Applied Soft Computing 110 (2021) 107738

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Prediction of Bitcoin price based on manipulating distribution strategy

Eunho Koo [a], Geonwoo Kim [b,*]

[a] Center for Mathematical Analysis & Computation, Yonsei University, Seoul 120-749, Republic of Korea
[b] School of Liberal Arts, Seoul National University of Science and Technology, Seoul 01811, Republic of Korea

ARTICLE INFO

Article history:
Received 1 December 2020
Received in revised form 31 May 2021
Accepted 17 July 2021
Available online 26 July 2021

Keywords:
Bitcoin
Prediction
Artificial neural networks
Copula
Distribution manipulation

ABSTRACT

Since Bitcoin has been the most popular digital currency in the global financial market, the prediction of its price has been an important area in finance. Recently, numerous researches on prediction of financial indices including Bitcoin based on machine learning techniques have been developed. However, little studies pay attention to the strategy of manipulating original distribution to obtain better performances. Because the return data of Bitcoin prices are highly concentrated near zero, small changes of values in the components can cause different results. Based on this characteristic, we propose flattening distribution strategy (FDS) based on the copula theory as a strategy of the manipulating distribution of components artificially to improve the prediction of Bitcoin price return. We consider multilayer perceptron (MLP), recurrent neural networks (RNN), and long short-term memory (LSTM) to assess the performances of FDS. Finally, we find that the proposed algorithms based on FDS improve significantly the prediction accuracy of the return of Bitcoin price for each of the three architectures.

1. Introduction

It is well known that financial markets have the characteristics of complex systems because there are numerous interacting elements in the markets and nonlinear features. To understand the characteristics of the market and analyze it, there have been a lot of studies with financial time series data. Among them, prediction of financial data is the most popular topic because it is very practical for real investors and useful for analyzing the financial market. However, this is a difficult problem due to various influences such as economic environment, political policy, asymmetric and incomplete information and natural factors. In order to improve the prediction of financial time series with the nonlinear nature and complexity, the machine learning techniques have been used widely in recent years. This paper also focuses on the development of time series prediction based on the machine learning techniques. More specifically, we deal with Bitcoin data as financial time series. That is, the aim of this paper is to provide a new model for better prediction of Bitcoin price.

Bitcoin introduced by Nakamoto [1] is receiving much attention by investors and researchers and becoming the most popular and valuable cryptocurrency in the financial market. According to Iwamura et al. [2] and Yermack [3], Bitcoin has high volatility. In general, the high volatility of an investment means higher potential returns. Thus, Bitcoin has been often used as a speculative investment, and its popularity is increasing continuously as an investment vehicle [4]. Since all investors want to have high returns and the accurate prediction helps them, a more exact prediction of Bitcoin is very important to the investors. In this sense, the prediction of Bitcoin has been studied by many researchers. The accuracy of the prediction has been improved with various approaches in the literature review of the next section. In particular, the models based on the machine learning techniques have been considered widely to capture nonlinear features. However, despite machine learning technique using strategy for manipulating distribution is a suitable approach to predict financial time series, the approach has not yet been considered for the prediction of Bitcoin. Motivated by this, we propose a new approach by combining the existing machine learning algorithms and flattening distribution strategy (FDS) based on the copula theory. FDS is an approach based on transforming of the distribution of raw data into uniform distribution. Because of the property of the uniform distribution, there does not exist significant difference between outliers and non-outliers. That is, it can be argued that FDS is relevant to the works which deal with the noises and uncertainties in the real data [5–8].

This paper is organized as follows. Section 2 reviews the literatures on studies of Bitcoin and the prediction of financial time series based on the machine learning techniques. Section 3 presents the methodologies used in this paper. Section 4 describes the data used for this work and introduce our new algorithms based on MLP, RNN and LSTM. Section 5 provides the experiment results with several figures and tables by using the proposed algorithms. Section 6 concludes the paper.

* Corresponding author.
E-mail address: geonwoo@seoultech.ac.kr (G. Kim).

https://doi.org/10.1016/j.asoc.2021.107738
1568-4946/

2. Literature review

Many researchers have studied the prediction of stock prices or stock indices using artificial neural networks (ANNs) such as multi-layer perceptron (MLP), recurrent neural networks (RNNs), long-short term memory (LSTM), and so on. ANNs have been often utilized to predict time series of stock market with the back propagation neural networks (BPNNs). Kara et al. [9] improved the prediction of financial data in the Istanbul stock market using ANN model with ten technical indicators. Wang et al. [10] proposed a new approach for predicting Shanghai Composite Index. They developed an efficient method using the Wavelet De-noising-based Back Propagation (WDBP) neural network. Devadoss and Ligori [11] dealt with the prediction problem of stock price using ANN with MLP. Qui and Song [12] optimized the ANN using genetic algorithms (GA) and predicted efficiently the direction of the daily Japanese stock market index. Qui et al. [13] constructed a hybrid model based on GA and simulated annealing (SA) to improve prediction accuracy. RNN first proposed by Elman [14] is useful in time series prediction because RNN remembers each and every information through time. In [15], the RNN was used to predict the prices of the three stock, and authors showed the efficiency of the RNN in predicting time series with nonlinear behavior in the financial market. RNN has been developed for better efficiency. In particular, LSTM introduced by Hochreiter and Schmidhuber [16] has become one of the most popular RNN approaches for the prediction of time series. Chen et al. [17] studied the prediction of China stock return using LSTM. They showed that the LSTM outperforms the random method in stock market prediction. Fischer and Krauss [18] found LSTM to outperform memory-free classification methods and demonstrated that LSTM provides meaningful information from nonlinear financial time series. Baek and Kim [19] proposed a new modularized forecasting framework, which has two LSTM modules (an overfitting prevention LSTM module and a prediction LSTM module), to improve the forecasting accuracy. Kim and Kim [20] also constructed a hybrid model combining LSTM and convolutional neural network (CNN) to extract temporal features and image features and showed that the proposed model provides an efficient prediction for financial data. In addition, Bukhari et al. [21] developed LSTM using autoregressive fractional integrated moving average (ARFIMA) model to minimize the volatility problem and to overcome the over fitting in the prediction problem. U et al. [22] proposed a novel LSTM prediction model by using reversal point features including combinations of new defined candlestick indicators and technical indicators. Furthermore, these machine learning techniques have been applied the digital currency market, which is represented by Bitcoin, as well as the prediction of stock market. In this paper, we focus on the prediction of Bitcoin based on the machine learning techniques.

