where X_t depends only on its lags [4].

$$X_t = \alpha + \beta_1 X_{t-1} + \beta_2 X_{t-2} + \cdots + \beta_p X_{t-p} + \epsilon_1$$

Where X_t is the lag_1 of the series, β is the coefficient of lag_1 that the model estimates and α is the intercept term. A pure Moving average is a model, however, is one where the X_t depends on the lagged forecast errors.

$$X_t = \alpha + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

Where the error terms are the inaccuracies of the autoregressive models of the respective lags. ε_t and ε_{t-1} are the errors from the equations:

$$X_t = \beta_1 X_{t-1} + \beta_2 X_{t-2} + \cdots + \beta_0 X_0 + \epsilon_t$$
$$X_{t-1} = \beta_1 X_{t-2} + \beta_2 X_{t-3} + \cdots + \beta_0 X_0 + \epsilon_{t-1}$$

When we combine the Auto-Regressive and Moving Average model, we get the following equation:

$$X_t = \alpha + \beta_1 X_{t-1} + \beta_2 X_{t-2} + \cdots + \beta_p X_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

This equation is the ARIMA equation. The ARIMA model is one where the time series was differenced at least once to make it stationary. ARIMA (p, d, q) is the standard notation used; the bracketed parameters are substituted with integer values; this shows the specific ARIMA model used. The following are the definition of the settings:

p: refers to the number of lag observations included in the model, this helps to adjust the line that is being fitted to predict the series

d: refers to the quantity of differencing transformations needed by the time series to get stationary.

q: refers to the size of the moving average window.

Our data is sorted by time and daily latest price of the Bitcoin. This kind of data is called the Time Series. Time-Series data might have the following components [5]:

Trend: This exists when a series increases, decreases or remains at a constant for time.

Seasonality: This component displays periodical patterns that repeat at a constant frequency (*f*). If for instance for a monthly period, $f = 12$, this means that the periodical pattern repeats every 12 months.

Stationarity: This component refers to a time series that has a constant mean and variance. The covariance is also independent of time.

2.1.1 *Bitcoin price prediction: An ARIMA approach*. Amin Azari [6] research aims at illuminating the effectiveness of autoregressive integrative moving average (ARIMA) model in forecasting the price of bitcoin by analyzing Bitcoin prices over three years.

These experimental studies reveal that this simple scheme is competent in sub-periods in which the behavior of the time-series is nearly unaffected; this mainly happens when it is used for short-term prediction. There was, however, a different result when the ARIMA model was trained over the three years period, during which prices had experienced different behaviors.

There was a significant prediction error when the ARIMA model tried to make a long-term prediction. The ARIMA model was unable to capture the sharp instabilities in the Bitcoin price. The study further throws more light on how the interaction of the prediction accuracy, the window size w, and the choice of (p, q, d) interact.

2.1.2 *Online ARIMA Algorithms for Time Series Prediction.* C. Liu et al. [7] aim to solve the problem of inefficiency in the application of Autoregressive integrated moving average (ARIMA), in its handling of real large-scale data. They proposed an online learning algorithms for estimating ARIMA models under relaxed assumptions on the noise terms, this they believed would be suitable for a broader array of applications. It would also enjoy a high computational efficiency.

They reframed the ARIMA model into a task of full information online. This new model was to enable them to make efficient and scalable estimations of the parameters online. They also analyzed the regret bounds of the proposed algorithms, which guarantee that their online ARIMA model can be proven as useful as the best ARIMA model in hindsight. Their results confirmed the effectiveness and robustness of the proposed method.

2.2 Long Short-Term Memory (LSTM)

Long Short-term Memory (LSTM) networks are a distinct type of Recurrent Neural Networks. They are able to learn long-term dependencies and are very popular for working with sequential data such as time-series data. LSTM's address the vanishing gradient problem [8]. When the time step is significant, the gradient gets too small or large, which leads to a vanishing gradient problem. This problem occurs while the optimizer back propagates, and makes the algorithm run, while the weights almost do not change at all [6].

