A New Approach in Feature Generation for Time Series Modeling and Forecasting

Anthony Nwachukwu*

***Department of Mathematics, University of Ibadan**

Abstract

The modeling and forecasting of Time Series has great applications in life events particularly in Stock market prediction. Many mathematical models have been developed as well as Machine learning models. Particularly, Artificial Neural Network (ANN) from Machine learning has been gaining much ground and this requires input features in order to obtain the desired output (predicted values). Over the years, these features have been the data provided by the market industries. The aim of this work is to get the predicted values from mathematical models and use them as input features for the ANN. It was discovered that predicted values from the ANN given these new input features gave a better yield than when learning from the features provided by the market only.

KEYWORDS: Modeling, Forecasting, Artificial Neural Network, Machine Learning, Feature Generation, Prediction.

Introduction

Stock market time series prediction is gaining significant traction from mathematical analysts, investors and machine learning professionals. Many mathematical and stochastic models have been developed and are all geared toward making better predictions. Brownian motion and Geometric Brownian motion are one of the widely used models in stochastic analysis in making time series prediction, with both having their merits and demerits.

Many Machine learning algorithms have been employed in these predictions among which Artificial Neural Network stands out due to its capacity to do better on big data than most other Machine learning models. Machine learning models have the capacity of learning from many input data commonly known as input features. The input features for stock market prediction usually include daily open price, daily closing price, daily highest price, daily lowest price, daily volume and daily turnover. The output data is usually the closing price of the next trading day. There are also several ways of generating more features from existing ones like data augmentations.

This paper presents a new way of feature generation, using the predictions from mathematical models like the predictions from Brownian motion. It proposes either adding it to the existing features or replacing the existing features with it.

The data used for this work is the GOOGLE stock market index which can be downloaded on yahoo Finance and Artificial Neural Network (ANN) model was used in training the features.

Research Methodology

1. Brownian Motion Modelling with Adaptive Parameters

We shall employ Brownian motion process to model the randomness of stock price. A stock price process $(S_t, \mathrm{t} \geq 0)$ is represented by the stochastic differential equation (SDE) as shown below

$$dS_t = S_t(\mu dt + \sigma dW_t). \qquad (1)$$

Where the parameters μ and σ in (1) are adaptive and are also the rate of return and the volatility, respectively which we will estimate from the data used. The process $(W_t, \mathrm{t} \geq 0)$in (1) is a standard Brownian motion. The stochastic differential equation (1) is driven by the Brownian motion process $(W_t, \mathrm{t} \geq 0)$.

1.1. Parameter Estimation

The data of Google stock indices from 26th June 2006 to 11th June 2018 making it a total of 3011 data samples are used in this paper. The columns consist of Date, High, Low, Close, Adj Close and Volume of the market. The column titled "Close" is the response we seek to model. This data is divided into two sets. Eighty percent of the data was used for the parameter estimation while the remaining 20% was used to select the mu and sigma that produces the least error.

To estimate the rate of return and volatility, the following steps were taken:

$$\Delta S_j = S_{j+1} - \Delta S_j \qquad j = 1,2,\dots,n-1 \quad (2)$$

$$R_j = {\Delta S_j}/{S_j} \qquad j = 1,2,\dots,n-1 \quad (3)$$

$$\mu_i = \frac{1}{6+i}\sum_{j=1}^{6+i} R_j \qquad i = 1,\dots,m \quad (4)$$

$$\sigma_i = \sqrt{\frac{1}{(6+i)-1}\sum_{j=1}^{6+i}\left(R_j - \bar{R}\right)^2} \qquad i = 1, \dots, m \qquad (5)$$

Where $\bar{R}$ is the average of R_j, n is the length of the data under consideration and is given by,

$$m = length\_of\_R[\, floor\left(\frac{n-1}{7}\right) : end] \qquad (6)$$

The above parameters were computed from the dataset using MATLAB. μ_i and σ_i are generated by computing for 7 days, followed by 8 days, till you get to end of the R.

