eISSN: 2789-858X

SCIENTIFIC JOURNAL FOR THE FACULTY OF SCIENCE – SIRTE UNIVERSITY

VOLUME 3 ISSUE 2 OCTOBER 2023

Bi-annual, Peer- Reviewed, Indexed, and Open Accessed e-Journal

Legal Deposit Number@NationaL Library (Benghazi): 990/2021

1.02/2022

DOI: 10.37375/issn.2789-858X - Indexed by Crossref, USA

jsfsu@su.edu.ly

journal.su.edu.ly/index.php/JSFSU

Scientific Journal for the Faculty of Science-Sirte University Vol. 3, No. 2 (2023) 18-28

Scientific Journal for the Faculty of Science-Sirte University

Journal home page: http://journal.su.edu.ly/index.php/JSFSU/index

DOI: 10.37375/issn.2789-858X

A Comparison of the Effectiveness of Artificial Neural Network Models for Time Series Data Prediction

Umalkher S. Mohamed

Computer Science Department, Education Faculty, Sebha University, Ghat, Libya.

DOI: https://doi.org/10.37375/sjfssu.v3i2.1516

ARTICLE INFO:

Received: 13 June 2023

Accepted: 25 September 2023

Published: 26 October 2023

Keywords: Stock prices, Prediction models, artificial neural network, Time series data.

A B S T R A C T

Stock market prediction has become an important research area. During the last few years, various Artificial Neural Networks (ANNs) models have been proposed for stock market prediction problems. This study aims to compare the prediction performance of three artificial neural network techniques, i.e., Back Propagation Neural Network (BPNN), Radial Basis Function Neural Network (RBFNN), and Long Short-Term Memory (LSTM) in predicting the close prices of the next day. The data used in this study includes the daily close prices of two companies, Microsoft (MSFT) and Apple Inc. (AAPL), belonging to the NASDAK-100 stock exchange. The models were employed using Python software, a single hidden layer prediction model was constructed, and the effect of prediction accuracy on the number of neurons was identified. The performance of the models is measured in terms of their Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and R-squared (R^2) score values. The experimental results indicated that the LSTM forecasting model outperformed alternative models with a high degree of accuracy and was found to be very efficient in learning time series data.

1 Introduction

Stock market prediction is an attractive topic for investors and researchers and has become an essential component in the financial field. Stock market prediction aims at building models based on past prices to simulate the future price, which is important for financial organizations It plays an increasingly important role in that it has a great impact on leading a proper decision on financial institutions, thus making a better profit. However, predicting the stock price itself is a challenging task due to its complex characteristics; stock markets are dynamic, nonlinear, complicated, nonparametric, chaotic, and exhibit wide variation (Binkowski et al., 2018, Shah et al., 2019). Accordingly, various solution techniques have been proposed for obtaining accurate prediction results over the years. Early techniques involved statistical methods such as Autoregressive Moving Average (ARMA) model (Rubi et al., 2022, Moradi et al., 2021) and Logistic Regression (LR) (Chen et al., 2020) (Mansouri et al., 2016). Such techniques treat the stock price movement as a function of a time series and are solved as a regression problem. However, the stock market has high volatility in nature. These statistical methods may suffer from difficulty in revealing the internal laws of the stock market. With recent developments in Computer Science, more developing techniques have been proposed to analyze nonlinear relationships in financial time series using Artificial Intelligence (AI) techniques.

Most previous studies in this area showed that AI techniques such as Artificial Neural Networks (ANN) (Gao et al., 2020), Fuzzy Logic (FL)(Hašková et al., 2023), Genetic Algorithm (GA) (Jafari et al., 2019), Support Vector Machine (SVM)(Kurani et al., 2023) are found to be more efficient than statistical methods. AI techniques have the capability of detecting the structures and nonlinear patterns of data (Mokhtari et al., 2021, Chen et al., 2018). In the most existing prediction methods, Neural Networks (NNs) models have been used in numerous studies for stock price prediction, and have revealed the superiority in solving nonlinear problems(Vijh et al., 2020, Nikou et al., 2019) (Siami-Namini et al., 2018, Moghar and Hamiche, 2020, Vijh et al., 2020). This advantage comes from the capability of NNs to model nonlinear techniques without previous information about the processing techniques, which can be very powerful for predicting the stock market. ANN has the benefit of storing experiential knowledge and making it accessible for use. Additionally, it has the advantages of automatic learning of features, high generalization and identification of unseen data. Thus, ANN has attracted the attention of many researchers. The first contribution of this study is the development of three artificial neural networks models, namely, Back Propagation Neural Network (BPNN), Radial Basis Function Neural Network (RBFNN), and Long Short-Term Memory (LSTM) for stock market prediction problems. The second contribution of this work is the comparison of forecasting performance of the proposed models, our aim is to verify which method is effective in predicting next day closing prices. This study focused on the daily stock close prices of two companies i.e., Microsoft (MSFT) and Apple Inc., which are from January 26, 2017, to December 26, 2020. The data of our experiment was collected from the Yahoo Finance website. Our dataset is divided into training sets which is used to train the model and update the model parameters, and test sets which would be used for testing so that we can use the data to optimize the model for data prediction. Similar input datasets were used to enable comparison between the proposed models. The performances were evaluated based on four metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and R squared (R2) score values. The rest of this paper is structured as follows. A literature review of the related studies is presented in Section 2. The descriptions of the artificial neural network techniques are given in Section 3. Martials and methods of our research are described in Section 4, we introduce our dataset and describe the evaluation indicators of our experiment, and we explain our experimental procedures. The experiential results and discussion are detailed in Section 5, a comparison of the effectiveness of the proposed methods based on obtained results explained in this section. Finally, the paper is concluded in Section 6.