Since the market capitalization of cryptocurrencies continues to grow and Bitcoin holds consistently the top spot when it comes to overall market capitalization, the study on Bitcoin is one of the most interesting topics in the field of cryptocurrency. In fact, there are many kinds of cryptocurrencies, but Bitcoin has become the most successful because the Bitcoin market capitalization is more than 40% of the total digital currency market. For this reason, Bitcoin has been studied by a lot of researchers. Some researchers investigated the properties of Bitcoin such as a multifractal spectra, chaos, randomness, long-range memory in volatility and multifractal behavior [23–27]. GARCH family models have been widely used for the analysis and prediction of Bitcoin volatility [28–31], and there have been the studies on efficient market hypothesis (EMH) of Fama [32] for the Bitcoin market [33–43]. Furthermore, the machine learning techniques have been used for the improvement of Bitcoin price prediction in recent years. Atsalakis et al. [44] predicted the trend in the price of Bitcoin using a hybrid neuro-fuzzy model. They showed that the proposed model has better performances compared to other models. Lahmiri and Bekiros [45] implemented deep learning LSTM neural networks in cryptocurrency prediction and compared to the generalized regression neural architecture. They found the superiority of LSTM for predicting of digital currencies. Mallqui et al. [46] proposed machine learning ensemble algorithms to predict the Bitcoin price direction. They compared their models with the state-of-the-art studies and found that the proposed methodology shows an improvement in accuracy. Lahmiri and Bekiros [47] first adopted statistical machine learning approaches to predict Bitcoin price. Specifically, they used support vector regression (SVR), gaussian Poisson regression (GRP), regression tree (RT), the k-nearest neighbors (k NN) algorithm, feedforward neural network (FFNN), bayesian regularization neural network (BRNN), and radial basis function neural network (RBFNN) for an accurate prediction of Bitcoin price and showed that the BRNN provides an outstanding accuracy in prediction. Alonso et al. [48] predicted the prices of six cryptocurrencies including Bitcoin using four network architectures (CNN, hybrid CNN–LSTM network, MLP, and radial basis function neural network) and found that LSTM combined with convolutional layers provides the best results. This paper also contributes on development of machine learning technique to improve the accuracy of the prediction of Bitcoin price. We employ MLP, RNN and LSTM for prediction and develop each model to enhance the accuracy of the approaches.

3. Methodology

3.1. Structures of MLP, RNN and LSTM

MLP (cf. FFNN) has been widely used both for classification problems and time-series predictions [14,49]. MLP supports the concepts of feeding forward and propagating back, which are the essential structures of ANN. RNN is an enhanced model of the conventional MLP, suitable for time-series prediction [50]. LSTM is a type of RNN that was proposed to solve the vanishing and diverging gradient issue of conventional RNN [16]. The MLP is one of the most popular techniques for tackling classification and prediction problems. MLP consists of an input layer, hidden layers, and an output layer, and the net flow is in the direction of the input to the output layer. It is necessary to determine weights that are located between layers such that errors between the prediction and the label (the observed values) can be as small as possible. We set the total number of layers to L, the dimensions for each layer to n_i for all $i = 1, 2, \ldots, L$ with $n_L = 1$ for the time-series prediction, and the error function to the usual square error function defined by $E = (a_L - d)^2/2$ where d denotes the label. Feed forwarding can be calculated as $z_{i+1} = \theta_i a_i + b_i$ and $a_{i+1} = \sigma(z_{i+1})$ where θ_i, b_i denote weights and biases, respectively, between the ith and $i+1$th layers, and the size of the weight matrix θ_i and bias b_i are $n_{i+1} \times n_i$ and n_{i+1}, respectively for $i = 1, 2, \ldots, L-1$. σ denotes the activation function, and the sigmoid function $1/(1 + e^{-x})$ and the rectified linear unit (ReLU [51]; If $x > 0$, x, otherwise 0.) are widely employed. Back-propagation using the gradient descent can be formulated as

$$\delta_L = (a_L - d) * \sigma'(z_L),\ \ \delta_i = (\theta_i^T(a_L - d)) * \sigma'(z_L)$$
$$for\ \ i = L-1, L-2, \ldots, 1,$$
$$\frac{\partial E}{\partial \theta_i} = \delta_{i+1} a_i^T,\ \ \frac{\partial E}{\partial b_i} = \delta_{i+1}$$

and eventually

$$\theta_i \leftarrow \theta_i - \alpha * \frac{\partial E}{\partial \theta_i}, \quad b_i \leftarrow b_i - \alpha * \frac{\partial E}{\partial b_i} \ \ for \ \ i = 1, 2, \ldots, L-1$$

where θ_i^T, $\sigma'(x)$,'$*$', α, and '$\leftarrow$' denote the transpose of matrix θ_i, the differentiation of $\sigma(x)$, the element-wise product of two vectors, the learning rate, and the updating of weights, respectively. RNN has been widely used since the back-propagation through time (BPTT) [52] was developed. Let $\mathbf{X} = \{x_t = (x_{1,t}, x_{2,t}, \ldots, x_{n,t}) | t = 1, 2, \ldots, T\}$ be a set of n-dimensional vectors, $\mathbf{H} = \{h_t = (h_{1,t}, h_{2,t}, \ldots, h_{s,t}) | t = 1, 2, \ldots, T\}$ be a set of s-dimensional vectors, and $\mathbf{O} = \{o_t = (o_{1,t}, o_{2,t}, \ldots, o_{m,t}) | t = 1, 2, \ldots, T\}$ be a set of m-dimensional vectors such that X, H and O are used as the input, hidden, and output data, respectively. We treat all vectors as column vectors. If $\mathbf{V}$, $\mathbf{U}$ and $\mathbf{W}$ are weight matrices of sizes $s \times n$, $s \times s$ and $m \times s$, respectively, then feed forwarding in terms of $\mathbf{X}$, $\mathbf{H}$, $\mathbf{O}$, $\mathbf{V}$, $\mathbf{U}$ and $\mathbf{W}$ is given as $y_t = \sigma(\mathbf{V}x_t + \mathbf{U}y_{t-1})$ and $o_t = \sigma(\mathbf{W}y_t)$ for all $t = 1, 2, \ldots, T$, where $y_0 = 0$. We set $\mathbf{V}$, which is located at each part between input and output layers, to the same matrix for each depth. It follows that this property is also true for $\mathbf{U}$ and $\mathbf{W}$. If we simply use square error E defined by $E = \sum_{t=1}^{T}(o_t - d_t)^2/2$, where d_t is the observed value at time t, and weights $\mathbf{V}$, $\mathbf{U}$ and $\mathbf{W}$ must be updated to minimize the error E, then we define A_i and B_i for $i = 1, 2, \ldots, D$ for the purpose of BPTT, where D denotes the total depth of the unfolded version of RNN.

$$A_i = (\mathbf{W}^T(o_i - d_i)) * \sigma'(\mathbf{W}y_i), \quad i = 1, 2, \ldots, D \tag{1}$$

$$B_D = A_D, B_i = (\mathbf{U}^T B_{i+1}) * \sigma'(\mathbf{V}x_{i+1} + \mathbf{U}y_i) + A_i, \quad i = D-1, D-2, \ldots, 1. \tag{2}$$

Here, x_i, y_i, o_i and d_i are the input, hidden, output, and observed data, respectively, which would be placed at the ith depth of the unfolded version of RNN. Note that B_j is defined in the reverse order: B_D is defined first, followed by B_{D-1} and finally B_1. This is because B_j appears in BPTT. The following weights update formula can be obtained by

$$\mathbf{V} \leftarrow \mathbf{V} - \alpha * \frac{\partial E}{\partial \mathbf{V}}, \ \mathbf{U} \leftarrow \mathbf{U} - \alpha * \frac{\partial E}{\partial \mathbf{U}}, \ \mathbf{W} \leftarrow \mathbf{W} - \alpha * \frac{\partial E}{\partial \mathbf{W}}, \tag{3}$$

where

$$\frac{\partial E}{\partial \mathbf{V}} = \sum_{i=1}^{D} B_i x_i^T, \quad \frac{\partial E}{\partial \mathbf{U}} = \sum_{i=1}^{D-1} B_i y_i^T,$$

$$\frac{\partial E}{\partial \mathbf{W}} = \sum_{i=1}^{D}((o_i - d_i) * \sigma'(\mathbf{W}y_i))y_i^T. \tag{4}$$

It is worth noting that the approximated total number of parameters of RNN is given by $ns + s^2$.