1.2. Modelling Brownian Motion

In this paper, the Euler discretization method is applied to solve the SDE in(1). The solution of this SDE is denoted by S_j and given by the equation below;

$$S_{j+1} = S_j + \mu_j S_j \Delta t + \sigma_j S_j \Delta W_j\,, \qquad (7)$$

where j is the time index $(j = 0, \dots., N)$, N is the length of the datasets, Δt is a sampling time and μ_j and σ_i are estimated in the previous section. The initial value S_0 is the initial stock price in the dataset. The term ΔW_j is approximated as

$$\Delta W_j = Z_j\sqrt{\Delta t} \qquad (8)$$

The random variable Z_j is the standard normally distributed random variable with mean = 0 and variance = 1. It is generated by method of Box and Muller [reference 8 of material]. The stock prices calculated from (7) are simulated by the MATLAB program given in appendix 1. These simulated data are compared with the second dataset for model validation. The various μ_i and σ_i from (4) and (5) were used for the simulation and the errors were computed using the Mean Square Error Formula given below;

$$MSE = \frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2 \qquad (9),$$

where n is the number of datasets, y_j is the empirical price (market price), and $\hat{y}_j$ is the model price (simulated price).The stock price predicted with μ_i and associated σ_i that has the least RMS was adopted. The graph of the empirical price and the model price is shown in **fig 1**. Equations (2) *to* (9) where all implemented in MATLAB.