2 Related Work

The effectiveness of neural networks for stock market prediction problems has been examined for many years. Prediction of stock price movements was explored in (Ramesh et al., 2019), the authors presented a detailed analysis to show the efficiency of Back Propagation Neural Network (BPNN) model in predicting stock returns. They explained the importance of the choice of activation function, learning rate, and the number of neurons in the hidden layer to increase the performance of the BPNN. The results of their work were compared with a Multiple Linear Regression (MLR) model, and they found that the BPNN based model gave better results in predicting stock returns with good accuracy. The authors in (Song et al., 2018) evaluated the performance of neural network models in stock market index perdition using adjusted close prices of three different stocks, namely Bank of China, Vanke A, and Kweichou Moutai. They reported a comparison between BPNN, Radial Basis Function Neural Network (RBFNN), General Regression Neural Network (GRNN), Support Vector Regression (SVR), and Least Squares Support Vector Regression (LS-SVR). The performance of the proposed models is evaluated using statistical indicators such as Adopting Mean Square Error (AMSE) and Mean Absolute Percentage Error (MAPE). They observed that the BPNN model outperforms the other four models. The performance of ANN was also investigated in (Mansor et al., 2020). Their work aims to compare the predictive performance of five neural network architectures, namely: Multiple Linear Regression (MLR), Elman Neural Network (ENN) Elman, Jordan Neural Network (JNN), Radial Basis Function (RBF), and Multilayer Perceptron (MLP), in predicting six traded stocks of the Brazilian stock exchange. They use the test data to tune the number of input variables and suitable hidden layers. The models were trained to predict the closing price of the next day from the previous values. The performance of all fitted models was assessed by the Root Mean Square Error (RMSE). In their results, they found that the ENN, JNN, MLP, and MLR networks presented quite similar RMSE for all times series analyzed in their research and the MLR may be tuned to provide results quite similar to more complicated models such as the MLP, ENN, and JNN neural networks, since it provides the best result .The performance of the neural network was also investigated in (Moghar and Hamiche, 2020) using LSTM for forecasting two stocks on the New York Stock Exchange (NYSE), i.e., Google (GOOG) and Nike (NKE) stock prices extracted from the Yahoo Finance website. The reported results indicated that the proposed Long Short-Term Memory (LSTM) model could improve forecasting accuracy and is capable of tracing the evolution of stock prices for both stocks. A comparison study was reported in (Tiwari et al., 2020). The authors compared ANN and Fuzzy Logic (FL) in a problem of stock market prediction using the daily historical prices listed on the Bombay Stock Exchange (BSE). A Back Propagation (BP) algorithm was used to train the neural network. The experimental results of their work revealed that the neural network gave better prediction results in terms of the minimum value of mean square error. The authors in (Nabipour et al., 2020) conducted a comparison study between Decision Tree

(DT), Random Forest (RF), Adaptive Boosting (Adaboost), Extreme Gradient Boosting (XGBoost), Support Vector Classifier (SVC), Nave Bayes (NB), K-Nearest Neighbors (KNN), Logistic Regression (LR), and ANN and two deep learning methods, Recurrent Neural Network (RNN) and LSTM. Ten years of historical data from four stock markets, namely Petroleum, Diversified Financials, Basic Metals, and Non-Metallic Minerals, were used. Ten technical indicators from the selected historical data were chosen as input values. They conclude that RNN and LSTM are superior models in both approaches compared with other models, and a significant improvement in the performance of models is observed when they use binary data as input values. In (Hiransha et al., 2018), four types of ANN architectures have been utilized, namely Multilayer Perceptron(MLP), Recurrent Neural Network(RNN), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN), for predicting the stock prices of two different stock markets, the National Stock Exchange (NSE) of India and the New York Stock Exchange (NYSE). The authors compared the obtained results with the Autoregressive Integrated Moving Average (ARIMA) model. The results show that the neural network models outperform the ARIMA model. Another study is discussed in (Botunac et al., 2019). The performance of three different neural network models was compared on financial data using the Feed Forward Neural Network (FFNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM). In their study, the data were preprocessed with a simple moving average and filtered with a discrete wavelet transformation to achieve better results. Among the tested neural network models, better results were obtained using LSTM than other neural network models. A comparison between Autoregressive Integrated Moving Average (ARIMA), Back Propagation Neural Network (BPNN), and Genetic Algorithm (GA) was conducted by (Alfred et al., 2015) to predict the short-term time series network traffic activity datasets, which were obtained from the ICT Universitas Mulawarman. The performances of these models are compared based on mean squared error. Based on the results obtained, BPNN is found to be very efficient in capturing the structural relationships in time series data. Another study reported by (Nikou et al., 2019), the study compared the performance of Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), Random Forest(RF), and Support Vector Regression (SVR) with daily closing values of iShares MSCI United Kingdom exchange rates. They report that the LSTM model is a better predictive model in the prediction of the close price of iShares MSCI United Kingdom.

3 Background

In this section, we review the basic concepts of the underlying technologies used in this study.

3.1 Back Propagation Neural Network

The Back Propagation Neural Network (BPNN) is a machine learning algorithm, that aims to minimize the mean square error gradually between the actual and target outputs of the network .The common structure of BPNN model is illustrated in Fig. 1.

As viewed in architecture, there are three layers: an input layer, a hidden layer, and an output layer. The input layer is connected to the hidden layer by interconnection weights, and the hidden layer is connected to the output layer by interconnection weights**.** Each layer has a number of nodes and each node represents a neuron.

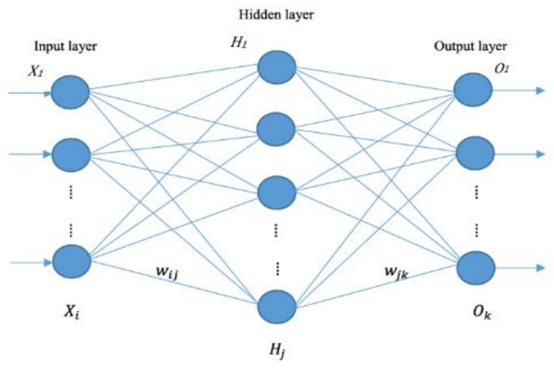

Figure (1). Back Propagation Neural Network Architecture

BPNN requires a suitable choice of network architecture, which is the number of neurons required to form the network. It is important to identify the total number of neurons in the hidden layer. Selection of the neurons is done by the trial and error method. A small number of neurons may result in incorrect estimation whereas more neurons result in overestimation. In addition, the formation of the network requires a proper choice of network layers. Increasing the number of layers increases the computational complexity of the neural network, which can increase the time taken for convergence. Each hidden neuron in BPNN performs a weight summation operation. The outputs of all hidden layer nodes are calculated as follows:

$$h_j = f(\sum_{i=1}^{n} w_{ij}x_i) \quad (1)$$

where *i* is the input node (*i*=1,2,…,*n*), *j* is the hidden layer node (*j*=0,1,2,…,*m*), w_{ij} is connection weight from input node *i* to hidden node *j*, h_j is the output of the *j*th node in the hidden layer, and f is the activation function of nodes. In our an model, we use the sigmoid function as activation function, which is defined as

$$f(x)=\frac{1}{1+exp(-x)} \quad (2)$$

A suitable selection of the activation function is a critical step that introduces nonlinearity into the network for the hidden layer, because it provides the ability of the network to capture the nonlinear relationship between input and output. Several BPNN training algorithms have been proposed for adjusting the connection weights, such as Gauss-Newton's algorithm (Nandy et al., 2012) and the Levenberg–Marquardt algorithm (Mustafidah et al., 2019). All algorithms use the gradient of the performance function to determine the adjustment of the weights to minimize the performance. That is the Widrow-Hoff learning rule, in which the network weights are moved along the negative of the gradient of the performance function, which is the steepest descent direction. The training process of the network occurs in the following steps. In the first step, the training samples were entered into the input layers. Then, the presented data propagates through the hidden layers to the output layers. In the second step, the errors are calculated by taking the difference between the actual and desired outputs. The network weights will be continuously adjusted by propagated back network error until the desired output error is obtained, which represents the minimum output error. This means that the smaller the output error, the better fit the model. Python software provides effective machine learning libraries for building and optimizing neural network models which, has been considered in this study. The formula for the BP algorithm is described by the following equations:

$$W(n) = W(n-1) - \Delta W(n), \quad (3)$$

where,

$$\Delta W(n) = \gamma \frac{\partial E}{\partial W}(n-1) + \theta \Delta W(n-1), \quad (4)$$

where γ is the learning rate, E is the gradient of error function, and $\theta \Delta W(n-1)$ the quantity of incremental weight.

Back propagation networks have the ability to learn complicated multidimensional mappings. It can be used to simulate nonlinear mapping models, solve some real-world problems, such as classification, and prediction; it is a well-known, powerful tool in problem solving for various stock price predictions.

3.2 Radial Basis Function Neural Network

A Radial Basis Function Neural Network (RBFNN) is a type of feed forward neural network with the use of radial basis functions as activation functions. The basic structure of RBFNN is shown in Fig. 2.

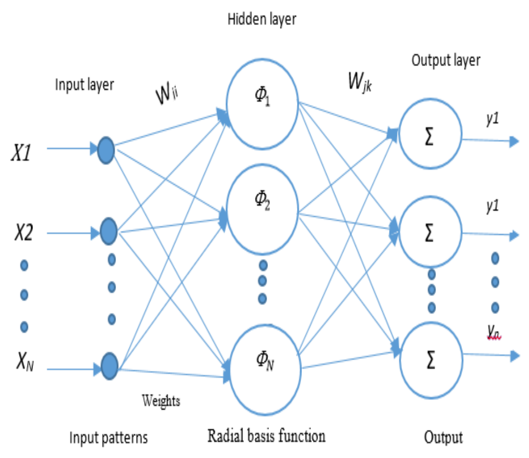

Figure (2). The basic structure of an RBF neural network

Typically, the network has three layers: an input layer, a hidden layer with activation nodes based on the Radial Basis Function (RBF), and a final output layer. Similar to the typical ANN structure, each layer consists of a set of nodes. The input layer receives the input vector and transforms its value to the hidden layer. Each node in the hidden layer has its own RBF centered at a point and is connected to the input layer. The output layer produces the outcome of the prediction model. The output of the RBFNN is defined as a weighted average of the incoming signals from the hidden layer. In practice, the structure of RBFNN may require more neurons in a hidden layer than Feed Forward Neural Network (FFNN). The number of hidden neurons in the hidden layer is critical for determining the RBF model capability.

The nonlinearity of the RBFNN model is presented by mapping the input data to hidden layers using the basis function, whereas a linear mapping appears from the hidden layers to the output layer. The activation function of the RBFNN can take different types of radial basis functions. A Gaussian activation function has been commonly used as an activation function in the hidden neurons of the RBF model. However, it is difficult for the Gaussian activation function to approximate constant values, and the models may suffer from an approximation of these values. A sigmoid function as the basis function of the network may replace the Gaussian functions, so more accurate results can be found by an RBF network(Wu and Wilamowski, 2013) . In our model, a sigmoid function was used to deal with this problem. When the sigmoid function is used, the common form of the kth node of the output layer that has the weight W_k and the neuron i of the hidden layer is as follows:

$$y_k = \sum_{i=1}^{n} w_{ki} exp(\frac{1}{1+exp(-x)}), k = 1,2,3,..,q \quad (5)$$

RBFNN generally trains faster than BPNN due to the use of the radial basis functions and can effectively fit any

nonlinear function, and it is not easy to fall into the local optimal solution; RBFNN can improve the accuracy and decrease the training time complicity.
RBF networks have many uses, such as time series prediction (Sohrabi et al., 2023), system control (Mehrsai et al., 2013), and classification(Jiang and Li, 2019). Many applications have shown that RBFNN can be a useful method for the stock market prediction. This is due to its ability to perform universal approximation and regularization, it can approximate any continuous function with high precision (Liao et al., 2003).