The long-short-term memory cell uses an input gate; a forget gate and an output gate (a simple multilayer perceptron). These gates define whether the data can pass through or not depending on the data's priority. The gates also enable the network to learn what to save, what to forget, what to remember, what to pay attention, and what to output. The cell state and hidden state are used in gathering data for processing in the next state. The vanishing gradient can, therefore, be protected. Figure 1 shows the structure of the nodes of the LSTM cell.

The gates have the following equations:

- Input Gate: $i_t = \sigma(W_i h_{t-1} + W_i h_t)$
- Forget Gate: $f_t = \sigma(W_f h_{t-1} + W_f h_t)$
- Output Gate: $o_t = \sigma(W_o h_{t-1} + W_o h_t)$

- Intermediate Cell State: $\tilde{C} = tanh(W_c h_{t-1} + W_c h_t)$
- Cell State (next memory input): $c_t = (i_t * \tilde{C}) + (f_t * c_{t-1})$
- New State: $h_t = o_t * tanh(c_t)$
- $\boldsymbol{X_t}$ Input Vector
- $\boldsymbol{h_t}$ Output Vector
- ***W, U,*** and ***f*** Parameter matrices and vector

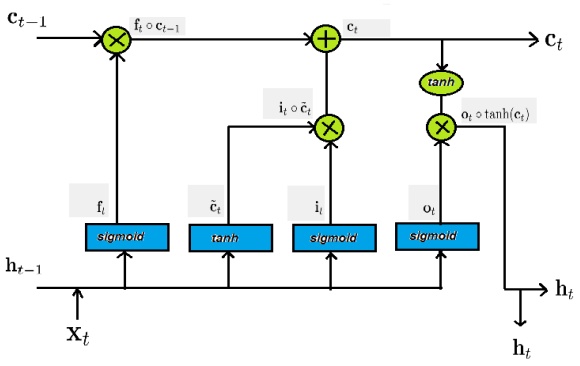


Figure 1: Diagram of an LSTM cell

⊗ Element-wise multiplication

⊕ Element-wise summation / concatenation

2.2.1 *Predicting the Price of Bitcoin using Machine learning.* McNally et al. [9] researched into the accuracy the track of Bitcoin price prediction in American dollars (USD). The task was accomplished with varying degrees of success through the implementation of a Bayesian optimized recurrent neural network (RNN) and a Long Short Term Memory (LSTM) network.

They generated the models on the features from CoinDesk and hash rate data. Their results revealed that the LSTM achieves the maximum classification accuracy of 52% and an RMSE of 8%. The non-linear deep learning methods performed better than the ARIMA forecast.

2.2.2 *Bitcoin Price Prediction with Neural Network.* Kejsi et al. [10] research aimed at developing a better understanding of bitcoin price influencers and its general vision. The investigation outlined the results of predicting Bitcoin price for 30 and 60 days ahead. It also included implementing hyper-parameter tuning to get a more accurate network architecture of the LSTM model.

The results of the LSTM model showed that, during training that the higher the batch size, the worst the prediction on the test set. This result is mainly because, the more training, the more prone to overfitting the model becomes. The Mean Absolute Error function was used when using the model to predict the training and test data.

2.2.3 *EA-LSTM.* Li et al. [11] research aim to address the problem of LSTMs insufficient capacity to pay different degree of attention on a sub-window feature within multiple time-steps. They proposed an evolutionary attention-based LSTM training, with competitive random search, this is to be used for multivariate time series forecast.

By moving standard parameters, they introduced an evolutionary attention learning method to the LSTMs model. To desist from being confined into partial optimization, they proposed an evolutionary computation enthused competitive random search method, which can well configure the parameters in the attention layer.

Their results showed that the proposed model could attain competitive prediction performance compared with other methods. The model also effectively employ local information within a sampling window according to a various measure of attention distribution.

2.3 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit, just like the LSTM, is a Recurrent Neural Network. It, however, has a less complicated structure compared to LSTM. It lacks an output gate but has an update ***z*** and a reset gate ***r***. These gates are vectors which decide what information should be passed to the output. Figure 2 shows the GRU cell.