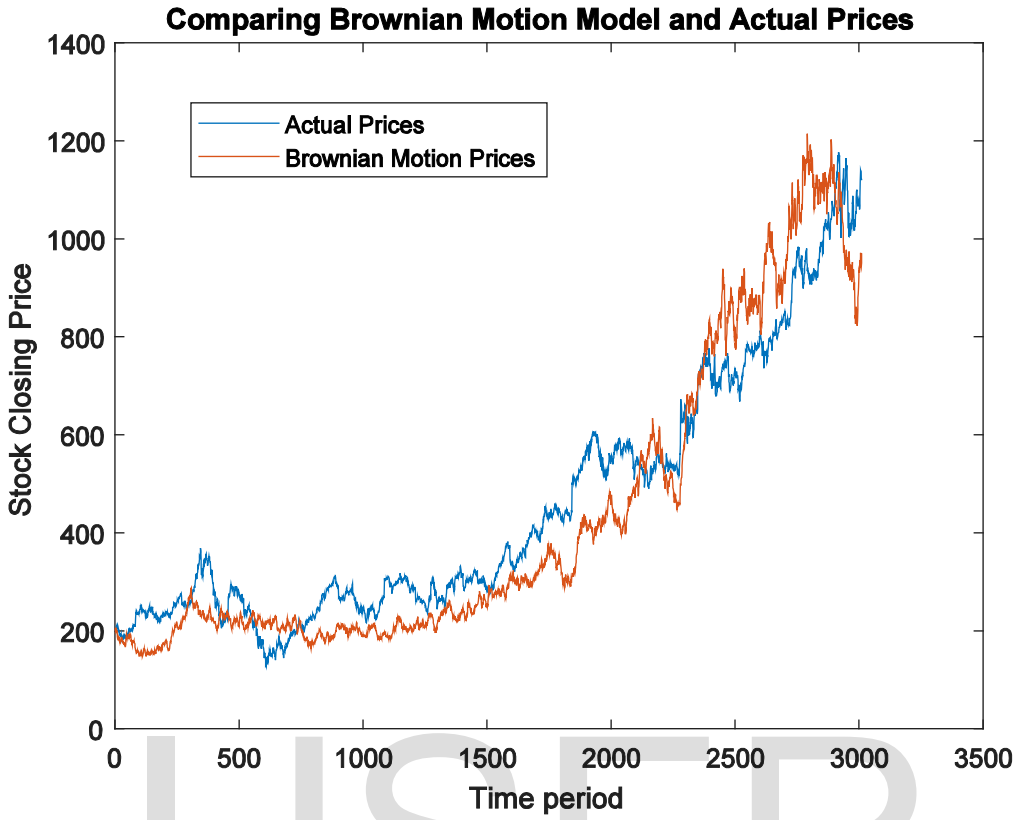

Fig. 1: Empirical price and the model price

2. Artificial Neural Network Modeling of Stock Prices

2.1 Overview of ANN

Artificial Neural Network (AAN) is a system inspired by biological concept with several single processing elements known as neurons. There exists an interconnection among the neurons which consists of sets of assigned weights. Each neuron receives signals from outside sources or other neurons and then processes them in activation function to produce its output before sending it to other neurons.

Each input impact is different from other inputs. The ANN under consideration has three layers: input layer, output layer, and hidden layer. Each neuron takes the values of inputs parameters, sums them up according to the weights assigned to them, and adds a bias. By applying the transfer function, the value of the outputs is then determined. The number of neurons in input layer corresponded to the number of input parameters. The architecture of a typical ANN with multilayer is presented in **fig 2** below:

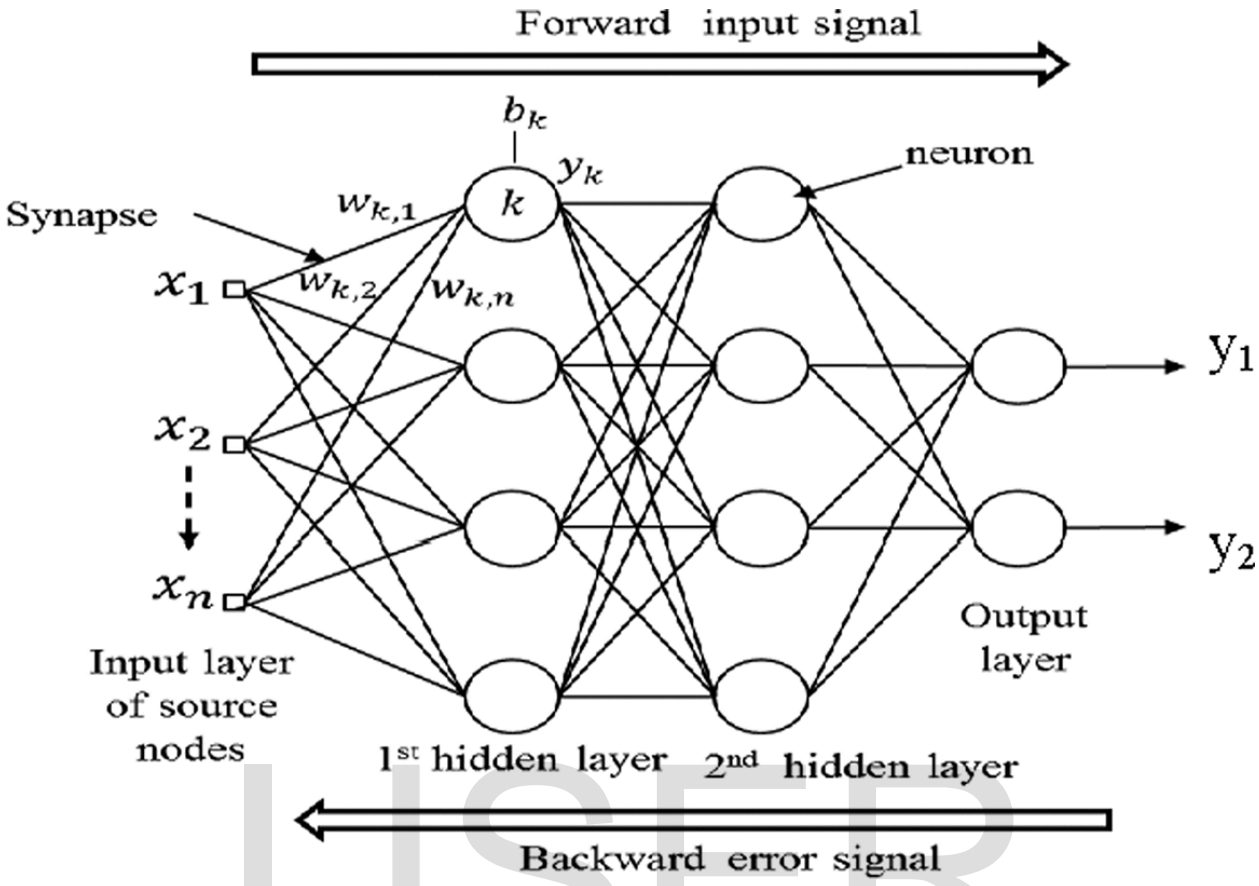

Fig. 2: ANN Diagram

In mathematical terms, the performance of neuron P can be described as follows:

$$u_p \sum_{i=1}^{n} W_{pi} X_i \qquad (10)$$

$$y_p = \varphi(u_p + b_p) \qquad (11)$$

Where $X_1, \dots, X_n$ are the input parameters; $W_{p1}, \dots, W_{pn}$ are the weights assigned to neuron, p; u_p is the input combiner; b_p is the bias; φ is the activation function; and $y_p$is the output of the neuron.

2.2 Predicting stock price with ANN

In this study, the short-term historical stock prices with the days of the week as inputs were used. The training was done on two sets of inputs. The first input set contains the Open, High, Low, Adj Close and Volume of the market data given. The second set of inputs contains the given market data as in the first with addition to their predicted figures from the Brownian motion model given in (7). This second set of

input sets differentiates this work from the norm. The aim of this work is to compare the results from this second input sets with that of the first.

The mathematical equations governing these processes are given as:

$$y(k) = g\big(y(k-1), y(k-2), \dots, y(k-n)\big) \tag{12}$$

$$y''(k) = g\big(y(k-1), y(k-2), \dots, y(k-n), y'(k-1), y'(k-2), \dots, y'(k-n)\big) \tag{13}$$

Where $y(k)$ is the stock price at time k using the first set of inputs and $y''(k)$ is the stock price at time k using the second set of inputs, n is the number of historical days, and $y'(k-n)$ is the predicted input of $y(k-n)$ using $eqn.\ (7)$.

The data of Google stock indices from 26th June 2006 to 11th June 2018 making it a total of 3011 data samples are used in this paper. The columns consist of Date, High, Low, Close, Adj Close and Volume of the market. The Date column was dropped and the other columns were adopted as the input features except the Close column which is the required Response. The data was divided into three partitions, the first which consists of 80% of the dataset (i.e. 2409) was used for training the model, 10% (i.e. 301) was used for validation while the raining 10% (i.e. 301) was used as the test set.

MATLAB software R2017a was used for training, validation and testing of this model. The performance of the ANNs was calculated using the mean square error (MSE) of the output. MSE represents the average squared difference between the predicted values and the actual values. MSE was determined using:

$$MSE = \frac{1}{n}\sum_{j=1}^{n}\big(y_j - \hat{y}_j\big)^2 \tag{14}$$

where y_j and $\hat{y}_j$ are the actual and predicted values respectively and n is the total number of data Used. The network was trained with 10 neurons and a one day delay time series data as shown in the figure below:

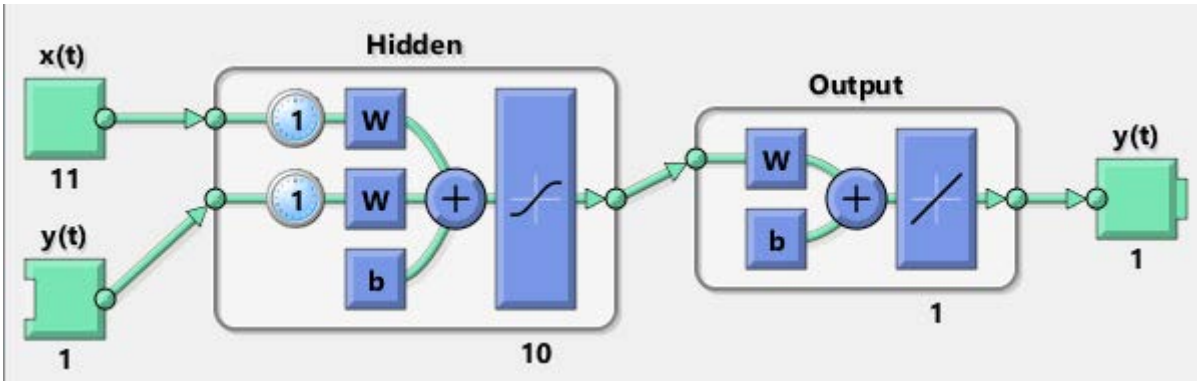

Fig. 3: ANN Model on where the training was done

Result and Discussion

This section presents the results of the trainings on MATLAB ANN Application.

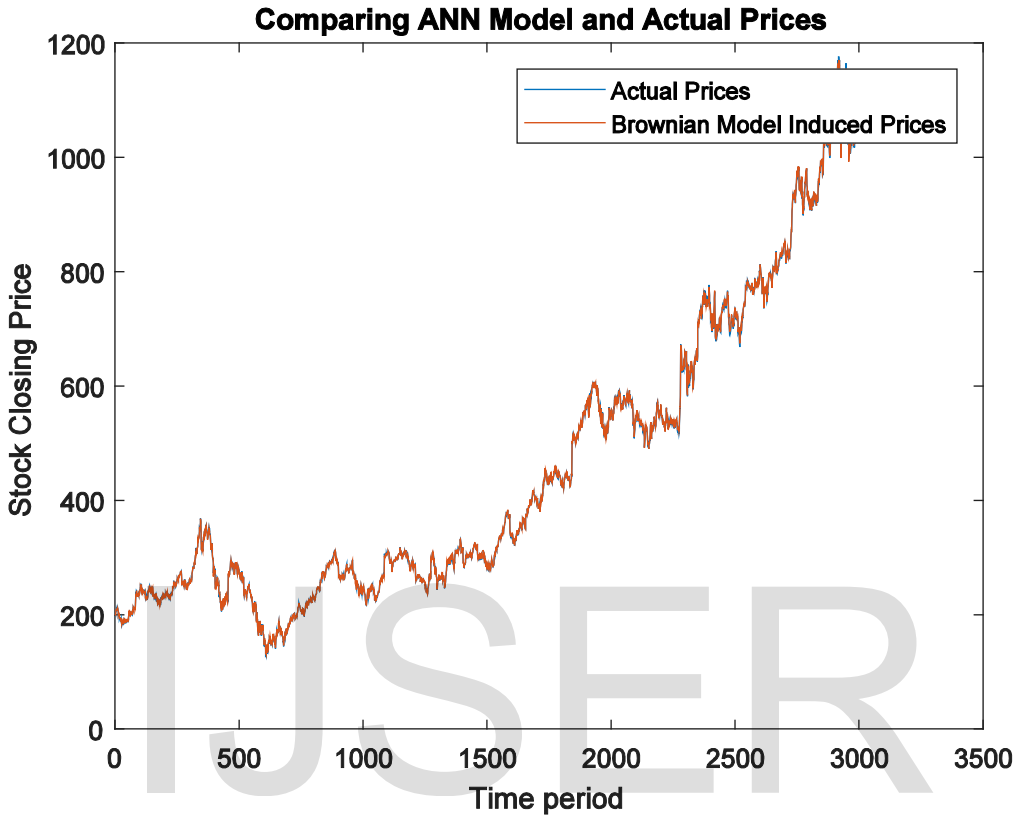

Fig. 4: Plot of ANN with Brownian motion Model as Input Feature and the Actual Data