3.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) neural network is an improved type of RNN with better performance in long-term dependency in many applications (Qu and Zhao, 2019). RNN is a form of neural network that has the ability to process sequential inputs recurrently due to the internal time loops at each hidden layer unit in which the output of the unit at a specific time step is taken as the input for the next time step. However, the RNN has a limit in real practice, in that it is just able to record the information in a few past steps. It appears to have vanishing gradient problems when dealing with long time series data. The LSTM units have been proposed to address the vanishing gradient problem where the hidden layer is replaced by recurrent gates called forget gates. These gates allow modelling of long-term dependences in sequence data and prevent the vanishing gradient problem (Gers et al., 2000, Graves and Graves, 2012, Nugaliyadde et al., 2019). LSTM networks are adapted for identifying long-term dependencies and using them for future prediction, which was impossible in a standard RNN architecture where the network could just learn a limited number of short-term time series. The key element of the LSTM is the appending of cell state or a memory cell, which **is** comprised of three basic gates: an input gate, a forget gate, and an output gate. The forget gate decides which information from the previous cell is completely to be kept or ignored. The input gate chooses which new values need to be updated in the cell state, and the output gate lets the cell state have an impact or not on the latest present time step**.** LSLM also has a number of hidden units.

The sigmoid neural net layer and weights are used to assign importance to information. The stochastic gradient descent based algorithm is used to enforce constant error propagation (neither exploding nor vanishing) through its internal units. Fig.3 shows the architecture of the LSTM**.**

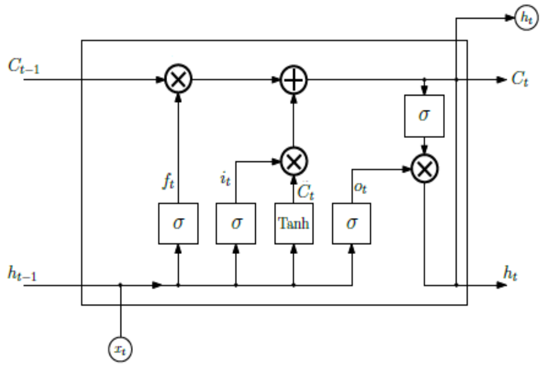

Figure (3). Basic architecture of a LSTM

The nature of the three basic gates of the LSTM can be described in the following steps: In the first step, the forget gate has to decide which information to discard from cell state. The formula of the forget gate is characterized by the following equations:

$$f_t = \sigma\left(W_f \,.\, [h_{t-1}, x_t] + b_f\right) \tag{6}$$

$$R_t = \frac{1}{1+e^{-t}} \tag{7}$$

where h_{t-1}, x_t , $R_t$and f_t represent the output at the previous time (*t-1*), input at the present timestamp (*t*), a Sigmoid function and a forget gate, respectively, W_f and $b_f$are the weight matrices and bias vectors, respectively, that need to be learned during the training process.
In the second step, the input gate layer decides which new information should be saved in the cell state. The input gate has the following mathematical representation:

$$i_t = \sigma(W_i \,.\, [h_{t-1}, x_t] + b_i \tag{8}$$

$$\tilde{C}_t = tanh(W_c \,.\, [h_{t-1}, x_t] + b_c) \tag{9}$$

Updating the information stored in the cell state has been done by the following equation:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \,, \tag{10}$$

where C_t, C_{t-1} and $\tilde{C}_t$ represents the current cell state value , the last timestep cell state value, and an update for the current cell state value at timestamp *t* respectively.

In the third step, the output gate layer determines the output information. The characteristics of the output gate layer are expressed in the following equations:

$$O_t = \sigma(W_o \,.\, [h_{t-1}, x_t] + b_o) \,, \tag{11}$$

$$h_t = O_t \,.\, tanh(C_t) \,, \tag{12}$$

where O_t is the output gate that indicates the candidate for cell state at timestamp *t* and h_t is the LSTM block's output information at time *t*.

Building an LSLM model includes the following steps:
Step1: The collected data is preprocessed to eliminate any undesirable or incomplete data.
Step2: Divide the data into training and testing data. The training data is used to build the model and the test data to evaluate the model's performance.
Step3: Selection of a suitable architecture for the LSLM Model. This step requires an appropriate choice of the number of the nodes and layers.
Step4: Train the model by repeating the training cycle until the desired result is obtained.
Step5: Making predictions and assessing the model's performance.

4 Materials and Methods

4.1 Dataset

The historical data of the daily stock close prices for two different well-known NASDAQ-100 companies has been chosen to represent our dataset, namely Microsoft Corporation (MSFT) and Apple Inc. (AAPLE). Our dataset is taken from the Yahoo Finance website and includes four years of data, which is from January 26, 2017 to December 26, 2020. Fig.4 shows two different closing stock prices.

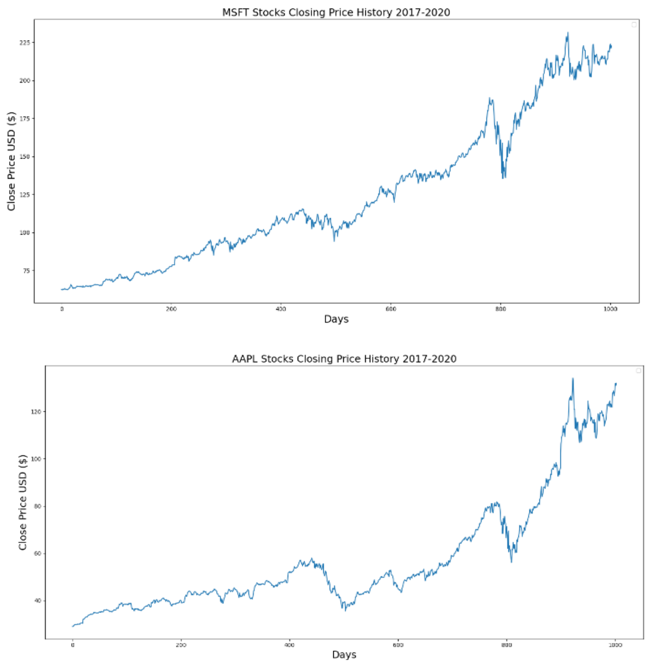

Figure (4). Closing stock prices for MSFT and AAPLE

Before using the dataset for the training process, it is important to take a preprocessing step. The preprocessing step used in this study is data normalization using the equation formula:

$$Z'_i = \frac{Z_i - min(Z_i)}{max(Z_i) - min(Z_i)} \quad , \qquad (13)$$

where $\boldsymbol{Z'_i}$ is normalized values, $min(Z_i)$ is the minimum value of input $\boldsymbol{Z_i}$, and $\boldsymbol{max(Z_i)}$is the maximum value of input $\boldsymbol{Z_i}$.
The normalization process aims to make data statistically comparable. This can also help the learning process be more stable. Data normalization helps to smooth the convergence, which prevents dramatic changes in the gradient. After processing, we anti-normalize the output with the following equation:

$$\hat{y_t} = y'_t(y_{max} + y_{min}) + y_{min} \,, \qquad (14)$$

where y'_t represents the predicted data after anti-normalization and y_{max} and y_{min} the minimum and maximum data, respectively, of output y'_t.