LSTM has a similar structure to the traditional RNN, but it is an architecture that complement the hidden layer in RNN and can use information that have a longer past. Hence, we present a brief description of structures of the hidden layers in LSTM. In LSTM, a state called as 'cell' (denoted by c in (9)) goes forward with the hidden layer. The operations of the hidden layer and cell in LSTM can be formulated as

$$i_i = \sigma(V^1 x_i + U^1 y_{i-1}), \tag{5}$$

$$f_i = \sigma(V^2 x_i + U^2 y_{i-1}), \tag{6}$$

$$o_i = \sigma(V^3 x_i + U^3 y_{i-1}), \tag{7}$$

$$g_i = \tanh(V^4 x_i + U^4 y_{i-1}), \tag{8}$$

$$c_i = c_{i-1} * f_i + g_i * i_i, \tag{9}$$

$$y_i = \tanh(c_i) * o_i, \tag{10}$$

where x_i, y_{i-1} denote current input data, previous hidden nodes and σ, tanh denote the sigmoid function and tangent hyperbolic function, respectively. The names of the variables i, f, o and g represent the input, forget, output, and write gates, respectively. The plus term in (9) solves the vanishing gradient problem when the activation function is used as the sigmoid and tangent hyperbolic functions. In this study, we attach a fully connected net at the final hidden layer of LSTM (size of weight between nodes is $1 \times m$) so that the dimension of the final prediction can be one. It is worth noting that the approximated total number of parameters is $4(nH + H^2)$, where V^i and U^i are of the shape $H \times n$ for each $i = 1, 2, 3, 4$. Implementation models are illustrated in Fig. 1.

3.2. Methodology

The training procedure in artificial neural networks (ANN) architectures including MLP, RNN, and LSTM aims to determine the proper values of weights located between layers to minimize the averaged differences between the predicted and observed values. To accomplish the goal, numerous researches on the activation function [53] and initialization of weights [54] have been conducted, and these trials can be considered as the studies on artificially controlling the distribution of values in the nodes (or weights) inside the architecture of ANN to obtain better results. Meanwhile, many methodologies have also been applied and tried to make accurate predictions on the return of stock prices (signed relative magnitude of the current stock price against previous day) by overcoming its intractability (such as the randomness and skewness). Copula [55], which is one of the methodologies, improves the efficiency of calculation by mapping the distribution of the return data to the uniform distribution using the cumulative distribution function (CDF) of the data [56]. In addition, the simplification strategy on the distribution enables a meaningful analysis in the study of the stock market crisis [57]. In this study, we suggest the FDS as a methodology based on the copula for artificially manipulating the distribution of values in the nodes within the ANN architectures. Our suggestion is motivated by the trait of the return data, which is noisy and extremely concentrated near zero. This aspect may cause instability, that is, small changes of the weight values in networks can cause totally incorrect results, therefore, it is necessary to mitigate the concentration characteristics of the return. Note that two approaches are applicable to predict the return, that is, the classification and regression. Since this study is motivated by the idea of manipulating the distribution of the return data, it is essential to trace the change of the distribution continuously. We select the regression approach to trace the change.

We formally describe the FDS based on the copula methodology. Let X and Y be two random variables such that $X \sim f_X(y) = F_X'(x)$ and $Y \sim f_Y(y) = F_Y'(y)$, i.e. $f_X(x), f_Y(y), F_X(x)$ and $F_Y(y)$ are probability density functions (PDFs) and CDFs of X, Y, respectively. The notation '$X \sim f_X(x)$' implies that the function $f_X(x)$ is the PDF of the random variable X. And let $g(x)$ be a function such that $Y = g(X)$. Direct calculation yields

$$F_Y(y) = P(Y < y) = P(g(X) < y) = P(X < g^{-1}(y)) = F_X(g^{-1}(y)) \tag{11}$$

where g is an increasing function. Differentiation on both side of (11) yields

$$f_Y(y) = F_Y'(y) = \frac{d}{dy}F_X(g^{-1}(y)) = f_X(g^{-1}(y))\frac{d}{dy}g^{-1}(y). \tag{12}$$

In general, it is a challenging problem to obtain the function g explicitly which satisfies $Y = g(X)$ and formula (12) for the arbitrarily given two random variables X and Y. However, if X is uniformly distributed on $[a, b]$, (12) can be expressed simply as

$$f_Y(y) = \frac{1}{b-a}\frac{d}{dy}g^{-1}(y). \tag{13}$$

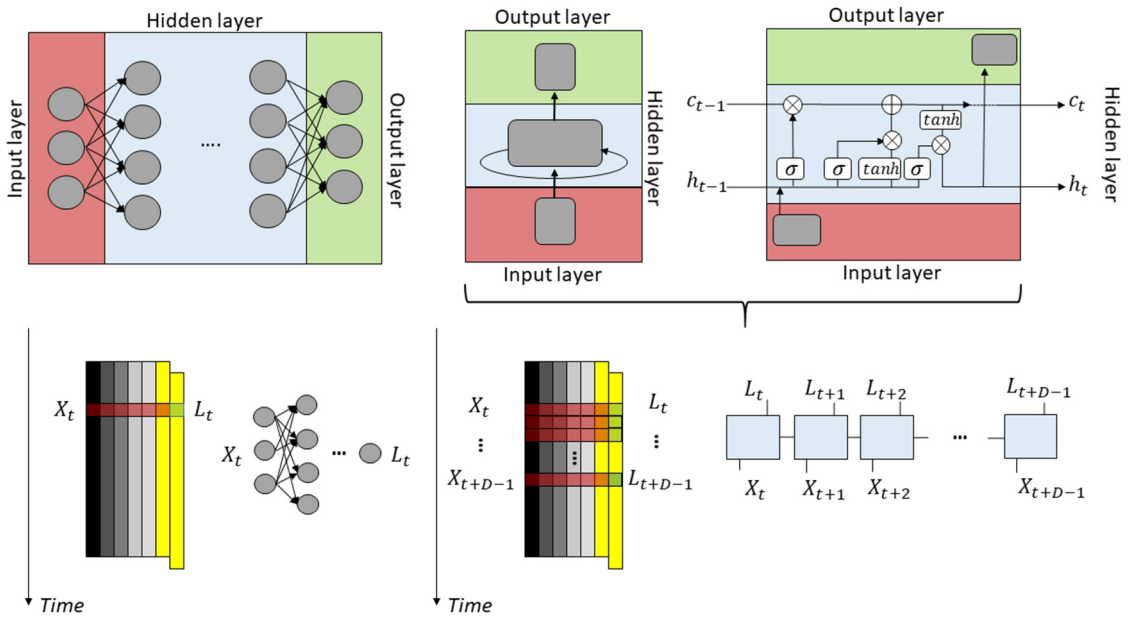

Fig. 1. Illustration of implementation models. Schematic diagram for MLP (below left) and recurrent nets (RNN, LSTM; below right) are presented.

By integrating on both side of (13), we have

$$g(y) = ((b-a)F_Y(y) + c)^{-1}, \tag{14}$$

where c is an integrating constant. In other words, if X is uniformly distributed, the function g which satisfies $Y = g(X)$ has the form of the inverse function of the CDF of Y.

4. Data and algorithm

In this study, we carry out three kinds of experiments to verify applicability of the FDS based on MLP, RNN and LSTM. For the experiments, we use daily Bitcoin data for the period from 28th April 2013 to 27th April 2020 and separate into two sets as training and test sets. The first 70% of data is used for training, and the remaining 30% of data is used for testing. Each dataset consists of six types of data (volume of business, market capitalization and prices of Bitcoin in terms of open, high, low, close). Here, we consider close price of Bitcoin to predict return direction. The return is calculated as $(p_t - p_{t-1})/p_{t-1}$, where p_t is the close price of Bitcoin at time t. Fig. 2 shows the daily market data with the calculated returns used in this study. It can be observed that there was a tremendous rise in values during the period between 2017 and 2018. We utilize all information up to the current day to predict the return direction of the close price at next day. Specifically, volume of business, market capitalization, open price return, high price return, low price return, and close price return are used as endogenous explanatory variables.