The Reset gate defines how to combine the new input with the previous memory. The definition of how much of the last memory to keep is done by the Update [12]. The GRU has the following equations:

Update gate: $z = \sigma(W_z h_{t-1} + U_z x_t)$

Reset gate: $r = \sigma(W_r h_{t-1} + U_r x_t)$

Cell state: $c = tanh(W_c(h_{t-1} * r) + U_c x_t)$

New state: $h_t = (z * c) + ((1 - z) * h_{t-1})$

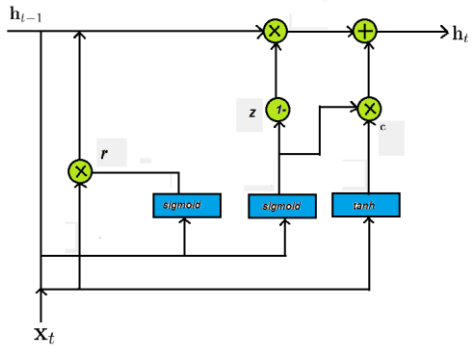


Figure 2: Diagram for GRU Cell

2.3.1 *Recurrent Neural Networks for Multivariate Time Series with Missing Values.* Z. Che et al. [13] sorted to improve prediction and other related tasks in multivariate time series data by studying the missing patterns for sufficient imputation and improving prediction performance. They developed a new deep learning model, called GRU-D, which is based on Gated Recurrent Unit (GRU).

It takes two representations of missing patterns and effectively integrates them into a deep model architecture. This process is done so that it captures the long-term temporal dependencies in time series as well as applying the missing patterns to gain better prediction results.

The results of their experiments of time series classification tasks on clinical datasets (MIMIC-III, PhysioNet) and synthetic datasets, showed that their models attain excellent performance. It also gave valuable insights for better understanding and application of missing values in time series analysis.

3 Setup and Experiments

For the model building and data prediction, we use Keras with Tensorflow as backend. We also employ Scikit-learn during the data processing phase. It was used in the execution of the Min-Max normalization. We also used Pandas and Numpy extensively. For plotting the various graphs, we used Matplotlib and Seaborn

3.1 Data Collection

For this time series forecasting, we use data from Bitcoin transaction from the Crypto data download website. The daily trading exchange data rate in dollars (USD) from 28th November 2014 to 5th June 2019 was used for this research.

The data has 1639 rows, this was split into a training and test set, with a ratio of 70:30. This split gives a training set of 1147 rows and 492 rows for the test set. The data has the following features; Date, Symbol, Open, High, Low, Close, Volume Bitcoin (BTC) and Volume Dollars (USD). The features are explained below.

Feature	**Summary**
Date	Data recorded Date
Symbol	The symbol for the transaction currency
Open	Value at which Trading Opened
High	The day's highest trade value
Low	The day's lowest trade value
Close	Latest trade
Volume BTC	The day's total trade volume in BTC
Volume USD	The day's total trade volume in USD

We then proceed to calculate the correlation between the features in the dataset. Figure 3. Shows the Heat map of features of the data. From the graph, it is evident that all the features are highly correlated. We are, however, going to fit and make predictions for the Close feature column.