Our dataset was divided into 90:10 ratios for training and testing purposes, respectively. For all models, the training dataset is used to train the neural network models and update model parameters. The test dataset was used to optimize the models for data prediction, in other words, the test dataset was used for simulating the trained network and checking the accuracy of the trained network. The accuracy of the model on the test dataset gives you a very rough estimate of how accurate the model will be when presented with new, previously unseen data.

4.2 Model Design

Our aim looks at the comparison of the three NNs models BPNN, RBFNN, and LSTM in a problem of stock market prediction. The basic principles of them have been detailed in the previous section. The process of training algorithms is described later in the paper. After training the proposed models and evaluating their accuracy, we compared the output data given by the network with the testing dataset. The results of these comparisons are given in detail in the later part of this paper. Fig.5 shows the block diagram of the methodology.

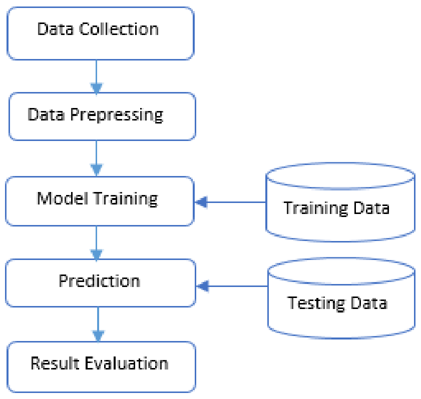

Figure (5). Methodology block diagram

4.3 Performance Evaluation Criteria

We select the following statistical metrics to evaluate the predictive performance of proposed models: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and coefficient of determination (R2). MAE is the measure of the deviation between the actual and predicted values. MAPE is commonly presented the accuracy of the model as a percentage in which equation (16) is multiplied by 100. The statistical matrix RMSE measures the mean square error of the actual and predicted values; its value is always positive and zero in the ideal case. The lower the MAPE, MAE and RMSE values, the closer the predicted time series values are to actual values, indicating that the model is accurate. R2 also gives the accuracy of the model as percentage. The smaller values present the better predictor model; it gives you knowledge of how well the model predicts the new dataset. Its values are between zero and one, and the largest value is the better value. The equations for these criteria are as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_t - y_t')^2}, \qquad (15)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_t - y_t'|}{y_t} \times 100\,, \qquad (16)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_t - y_t'|}{y_t} \qquad (17)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_t - y_t')^2}{\sum_{i=1}^{n}(y_t')^2}, \qquad (18)$$

Where

y_t is the actual value at the time *t*
y_t' is the forecast value at the time *t*
n is the total number of tested datasets

4.4 Experimental implementation

All experiments were conducted in Python with PyCharm Professional Edition 2020.2.1 on an Intel Core i5-5200U CPU machine. The implementation includes construction, training, testing, and evaluation of neural network models. All neural network models were developed using the open source deep learning tool Tensorflow (Abadi et al., 2016) with Keras (Ketkar, 2017) version 2.0.8 as the front-end interface. The Adam optimizer was applied to all of the neural network models for weight modification and Mean Square Error (MSE) was used as the cost function. The formula for MSE is given as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_t - y_t')^2, \qquad (19)$$

where $\mathbf{y_t}$ and $\mathbf{y_t'}$ are actual and predicted values at time ***t***, respectively, and ***n*** is the data size of the trained set.

To ensure keeping the same conditions when training different models, a single hidden layer was used in all prediction neural network models and one fully connected layer has been used as the output layer, which gives the predicted next day value. Furthermore, the batch size is considered as 32, and epochs are kept constant at 200 for all prediction models. To evaluate the performance of the models, we used the prediction accuracy indicators. The accuracy indicators tell us how accurate the forecast model is at predicting future trend movements**.** To indicate the performance of the ANN models, the optimal number of hidden layer neurons should be decided in ANNs. We start our implementation with the first method of PBNN. A number of hidden layer neurons have been tried, which are illustrated in Table 1 for two datasets. A sigmoid transfer function was used in the hidden layers, and a rectified linear unit activation function was used in the output layer. The optimal number of hidden layers of neurons is based on the smallest MSE value generated, which is shown in bold in Table 1. It is noticed that as the number of hidden neurons increases, the mean square error increases gradually. The second method is RBFNN; we start the experiment by determining the optimal number of hidden neurons in the network architecture. A number of hidden layer neurons were modified, and Table 2 shows the results of this experiment for two data series. The value of the optimum hidden neuron that was chosen in this study is given in bold in the table with the smallest test mean square error value. Furthermore, the output layer of transform functions was chosen to be the sigmoid transfer function in the RBF model**.**

Table (1). The result of the determination of the number of neurons in the hidden layer of PBNN

No. of neurons	MSE	
	AAPLE	MSFT
100	**0.000105**	**0.000108**
200	0.000340	0.000120
300	0.001482	0.000407

Table (2). The result of selecting the optimal number of neurons for the RBFNN's hidden layer

No. of neurons	MSE	
	AAPLE	MSFT
100	**0.000184**	**0.000200**
200	0.000208	0.000209
300	0.000220	0.000261

The third model is the LSTM model. Similar to previous methods, the optimal number of neurons should be specified. In this model, a sigmoid activation function was used in the LSTM cells. The optimum number of neurons has been obtained after trying different values and the results of the experiment are presented in Table 3. The optimal number is given in bold in the table**.** Furthermore, we used the dropout method on all layers of the network to prevent overfitting during network training and improve generalizability. The dropout units technique in a neural network depends on the value of the probability ***p*** of retaining a hidden unit in a neural network. To obtain the optimal value of *p*, which is a hyperparameter in the dropout, we tried different values of ***p*** ranging from 0.2 to 0.6 after considering the optimal number of the neurons as 300 in the LSTM model and 100 neurons in BPNN and RBFNN models, then we chose the best value of ***p*** that gives the best accuracy with a small amount of error. The results are given in Table 4, which indicates that the best accuracy is obtained with ***p*** = 0.2 in the BPNN and LSTM models, whereas the dropout with the probability of 0.4 in the RBFNN model has fewer errors, thus has been considered the optimum value of *p* in RBFNN.