After changing the data in the form of the return and dividing the changed data into the training and the testing portion with ratio 7:3, we apply the FDS algorithm to each index of the data. Under the trained weights obtained by the train data, the prediction is attainable by applying the test data in the architectures. The algorithm of the detailed structures based on MLP, RNN and LSTM is as follows.

Algorithm: The FDS strategy on MLP, RNN and LSTM
Input: Data set up to time T
Output: The prediction of the label (return of close price) at time $T+1$
Procedure:
(Step 1): Change Bitcoin indices of the form of return for each index.
(Step 2): Divide the made data into the training and testing portion with ratio 7:3.
(Step 3): Attain $g_i(x)$ that maps the uniformly distributed data into the observed distribution of
the train data for $i = 1, 2, \ldots, 6$ and define $g_4(x) \equiv g(x)$ where the fourth index denotes the close index.
(Step 4): Apply the FDS strategy, that is, apply $g_i^{-1}(x)$ to each index of the train data for $i = 1, 2, \ldots, 6$
(Step 5): Reshape the train data to match the input structure of MLP, RNN and LSTM illustrated in Fig. 1.
(Step 6): Train using three implementation models with the selected hyper-parameters, that is,
determine weights that are located between layers.
(Step 7): Obtain prediction using the trained weights and the test data, decode the prediction using $g(x)$,
and compare the final output with the label in the test data.

Furthermore, an overview of the proposed algorithms based on the FDS strategy is shown in Fig. 3, and the encoding and decoding processes are illustrated in Fig. 4. All experiments are implemented using Python 3.

5. Experimental results

5.1. Setup

In this subsection, we introduce optimizer and weight initialization used in this study. For the fast and stable reduction of error, the Adam optimizer [58] is employed, which combines the gradient descent with momentum [59], the moving average, and RMSprop [60]. The uniform weight initialization on [-0.1, 0.1] is chosen for each weight in the architecture. The common applications for the three implementation models are mini-batch size (set to 128), epochs (set to 500), ratio of training and testing

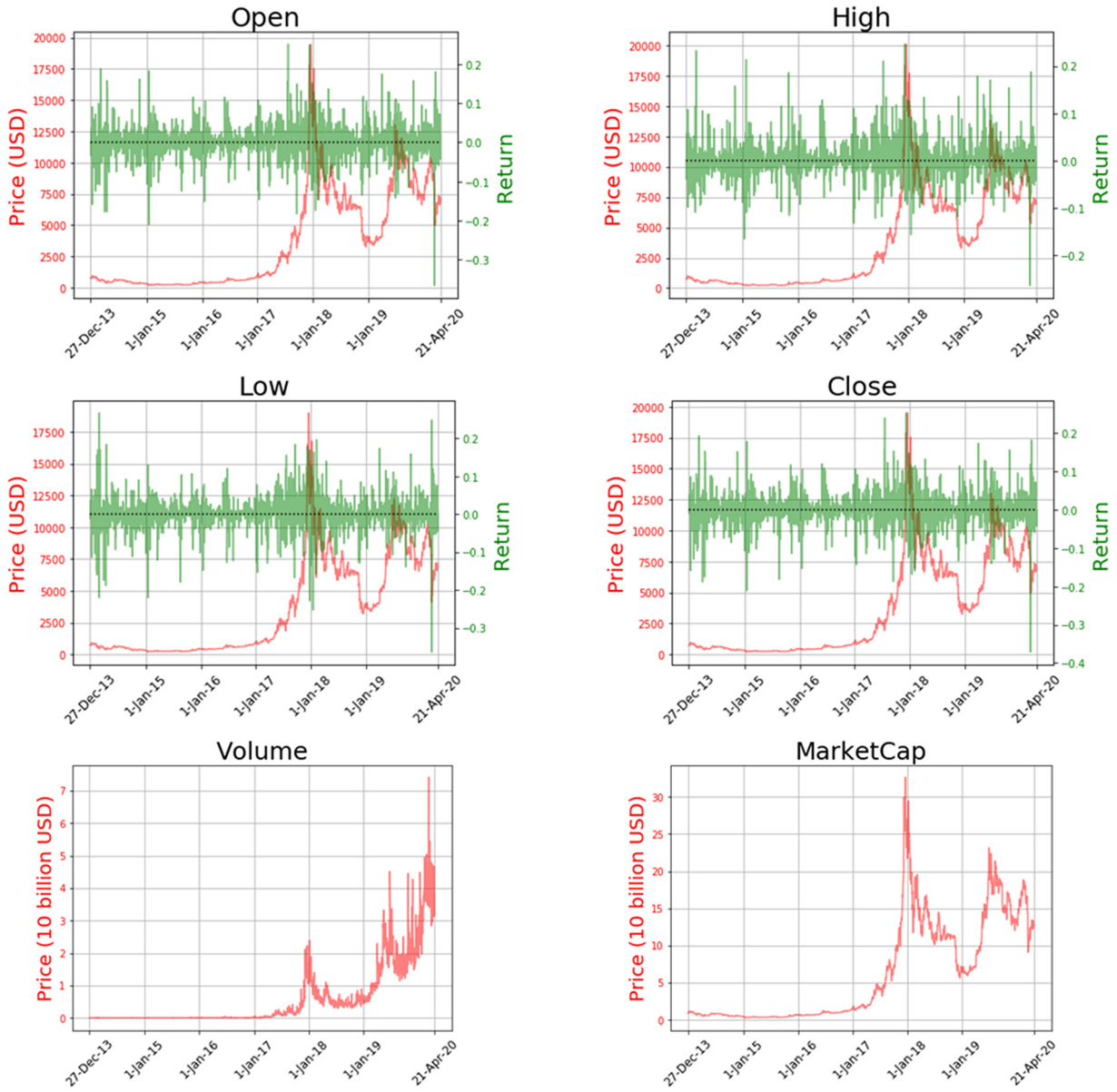

Fig. 2. Daily data used in this study from 28th April 2013 to 27th April 2020.

data (set to 7:3), and learning rate which decreases linearly (from 0.001 to 0.0005) as the epoch increases. Also, the mean square error function is utilized as the loss function. The sigmoid function is used as the activation function in case of MLP and RNN (For LSTM, see formulas from (5) to (10).) except for the output layer (the linear function is located at the output node for all models). For the MLP, four layers are employed where the corresponding number of nodes are 6, 10, 10, and 1. The approximated (excluding bias terms) total number of weights is 170. For the RNN, the depth and dimension of the hidden layer are set to 5 and 10 respectively, and the approximated total number of weights is 170. For the LSTM, the depth and H (See the end of Section 3.1 for detail.) are set to 10 and 4 respectively, and the approximated total number of weights is 160. The number of parameters in each model is approximated ignoring their bias terms. For the hyperparameter selection, we first extract the learning rate, batch size, and epochs using the random search method. And then, using the grid search method, we extract the other hyperparameters whose candidates are selected around the values that show good performance in the random search stage. All hyperparameters are selected so that the test error can be minimized. To measure the classification performances, the percentage correctly classified (PCC) accuracy in [61] can be used as a representative error metric. However, we employ additional error metric (recall, precision, and F1-score) to cross validate biased predictions in one direction. Recall, precision, $F1$-score along with accuracy which are obtainable from the resultant confusion matrix are employed as our error metrics. Here, if both the label and prediction are all positive numbers, the corresponding situation is set to be true positive. To compare the performances of the proposed approaches, we employ four error metrics. If TP, TN, FP, and FN denote the number of true positive, true negative, false positive, and false negative respectively, then the error metrics used in this study are defined by

$$Accuracy = (TP + FN)/(TP + TN + FP + FN), \quad (15)$$

$$Precision = TP/(TP + FP), \quad (16)$$

$$Recall = TP/(TP + FN), \quad (17)$$

$$F1 - score = (2 \times Precision \times Recall)/(Precision + Recall). \quad (18)$$