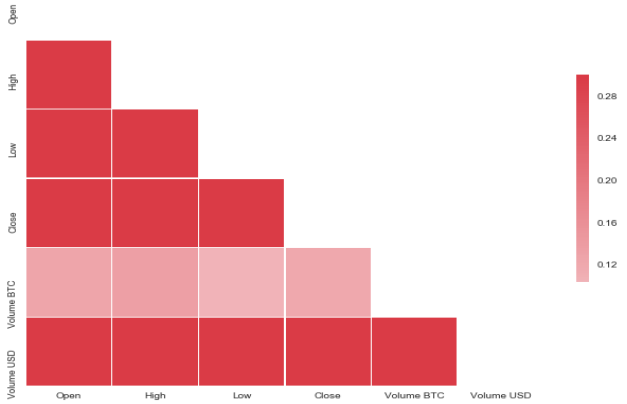


Figure 3: Heatmap correlation coefficient values for features of Dataset

3.2 Dataset Normalization

To make it easy for the network to learn, the data should take small values; mostly between the range of 0 and 1. It should also be homogeneous, meaning all the features should have values at approximately the same range [14]. We, therefore, reduced the data to a scale between 0 and 1. The process of putting the values in the same range is called the MinMax normalization. We import MinMaxScaler from Scikit-learn to do this. The equation for the Min-Max normalization is as follows:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Where ***z*** represents the normalized value and ***x*** represents the observed values in the set. ***Min*** and ***max*** are the minima and maxima values in ***x.***

3.3 Testing and Removing Stationarity

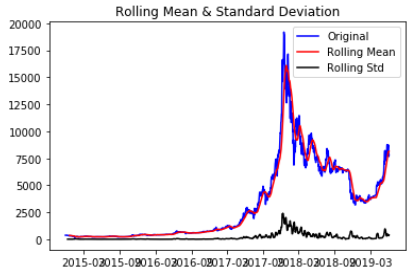
Since one of the models to be used is the Auto-Regressive Integrated Moving Average (ARIMA), we test for Stationarity in the data. We use the Augmented Dicky Fuller (ADF) test to screen for stationarity. The ADF test is a statistical test called a unit root test. The idea behind the unit test is that it finds out how strongly a time series is determined by a trend.

- Null Hypothesis (H0): The time series can be represented by a unit root that is not stationary.
- Alternative Hypothesis (H1): The time series is stationary.

Interpretation of the p-value.

- P-value > 0.05: Accepts the Null Hypothesis
- P-value < 0.05: Rejects the Null Hypothesis

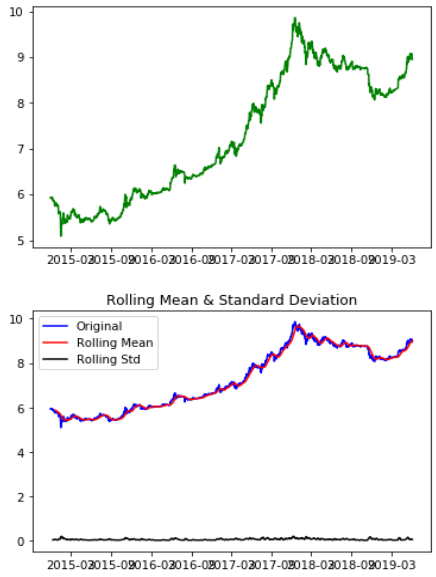
The graph plot of the ADF test is shown in Figure 4. This show that our data is nonstationary since our p-value is greater than 0.05




```
ADF Stastistic: -1.820419
p-value: 0.370372
The graph is non stationary
Critical values:
        5%: -2.863
        1%: -3.434
        10%: -2.568
```

Figure 4: ADF test for stationery in data

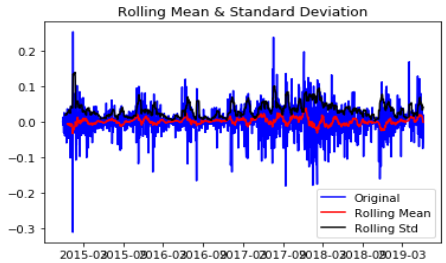
We proceed to make the data stationary by first using log transformation to un-skew the highly skewed data. This does not make the data stationary, as shown in Figure 5 below. The p-value after the log-transforming it is still higher than 0.05.




```
ADF Stastistic: -0.319842
p-value: 0.922647
The graph is non stationary
Critical values:
        5%: -2.863
        1%: -3.434
        10%: -2.568
```