Table (3). The result of selecting the LSTM's optimal number of neuron.

No. of hidden layer	Number of neurons	MSE	
		AAPLE	MSFT
1	100	0.000183	0.001925
1	200	0.000153	0.001326
1	**300**	**0.000140**	**0.001100**

Table (4). The result of estimating the optimal dropout rate *p*

Dropout rate *p*	MSE					
	PBNN		RBFNN		LSTM	
	AAPLE	MSFT	AAPLE	MSFT	AAPLE	MSFT
0.2	**0.000401**	**0.000350**	0.000223	0.000211	**0.000135**	**0.000841**
0.3	0.000941	0.000474	0.000151	0.000179	0.000322	0.001344
0.4	0.001048	0.002277	**0.000130**	**0.000162**	0.000281	0.001915
0.5	0.003536	0.002015	0.000148	0.000183	0.000422	0.002907
0.6	0.005195	0.005153	0.000161	0.000150	0.000683	0.003288

5 Results and Discussion

Table 5 shows the summarized experimental results of three neural network models on the two different datasets; the best performing model is displayed in boldface. By comparing the statistical errors for all models, the LSTM model achieves the lowest RMSE and MAPE, and the highest R^2 on the prediction of both datasets. However, in terms of MAE, RBFNN reported the smallest errors value of 0.009586 on the prediction on AAPLE dataset, while on the prediction of MSFT dataset, LSTM has a better performance. LSTM model performed well because it does not depend on any previous information for prediction, which enables the model to understand the dynamic changes and patterns occurring in the current window. This can be beneficial for non-stationary time series such as stock market. RBFNN model also performed well and has very close results to LSTM model. However, in the case of BPNN, the model has the worst performance across two stacks. This is due to its simple architecture, which could limit its capability to make predictions for non-stationary time series. We could also find from Table 5 that the AAPLE models show better results than the MSFT models. The AAPLE models have the smallest values of RMSE, MAPE, and MAE and the determination R^2 has a much closer fitted result on the AAPLE dataset. Fig. 6, Fig. 7 and Fig.8 display the actual test data versus predicted test data graphs of three models on two different stock prices for one-day ahead prediction. The red lines indicate the predicted test data, and the black lines indicate the real test data for MSFT stocks. The orange lines indicate the predicted test data and the black lines indicate the real test data in the AAPLE stock prices.

It can be seen that, generally, all of these models perform relatively well, demonstrating that the past prices of stocks have predictive power and can be used to predict future prices. The predicted output fairly overlaps the target output, indicating a good prediction. Additionally, the turning points are forecasted quite timely. When there is a trend in the actual price, the predicted value follows accordingly and closely. It is also clear that the models that were proposed using the AAPLE dataset have better fitted results than the MSFT dataset.

Table (5). Results of the three methods

Dataset		AAPLE				MSFT			
Metrics		RMSE	MAE	MAPE	R^2	RMSE	MAE	MAPE	R^2
Models	BPNN	0.024895	0.020301	4.493746	0.988906	0.036203	0.03343	4.791764	0.982899
	RBF	0.012556	**0.009586**	2.076750	0.997178	0.030193	0.015742	1.506608	0.988105
	LSTM	**0.011353**	0.010190	**1.213605**	**0.997692**	**0.013394**	**0.010510**	**1.895189**	**0.997659**

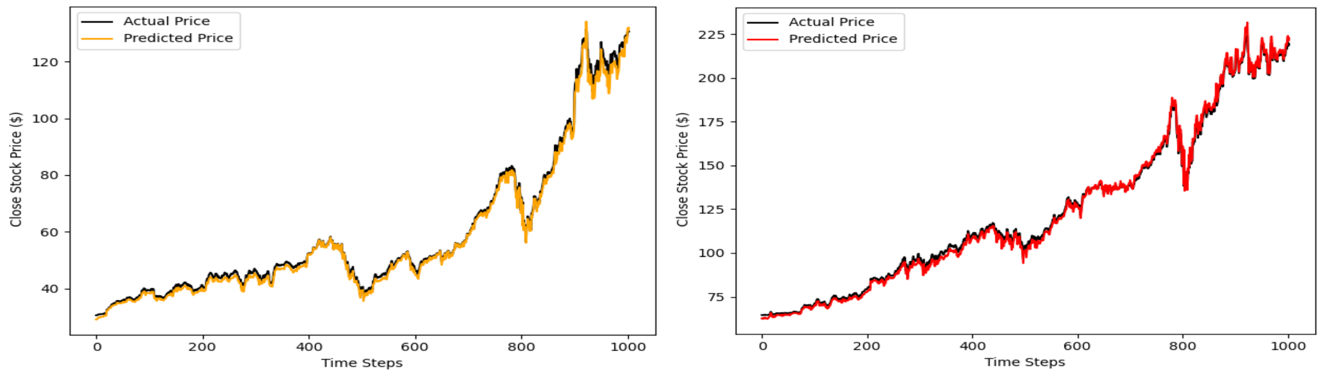

Figure (6). BPNN model predicted results.