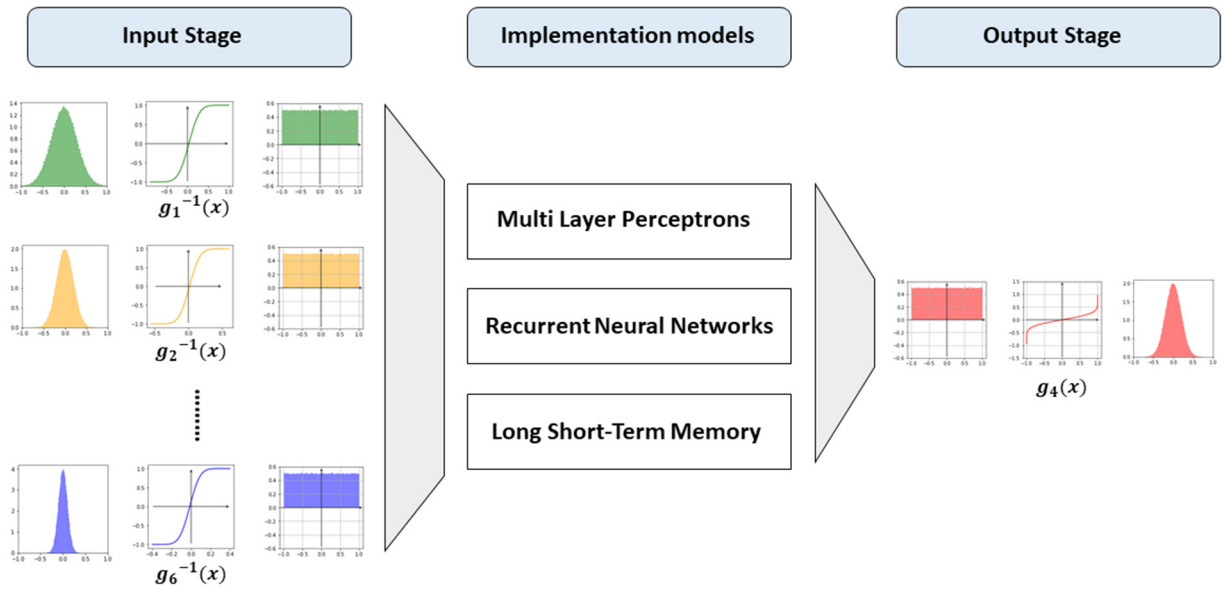

Fig. 3. Overview of the proposed algorithms based on the FDS strategy.

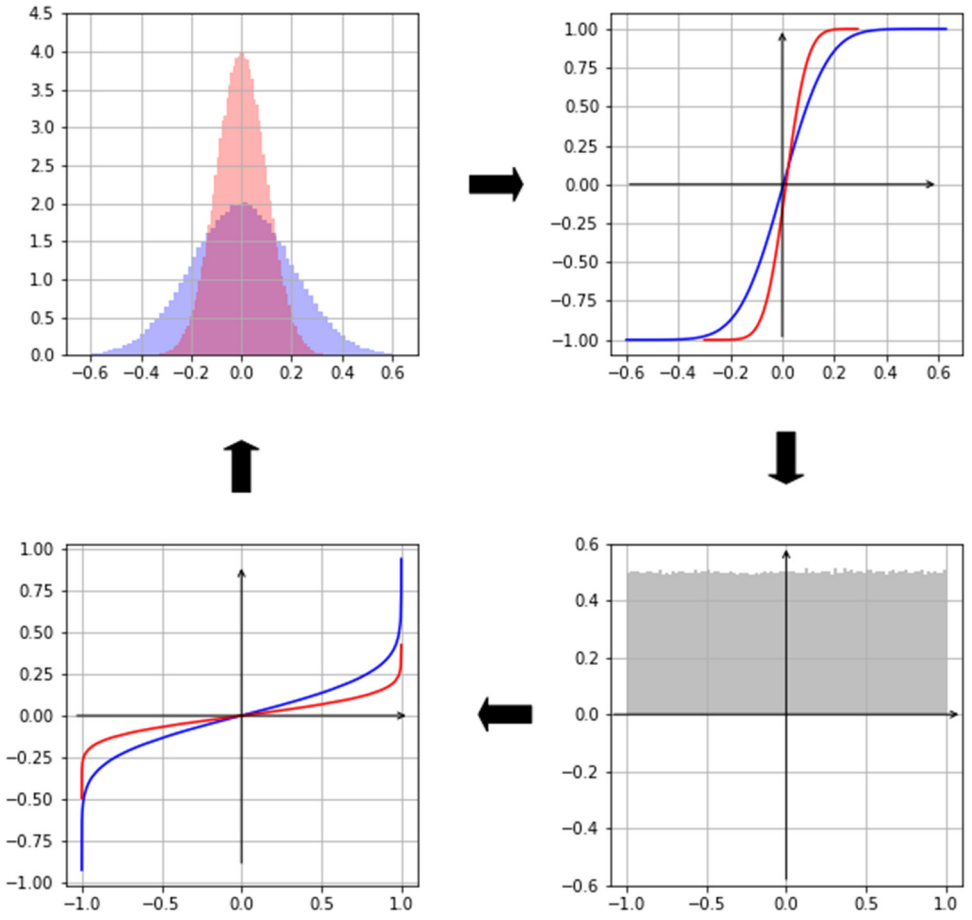

Fig. 4. The encoding and decoding processes. Modified CDF (upper right) changes the original distribution (upper left) to an uniform distribution (lower right). Furthermore, the modified inverse function of CDF (lower left) changes the uniform distribution to the original distribution.

5.2. Results

To assess the up–down prediction performance and to investigate distribution changes in the predictive distributions simultaneously, regression (instead of classification) experiments are conducted using mean square error loss function. It is noteworthy that the regression approach cannot take advantage of the cross entropy loss function which is a representative loss function in classification. However, the approach allows us not only to determine whether the return rises at the next day compared

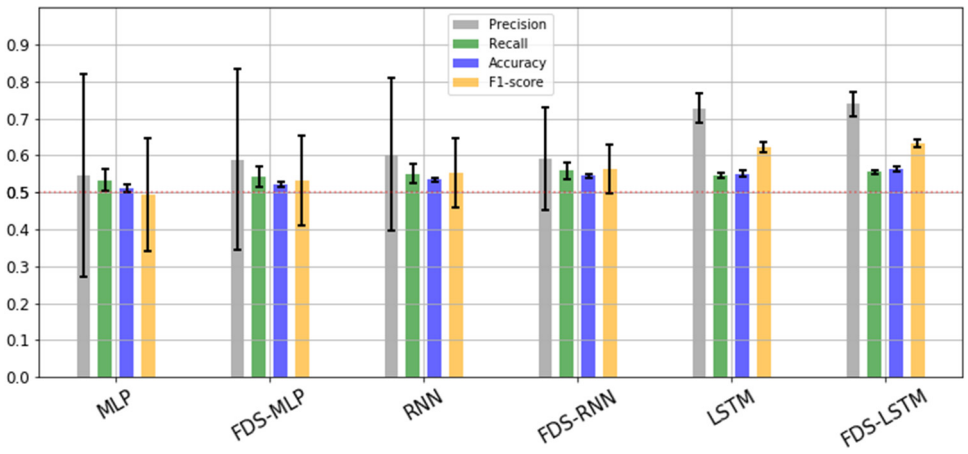

Fig. 5. Performances of FDS equipped algorithms against the native algorithms for MLP, RNN, and LSTM. Error bar indicates the error spread of 100 predictions for each case.