Figure 5: Graph for ADF test after Log transformation of the series

The next step in making the series stationary was to trend and seasonality with differencing; this we did by subtracting the current value from the previous values. This method makes the mean stabilized and increases the chances of stationarity of the time series. Figure 6 shows the graph for the ADF test after the trend and seasonality were removed. The p-value is now less than 0.05, hence the series is now stationary, this makes the data ready to fit the ARIMA model.




```
ADF Stastistic: -41.168018
p-value: 0.000000
The graph is stationary
Critical values:
        5%: -2.863
        1%: -3.434
        10%: -2.568
```

Figure 6: Graph for ADF test after remove trend and seasonality

3.4 Fitting the Models

3.4.1 *The ARIMA Model.* We start by fitting an ARIMA model to the Bitcoin Closing price dataset and review the residual errors. We fit an ARIMA (1, 1, 0) model. This sets the lag value to 1 for autoregression. We already made our dataset stationary, w, however, chose to use a Difference order of 1. We also wanted a moving average model of 0.

After the model has been fit to the training dataset, we proceed to make predictions and compare the results to the test set. We calculate the residual error for each prediction made and even find the final Mean Absolute Percentage Error score (MAPE) and the root-mean-square error score (RMSE).

3.4.2 *The LSTM Model.* Our LSTM model in Keras has the following parameter:

Parameters	**Value**
epochs	400
Batch size	170
Dense	1
Input shape	1
optimizer	adam

After fitting the model to the training dataset, we proceed to make predictions and compare the results to the test set. Just like we did for the ARIMA model, we advance to measure the accuracy using the proposed metrics.

3.4.3 The GRU Model. Our GRU model in Keras has the following parameter:

Parameters	**Value**
epochs	400
Batch size	170
Dense	1
Input shape	1
optimizer	adam

Just like the previous models, we proceed to measure the accuracy after fitting the model to the training set.

4 Results

We used two of the most common metrics to measure the accuracy of the models, the Mean Absolute Percentage Error (MAPE) and the Root Mean Squared Error (RMSE). Table 1 and 2 shows the results for the Accuracy measurement and the time in seconds.

The results show that the ARIMA model gave better results than the deep learning-based regression models. ARIMA gives the best results at 2.76% and 302.53 for MAPE and RMSE respectively. The Gated Recurrent Unit, however, performed better than the Long Short-term Memory, with 3.97% and 381.34 of MAPE and RMSE respectively.

ARIMA also had the best time at 0.007 seconds, followed by GRU. The graph for each prediction against the actual values is shown in the Figure 7, 8 and 9 below.