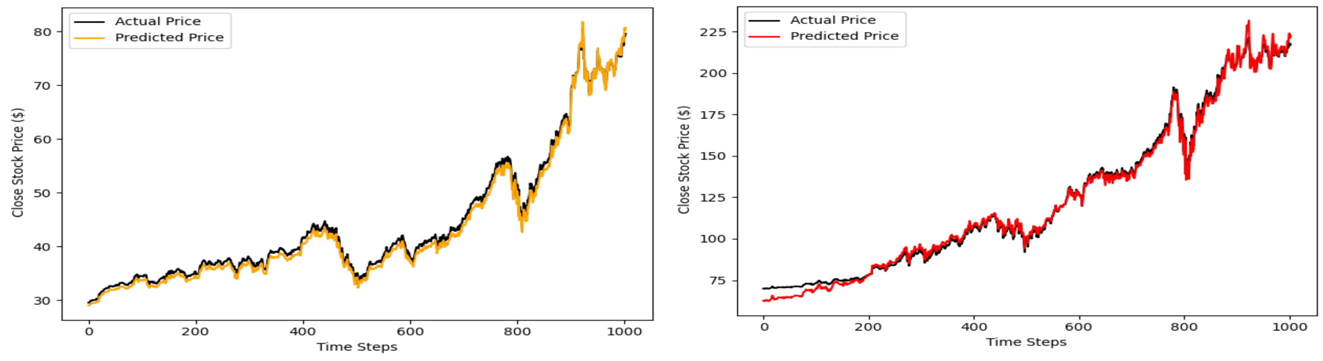

Figure (7). RBFNN model Predicted results.

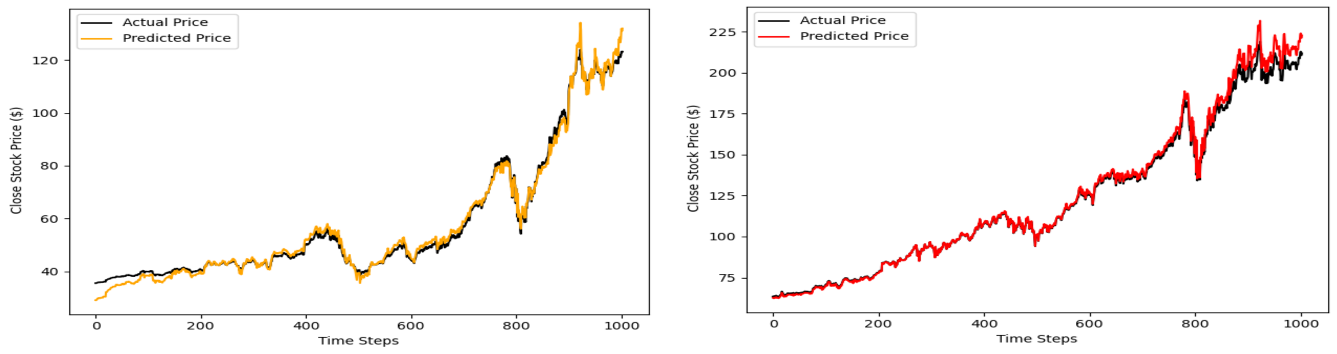

Figure (8). LSTM model predicted result.

6 Conclusions and Future Work

This study compares the performance of three neural network learning models, i.e., BPNN, RBFNN, and LSTM, by predicting movements in the one-day ahead of stock prices. The study address the following question: which neural network models provide the best predictive performance for both datasets? By implementing the proposed methods and checking the accuracy of the models using statistical errors, we conducted a comparative study between three NN models for predicting the stock market. With evidence from the forecast accuracy of two stocks' close prices. We find that all techniques perform well with acceptable accuracy, but the LSTM model beats other models in the prediction of the close prices on two datasets, and its performance was followed by that of RBFNN model. Which indicates that it may be conceivable to utilize the LSTM model as an effective approach to successfully predict the future pattern of stock prices based on the results of this study, the following recommendations can be made:

- Deep neural network models is better than the other techniques that have been utilized in this study. Researchers and investors are recommended to employ these methods for predicting the stock price.
- We recommend using hybrid models such as optimized LSTM with optimization techniques such as Particle Swarm Optimization (PSO) and combining BPNN and RBNN with other AI methods such as Fuzzy Logic (FL), Support Vector Machine (SVM), and Genetic Algorithm (GA), a lot of research work has been found that hybrid models improve stock prediction accuracy.
- The feature selection step should be taken under consideration in future work and compared with the results obtained in this study. It is suggested that the researchers use feature selection algorithms to extract the features of stock prices, such as Deep Belief Networks (DBN), the Discrete Wavelet Transformation (DWT) technique, Relief, maximum Relevance and Minimum Redundancy (mRMR), and LASSO. Such techniques successfully remove certain types of noise from data and improve the quality of the neural network model.

Conflict of Interest: The authors declare that there are no conflicts of interest.

References

ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G. & ISARD, M. {TensorFlow}: a system for {Large-Scale} machine learning. 12th USENIX symposium on operating systems design and implementation (OSDI 16), 2016. 265-283.

ALFRED, R., OBIT, J. H., AHMAD HIJAZI, M. H. & AG IBRAHIM, A. A. 2015. A performance comparison of statistical and machine learning techniques in learning time series data. *Advanced Science Letters,* 21**,** 3037-3041.

BINKOWSKI, M., MARTI, G. & DONNAT, P. Autoregressive convolutional neural networks for asynchronous time series. International Conference on Machine Learning, 2018. PMLR, 580-589.

BOTUNAC, I., PANJKOTA, A. & MATETIC, M. The importance of time series data filtering for predicting the direction of stock market movement using neural networks. Proceedings of the 30th DAAAM International Symposium, 2019. 0886-0891.

CHEN, L., QIAO, Z., WANG, M., WANG, C., DU, R. & STANLEY, H. E. 2018. Which artificial intelligence algorithm better predicts the Chinese stock market? *IEEE Access,* 6**,** 48625-48633.

CHEN, Z., LI, C. & SUN, W. 2020. Bitcoin price prediction using machine learning: An approach to sample dimension engineering. *Journal of Computational and Applied Mathematics,* 365**,** 112395.

GAO, P., ZHANG, R. & YANG, X. 2020. The application of stock index price prediction with neural network. *Mathematical and Computational Applications,* 25**,** 53.

GERS, F. A., SCHMIDHUBER, J. & CUMMINS, F. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation,* 12**,** 2451-2471.

GRAVES, A. & GRAVES, A. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks***,** 37-45.