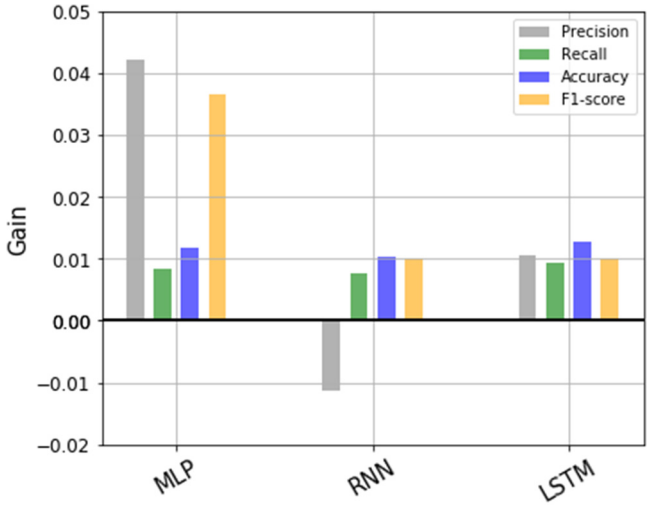

Fig. 6. Accuracy gain (or loss) achieved by the FDS algorithm for each implementation model.

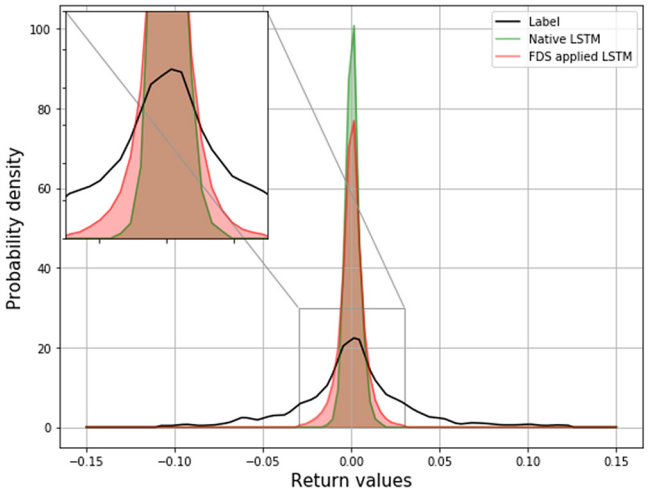

Fig. 7. Distributions of the mean of 100 predictions obtained from the native LSTM (in green) and FDS applied LSTM (in red). The PDF of the label is presented in black line.

to the previous day by comparing two consecutive predictions but also to track the change of the distribution of regressive prediction. In this experiment, the values of a, b, and c in formula (14) are set to -1, 1, and -1, respectively. That is, we manipulate the given return data artificially to the uniform distribution on $[-1, 1]$. The whole experimental results are presented in Fig. 5 and Table 1. Each value in Table 1 is expressed as the mean value and standard deviation derived from 100 experiments, and the best result is given in bold. Also, the accuracy gain achieved by FDS is presented in Fig. 6 and Table 2. For the determined hyperparameters, total spent time for 100 experiments of LSTM and FDS-LSTM is 722 and 761 s, respectively, and spent time of FDS applied case is increased by 5.4% compared to that of the native case.

To determine whether the model that provides the lowest RMSE is statistically significant, we employ the model confidence set (MCS) [62]. The MCS p-value obtained from MCS procedure for a fixed model can be used to determine whether the model belongs to the group of the best models (denoted by $\widehat{\mathcal{M}^*}_{1-\alpha}$ in the original paper where $1-\alpha$ is a level of significance). The model with a relative small MCS p-value is unlikely to be one of the best alternatives in $\widehat{\mathcal{M}^*}_{1-\alpha}$. We present the MCS p-values of 6 models in Table 3 with $\alpha = 0.05$ to evaluate the goodness of the models

The FDS has three notable effects. Firstly, it enhances the accuracy for each implementation model (Table 2), secondly, the FDS narrows the width of error spread, and thirdly, the common area of distributions (PDFs) between the label and prediction is enlarged. For the first, the achieved gains of accuracy by FDS are all over than 1% (0.0118, 0.0104, and 0.0128 for MLP, RNN, and LSTM, respectively), and the largest gain is in the case of LSTM. For the second, because the returns obtained from the financial data are usually very noisy, it is a challenging problem to reduce the error spread of the prediction for each error metric. Hence, the second point of view is a noteworthy result which means the FDS is an appropriate strategy that analyze the noisy nature of the return. For the third, the coincidence of the distribution between the label and prediction does not guarantee the coincidence of time series between them. However, the former is at least a necessary condition for the latter. The overlapping regions of PDFs between the label and the means of 100 predictions obtained from the native LSTM, FDS-LSTM are 0.3801 and 0.5309, respectively (Fig. 7). It is estimated that the improvement in accuracy and the increase in overlapping area are highly related, and the detail analysis of this phenomenon will be our future work. It is noteworthy to report that because the FDS transforms the distribution of return of Bitcoin price into the uniform distribution,

Table 1
Experimental results.

	Precision	Recall	Accuracy	F1-score
MLP	0.5467 ± 0.2741	0.5335 ± 0.0299	0.5105 ± 0.0103	0.4944 ± 0.1538
FDS-MLP	0.5887 ± 0.2443	0.5419 ± 0.0290	0.5223 ± 0.0073	0.5309 ± 0.1221
RNN	0.6020 ± 0.2069	0.5503 ± 0.0262	0.5344 ± 0.0053	0.5520 ± 0.0940
FDS-RNN	0.5906 ± 0.1382	0.5580 ± 0.0230	0.5448 ± 0.0044	0.5622 ± 0.0667
LSTM	0.7280 ± 0.0396	0.5453 ± 0.0071	0.5499 ± 0.0089	0.6231 ± 0.0146
FDS-LSTM	0.7387 ± 0.0325	0.5547 ± 0.0054	**0.5627 ± 0.0059**	0.6332 ± 0.0113

Table 2
Skill gain (or loss) achieved by the FDS algorithm.

	Precision	Recall	Accuracy	F1-score
MLP	0.0421	0.0084	0.0118	0.0365
RNN	−0.0114	0.0076	0.0104	0.0102
LSTM	0.0107	0.0094	0.0128	0.0102

Table 3
Model Confidence Set (MCS) test.