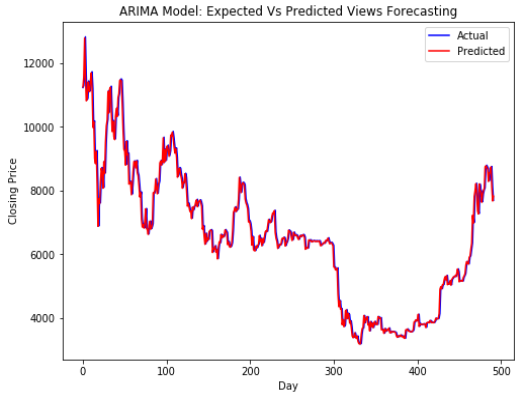


Figure 7: Graph of Actual and predicted values of ARIMA Model

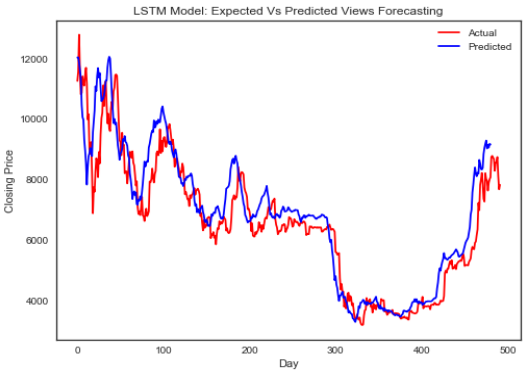


Figure 8: Graph of actual and predicted values of LSTM Model

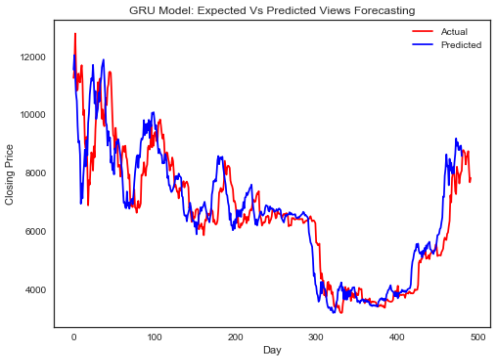


Figure 9: Graph of Actual and Predicted values for GRU Model

Table 1: MAPE and RMSE of the models

Model	MAPE	RMSE
ARIMA	2.76	302.53
LSTM	6.80	603.68
GRU	3.97	381.34

Table 2: Time of implemented models

MODEL	TIME (sec)
ARIMA	0.0007
LSTM	0.3267
GRU	0.2296

5 Conclusion

From the results, it shows that the ARIMA Model gives the best accuracy and time. This outcome might also be as a result of several factors. The parameters that are chosen and the total amount of data can also affect the results. The amount of data we have for this paper is relatively small. RNN typically performs on more massive datasets, as previous research papers have shown.

The selected feature selected might not be sufficient to predict the Bitcoin prices accurately. Several factors contribute to the increase or decrease in prices; Country policies and social media reactions [10], have been known to affect the prices. Adding other features might help with the performance of the models.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under grant 61876010, and the Research Fund for the Doctoral Program of Higher Education under grant 20121103110029.

REFERENCES

[1] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-peer Electronic Cash System.

[2] Garrick Hileman and Michel Rauchs. 2017. Global Cryptocurrency Benchmarking Study. Cambridge Centre for Alternative Finance.

[3] James Chen. 2019. Advanced Technical Analysis Concepts: Autoregressive Integrated Moving Average (ARIMA). Retrieved from https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp

[4] Selva Prabhakaran. 2019. Machine Learning Plus: Guide to Time Series Forecasting in Python. Retrieved from https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python.

[5] Roman J. H. Torres. 2018. 7 Ways Time Series Forecasting Differs from Machine Learning. Retrieved from https://www.datascience.com/blog/time-series-forecasting-machine-learning-differences

[6] Amin Azari. 2018. Bitcoin Price Prediction: An ARIMA Approach. KTH Royal Institute of Technology.

[7] Chenghao Liu, Steven C. H. Hoi, Peilin Zhao and Jianling Sun. 2016. Online ARIMA Algorithms for Time Series Prediction. In Proceeding of the 13th AAAI Conference on Artificial Intelligence (AAAI-16).

[8] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long Short-term Memory. Neural Computation vol. 9 no. 8. DOI: https://doi.org/10.1162/neco.1997.9.8.1735

[9] Sean McNally, Jason Roche and Simon Caton. 2018. Predicting the Price of Bitcoin Using Machine Learning. 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP). DOI: https://doi.org/10.1109/PDP2018.2018.00060

[10] Kejsi Struga and Olti Qirici. 2018. Bitcoin Price Prediction with Neural Network. In Proceeding of the 3rd International Conference of the Recent Trends and Applications in Computer Sciences and Information Technology.

[11] Youru Li, Zhenfeng Zhu, Deqiang Kong, Hua Han and Yao Zhao. 2018. EA-LSTM: Evolutionary Attention-based LSTM for Time Series Prediction. Knowledge-Based Systems. DOI: https://doi.org/10.1016/j.knowsys.2019.05.028

[12] Antonio Gulli and Sukit Pal. 2017. Deep Learning with Keras. Packt Publishing.

[13] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag and Yan Liu. 2018. Recurrent Neural Networks For Multivariate Time Series With Missing Values. Scientific Reports 8, 6085. DOI: https://doi.org/10.1038/s41598-018-24271-9

[14] Aurelien Geron. 2017. Hands-on Machine Learning with Sci-kit-learn and Tensorflow. Retrieved from https://www.oreilly.com/library/view/hands-on-machine-learning/9781491962282