HAŠKOVÁ, S., ŠULEŘ, P. & KUCHÁR, R. 2023. A Fuzzy Multi-Criteria Evaluation System for Share Price Prediction: A Tesla Case Study. *Mathematics,* 11**,** 3033.

HIRANSHA, M., GOPALAKRISHNAN, E. A., MENON, V. K. & SOMAN, K. 2018. NSE stock market prediction using deep-learning models. *Procedia computer science,* 132**,** 1351-1362.

JAFARI, A., KHALILI, T., BABAEI, E. & BIDRAM, A. 2019. A hybrid optimization technique using exchange market and genetic algorithms. *Ieee Access,* 8**,** 2417-2427.

JIANG, C. & LI, Y. 2019. Health big data classification using improved radial basis function neural network and nearest neighbor propagation algorithm. *IEEE Access,* 7**,** 176782-176789.

KETKAR, N. 2017. Introduction to keras. *Deep learning with Python.* Springer.

KURANI, A., DOSHI, P., VAKHARIA, A. & SHAH, M. 2023. A comprehensive comparative study of artificial neural network (ANN) and support vector machines (SVM) on stock forecasting. *Annals of Data Science,* 10**,** 183-208.

LIAO, Y., FANG, S.-C. & NUTTLE, H. L. 2003. Relaxed conditions for radial-basis function networks to be universal approximators. *Neural Networks,* 16**,** 1019-1028.

MANSOR, M. A., MOHD JAMALUDIN, S. Z., MOHD KASIHMUDDIN, M. S., ALZAEEMI, S. A., MD BASIR, M. F. & SATHASIVAM, S. 2020. Systematic boolean satisfiability programming in radial basis function neural network. *Processes,* 8**,** 214.

MANSOURI, A., NAZARI, A. & RAMAZANI, M. 2016. A comparison of artificial neural network model and logistics regression in prediction of companies' bankruptcy (A case study of Tehran stock exchange). *International Journal of Advanced Computer Research,* 6.

MEHRSAI, A., KARIMI, H.-R., THOBEN, K.-D. & SCHOLZ-REITER, B. 2013. Application of learning pallets for real-time scheduling by the use of radial basis function network. *Neurocomputing,* 101**,** 82-93.

MOGHAR, A. & HAMICHE, M. 2020. Stock market prediction using LSTM recurrent neural network. *Procedia Computer Science,* 170**,** 1168-1173.

MOKHTARI, S., YEN, K. K. & LIU, J. 2021. Effectiveness of artificial intelligence in stock market prediction based on machine learning. *arXiv preprint arXiv:2107.01031*.

MORADI, M., JABBARI NOOGHABI, M. & ROUNAGHI, M. M. 2021. Investigation of fractal market hypothesis and forecasting time series stock returns for Tehran Stock Exchange and London Stock Exchange. *International Journal of Finance & Economics,* 26**,** 662-678.

MUSTAFIDAH, H., PUTRI, C., HARJONO, H. & SUWARSITO, S. The most optimal performance of the Levenberg-Marquardt algorithm based on neurons in the hidden layer. Journal of Physics: Conference Series, 2019. IOP Publishing, 066099.

NABIPOUR, M., NAYYERI, P., JABANI, H., SHAHAB, S. & MOSAVI, A. 2020. Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis. *IEEE Access,* 8**,** 150199-150212.

NANDY, S., SARKAR, P. P. & DAS, A. 2012. An improved Gauss-Newtons method based back-propagation algorithm for fast convergence. *arXiv preprint arXiv:1206.4329*.

NIKOU, M., MANSOURFAR, G. & BAGHERZADEH, J. 2019. Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithms. *Intelligent Systems in Accounting, Finance and Management,* 26**,** 164-174.

NUGALIYADDE, A., SOHEL, F., WONG, K. W. & XIE, H. Language modeling through Long-Term memory network. 2019 international joint conference on neural networks (IJCNN), 2019. IEEE, 1-6.

QU, Y. & ZHAO, X. Application of LSTM neural network in forecasting foreign exchange price. Journal of Physics: Conference Series, 2019. IOP Publishing, 042036.

RAMESH, V., BASKARAN, P., KRISHNAMOORTHY, A., DAMODARAN, D. & SADASIVAM, P. 2019. Back propagation neural network based big data analytics for a stock market challenge. *Communications in Statistics-Theory and Methods,* 48**,** 3622-3642.

RUBI, M. A., CHOWDHURY, S., RAHMAN, A. A. A., MEERO, A., ZAYED, N. M. & ISLAM, K. A. 2022. Fitting multi-layer feed forward neural network and autoregressive integrated moving average for Dhaka Stock Exchange price predicting. *Emerging Science Journal,* 6**,** 1046-1061.

SHAH, D., ISAH, H. & ZULKERNINE, F. 2019. Stock market analysis: A review and taxonomy of prediction techniques. *International Journal of Financial Studies,* 7**,** 26.

SIAMI-NAMINI, S., TAVAKOLI, N. & NAMIN, A. S. A comparison of ARIMA and LSTM in forecasting time series. 2018 17th IEEE international conference on machine learning and applications (ICMLA), 2018. IEEE, 1394-1401.

SOHRABI, P., JODEIRI SHOKRI, B. & DEHGHANI, H. 2023. Predicting coal price using time series methods and combination of radial basis function (RBF) neural network with time series. *Mineral Economics,* 36**,** 207-216.

SONG, Y.-G., ZHOU, Y.-L. & HAN, R.-J. 2018. Neural networks for stock price prediction. *arXiv preprint arXiv:1805.11317*.

TIWARI, R., SRIVASTAVA, S. & GERA, R. 2020. Investigation of artificial intelligence techniques in finance and marketing. *Procedia Computer Science,* 173**,** 149-157.

VIJH, M., CHANDOLA, D., TIKKIWAL, V. A. & KUMAR, A. 2020. Stock closing price prediction using machine learning techniques. *Procedia computer science,* 167**,** 599-606.

WU, X. & WILAMOWSKI, B. M. Advantage analysis of sigmoid based RBF networks. 2013 IEEE 17th International Conference on Intelligent Engineering Systems (INES), 2013. IEEE, 243-248.