Loss function Ranking	Model	RMSE	Improvement	MCS
1	FDS-LSTM	0.03381	13.1%	1.000
2	LSTM	0.03438	11.8%	0.890
3	FDS-RNN	0.03474	10.8%	0.633
4	RNN	0.03522	9.51%	0.633
5	FDS-MLP	0.03845	1.29%	0.005
6	MLP	0.03892	0.00%	0.000

this study actually excludes outlier data (i.e. abnormal events). This aspect of FDS can be an obstacle to track the distribution changes. Although the overlapping region increases compared to the native case, the prediction performances on both tail regions are unsatisfactory. Modification for the problem will also be future work. In addition, because of the large standard deviations of error metrics exhibited in MLP and RNN, it is reasonable to discuss the performance of FDS mainly in terms of the LSTM which turned out to be the most reliable implementation model with small error spread. Hence, it is expected to have the accuracy gain (or gain rate) of FDS around 0.0128 (or 2.32%) in other neural networks type implementation models. We also expect that the FDS can be applied efficiently in other fields where their data are concentrated near a particular point and extremely noisy.

6. Conclusion

In recent years, the prediction of Bitcoin price has been the most important task because Bitcoin has become one of the most popular investment assets in the financial market. However, it is very difficult to predict Bitcoin price because it is affected by many factors such as high volatility, market supply and demand, stock market state, US dollar exchange rate, and so on. This paper studies the prediction of Bitcoin price based on the machine learning techniques. For the prediction, we consider the return of Bitcoin, which is intractable due to a near-zero concentration and a noisy movement, and three types of approaches: MLP, RNN and LSTM. These approaches are well known as good machine learning approaches for time series prediction. We improve these approaches with a new strategy. Specifically, we introduce the FDS based on the copula theory and construct our algorithms with FDS to predict the return of Bitcoin price. The FDS artificially manipulates the concentrated return into the uniform distributed data. Consequently, it alleviates the sensitivity of the neural networks and improves the prediction performances. Besides the benefit of improvement in accuracy, it turns out that the FDS reduces the error spread and enlarges the overlapping region between the label and the prediction.

Obviously, a more accurate prediction helps investors and traders who want to get a big profit in the financial market. That is, our results will be their interest. Additionally, the researchers who study the prediction of time series based on the machine learning techniques will be interested in our approach and challenge various extensions of the FDS and the copula theory for better prediction. We expect that the FDS can be a good starting point for the study of improving prediction performance through the manipulation or controlling of the data distribution. Also, the integration between FDS and other financial models is expected to improve the predictive performance of the financial time-series data.

Although our approach improves prediction accuracy of Bitcoin price, there are some limitations in this study. Firstly, we do not consider transaction costs in predicting Bitcoin price. In general, low transaction cost and high liquidity are assumed in cryptocurrency trading. However, transaction costs should be considered in predicting the price for a profitable strategy because transaction costs increase in high-frequency trading. Secondly, we consider only Bitcoin among cryptocurrencies in this study. In fact, there are many kinds of cryptocurrencies such as Ethereum, Bitcoin Cash, Litecoin, Ripple, etc, and several papers [48,63] dealt with the prediction of various cryptocurrencies including Bitcoin. Lastly, the explanatory variables for the experiment are rather simple. Recently, Liu et al. [64] used 40 explanatory variables to predict the price of Bitcoin using the machine learning methods. As in [64], the selection of suitable explanatory variables is an important task not only to avoid overfitting but also to use a lot of valuable information. Therefore, transaction costs, other cryptocurrencies and more explanatory variables will be considered for the extension of our approach in the future work.

CRediT authorship contribution statement

Eunho Koo: Methodology, Software, Writing – original draft, Writing – reviewing and editing, Visualization. **Geonwoo Kim:** Conceptualization, Data curation, Writing – original draft , Writing – reviewing and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Geonwoo Kim is supported by the National Research Foundation of Korea grant funded by the Korea government (Grant No. NRF-2017R1E1A1A03070886). Eunho Koo gratefully acknowledges the support of the National Research Foundation of Korea (NRF) grant funded by the Korea government (Grant No. 2015R1A5A1009350, 2020R1I1A1A0106928111 and 2018R1A2A3074566).

References

[1] Satoshi Nakamoto, Bitcoin: A Peer-To-Peer Electronic Cash System, Technical Report, Manubot, 2019.
[2] Mitsuru Iwamura, Yukinobu Kitamura, Tsutomu Matsumoto, Kenji Saito, Can we stabilize the price of a cryptocurrency?: Understanding the design of Bitcoin and its potential to compete with central bank money, Hitotsubashi J. Econ. (2019) 41–60.
[3] David Yermack, Is Bitcoin a real currency? An economic appraisal, in: Handbook of Digital Currency, Elsevier, 2015, pp. 31–43.
[4] Dirk G. Baur, Kihoon Hong, Adrian D. Lee, Bitcoin: Medium of exchange or speculative assets? J. Int. Financ. Mark. Inst. Money 54 (2018) 177–189.
[5] Vladimir Stojanovic, Dragan Prsic, Robust identification for fault detection in the presence of non-Gaussian noises: application to hydraulic servo drives, Nonlinear Dynam. 100 (2020) 2299–2313.
[6] Xuefei Dong, Shuping He, Vladimir Stojanovic, Robust fault detection filter design for a class of discrete-time conic-type non-linear Markov jump systems with jump fault signals, IET Control Theory Appl. 14 (14) (2020) 1912–1919.
[7] Longhui Zhou, Hongfeng Tao, Wojciech Paszke, Vladimir Stojanovic, Huizhong Yang, PD-type iterative learning control for uncertain spatially interconnected systems, Mathematics 8 (9) (2020) 1528.
[8] Peng Cheng, Mengyuan Chen, Vladimir Stojanovic, Shuping He, Asynchronous fault detection filtering for piecewise homogenous Markov jump linear systems via a dual hidden Markov model, Mech. Syst. Signal Process. 151 (2021) 107353.
[9] Yakup Kara, Melek Acar Boyacioglu, Ömer Kaan Baykan, Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul stock exchange, Expert Syst. Appl. 38 (5) (2011) 5311–5319.
[10] Jian-Zhou Wang, Ju-Jie Wang, Zhe-George Zhang, Shu-Po Guo, Forecasting stock indices with back propagation neural network, Expert Syst. Appl. 38 (11) (2011) 14346–14355.
[11] A. Victor Devadoss, T. Antony Alphonnse Ligori, Forecasting of stock prices using multi layer perceptron, International Journal of Web Technology 2 (2) (2013) 49–55.
[12] Mingyue Qiu, Yu Song, Predicting the direction of stock market index movement using an optimized artificial neural network model, PLoS One 11 (5) (2016) e0155133.
[13] Mingyue Qiu, Yu Song, Fumio Akagi, Application of artificial neural network for the prediction of stock market returns: The case of the Japanese stock market, Chaos Solitons Fractals 85 (2016) 1–7.
[14] Jeffrey L. Elman, Finding structure in time, Cogn. Sci. 14 (2) (1990) 179–211.
[15] Ivan Nunes da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni, Silas Franco dos Reis Alves, Forecast of stock market trends using recurrent networks, in: Artificial Neural Networks, Springer, 2017, pp. 221–227.
[16] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[17] Kai Chen, Yi Zhou, Fangyan Dai, A LSTM-based method for stock returns prediction: A case study of China stock market, in: 2015 IEEE International Conference on Big Data, Big Data, IEEE, 2015, pp. 2823–2824.
[18] Thomas Fischer, Christopher Krauss, Deep learning with long short-term memory networks for financial market predictions, European J. Oper. Res. 270 (2) (2018) 654–669.
[19] Yujin Baek, Ha Young Kim, ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module, Expert Syst. Appl. 113 (2018) 457–480.
[20] Taewook Kim, Ha Young Kim, Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data, PLoS One 14 (2) (2019) e0212320.
[21] Ayaz Hussain Bukhari, Muhammad Asif Zahoor Raja, Muhammad Sulaiman, Saeed Islam, Muhammad Shoaib, Poom Kumam, Fractional neuro-sequential ARFIMA-LSTM for financial market forecasting, IEEE Access 8 (2020) 71326–71338.
[22] U JuHyok, PengYu Lu, ChungSong Kim, Unsok Ryu, KyongSok Pak, A new LSTM based reversal point prediction method using upward/downward reversal point feature sets, Chaos Solitons Fractals 132 (2020) 109559.
[23] Gabriel Gajardo, Werner D. Kristjanpoller, Marcel Minutolo, Does Bitcoin exhibit the same asymmetric multifractal cross-correlations with crude oil, gold and DJIA as the Euro, Great British Pound and Yen? Chaos Solitons Fractals 109 (2018) 195–205.
[24] Salim Lahmiri, Stelios Bekiros, Chaos, randomness and multi-fractality in Bitcoin market, Chaos Solitons Fractals 106 (2018) 28–34.
[25] Salim Lahmiri, Stelios Bekiros, Antonio Salvi, Long-range memory, distributional variation and randomness of Bitcoin volatility, Chaos Solitons Fractals 107 (2018) 43–48.
[26] Şahin Telli, Hongzhuan Chen, Multifractal behavior in return and volatility series of Bitcoin and gold in comparison, Chaos Solitons Fractals 139 (2020) 109994.
[27] PRL Alves, Dynamic characteristic of Bitcoin cryptocurrency in the reconstruction scheme, Chaos Solitons Fractals 134 (2020) 109692.
[28] Anne Haubo Dyhrberg, Bitcoin, gold and the dollar–A GARCH volatility analysis, Finance Res. Lett. 16 (2016) 85–92.
[29] Paraskevi Katsiampa, Volatility estimation for Bitcoin: A comparison of GARCH models, Econom. Lett. 158 (2017) 3–6.
[30] Jeffrey Chu, Stephen Chan, Saralees Nadarajah, Joerg Osterrieder, GARCH modelling of cryptocurrencies, J. Risk Financ. Manag. 10 (4) (2017) 17.
[31] David Ardia, Keven Bluteau, Maxime Rüede, Regime changes in Bitcoin GARCH volatility dynamics, Finance Res. Lett. 29 (2019) 266–271.
[32] Eugene F. Fama, Efficient capital markets: II, J. Finance 46 (5) (1991) 1575–1617.
[33] Andrew Urquhart, The inefficiency of Bitcoin, Econom. Lett. 148 (2016) 80–82.
[34] Saralees Nadarajah, Jeffrey Chu, On the inefficiency of Bitcoin, Econom. Lett. 150 (2017) 6–9.
[35] Aurelio F. Bariviera, The inefficiency of Bitcoin revisited: A dynamic approach, Econom. Lett. 161 (2017) 1–4.
[36] Sashikanta Khuntia, J.K. Pattanayak, Adaptive market hypothesis and evolving predictability of Bitcoin, Econom. Lett. 167 (2018) 26–28.
[37] Ladislav Kristoufek, On Bitcoin markets (in) efficiency and its evolution, Physica A 503 (2018) 257–262.
[38] Yonghong Jiang, He Nie, Weihua Ruan, Time-varying long-term memory in Bitcoin market, Finance Res. Lett. 25 (2018) 280–284.
[39] David Vidal-Tomás, Ana M Ibáñez, José E Farinós, Weak efficiency of the cryptocurrency market: a market portfolio approach, Appl. Econ. Lett. 26 (19) (2019) 1627–1633.
[40] Yang Hu, Harold Glenn A. Valera, Les Oxley, Market efficiency of the top market-cap cryptocurrencies: Further evidence from a panel framework, Finance Res. Lett. 31 (2019) 138–145.
[41] Dirk F Gerritsen, Elie Bouri, Ehsan Ramezanifar, David Roubaud, The profitability of technical trading rules in the Bitcoin market, Finance Res. Lett. 34 (2020) 101263.
[42] Aylin Aslan, Ahmet Sensoy, Intraday efficiency-frequency nexus in the cryptocurrency markets, Finance Res. Lett. 35 (2020) 101298.
[43] Akihiko Noda, On the evolution of cryptocurrency market efficiency, Appl. Econ. Lett. (2020) 1–7.
[44] George S Atsalakis, Ioanna G Atsalaki, Fotios Pasiouras, Constantin Zopounidis, Bitcoin price forecasting with neuro-fuzzy techniques, European J. Oper. Res. 276 (2) (2019) 770–780.
[45] Salim Lahmiri, Stelios Bekiros, Cryptocurrency forecasting with deep learning chaotic neural networks, Chaos Solitons Fractals 118 (2019) 35–40.
[46] Dennys C.A. Mallqui, Ricardo A.S. Fernandes, Predicting the direction, maximum, minimum and closing prices of daily Bitcoin exchange rate using machine learning techniques, Appl. Soft Comput. 75 (2019) 596–606.
[47] Salim Lahmiri, Stelios Bekiros, Intelligent forecasting with machine learning trading systems in chaotic intraday Bitcoin market, Chaos Solitons Fractals 133 (2020) 109641.
[48] Saúl Alonso-Monsalve, Andrés L Suárez-Cetrulo, Alejandro Cervantes, David Quintana, Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators, Expert Syst. Appl. 149 (2020) 113250.
[49] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al., Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[50] Jerome T. Connor, R. Douglas Martin, Les E. Atlas, Recurrent neural networks and robust time series prediction, IEEE Trans. Neural Netw. 5 (2) (1994) 240–254.
[51] Vinod Nair, Geoffrey E. Hinton, Rectified linear units improve restricted boltzmann machines, in: ICML, 2010.
[52] Paul J. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (10) (1990) 1550–1560.
[53] Laszio Tora, John White, Christel Brou, Dlane Tasset, Nicholas Webster, Elisabeth Scheer, Plerre Chambon, The human estrogen receptor has two independent nonacidic transcriptional activation functions, Cell 59 (3) (1989) 477–487.
[54] Xavier Glorot, Yoshua Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[55] Umberto Cherubini, Elisa Luciano, Walter Vecchiato, Copula Methods in Finance, John Wiley & Sons, 2004.
[56] Eric Bouyé, Valdo Durrleman, Ashkan Nikeghbali, Gaël Riboulet, Thierry Roncalli, Copulas for finance-a reading guide and some applications, 2000, Available At SSRN 1032533.
[57] Juan Carlos Rodriguez, Measuring financial contagion: A copula approach, J. Empir. Financ. 14 (3) (2007) 401–423.
[58] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[59] Ning Qian, On the momentum term in gradient descent learning algorithms, Neural Netw. 12 (1) (1999) 145–151.
[60] Tijmen Tieleman, Geoffrey Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Netw. Mach. Learn. 4 (2) (2012) 26–31.
[61] Lean Yu, Yaqing Zhao, Ling Tang, Zebin Yang, Online big data-driven oil consumption forecasting with google trends, Int. J. Forecast. 35 (1) (2019) 213–223.
[62] Peter R. Hansen, Asger Lunde, James M. Nason, The model confidence set, Econometrica 79 (2) (2011) 453–497.
[63] Romina Torres, Miguel A Solis, Rodrigo Salas, Aurelio F Bariviera, A dynamic linguistic decision making approach for a cryptocurrency investment scenario, IEEE Access 8 (2020) 228514–228524.
[64] Mingxi Liu, Guowen Li, Jianping Li, Xiaoqian Zhu, Yinhong Yao, Forecasting the price of Bitcoin using deep learning, Finance Res. Lett. 40 (2021) 101755.