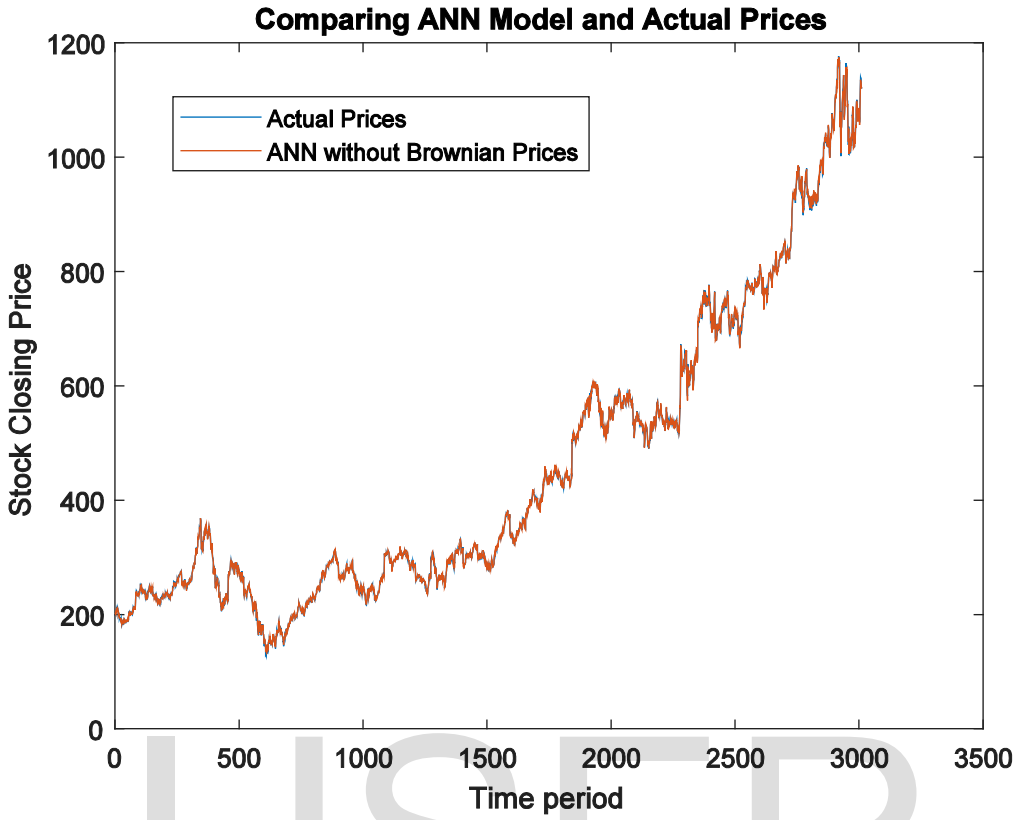

Fig. 5: Plot of ANN without Brownian motion Model as Input Feature and the Actual Data

Table 1: Model Statistics for the Stock Price Prediction

	R		MSE	
	ANN without Brownian Input	**ANN with Brownian Input**	**ANN without Brownian Input**	**ANN with Brownian Input**
Train	0.99957	0.99955	56.06	60.37
Validation	0.99940	0.99961	76.21	46.33
Test	0.99966	0.99969	44.36	42.50

From the plots of figure 4 and 5, it is difficult to tell the impact of the Brownian motion and even the errors in the predicted values. The figures in table 1 show these differences. The ANN model without prediction from Brownian motion as input features does better on the training and test sets but poor on

the validation set as seen in the values of R and MSE. While the ANN model with prediction from Brownian motion as input features does better on the validation and test sets hence generalizes the model well. Also the performance of the ANN model with prediction from Brownian motion as input features on the validation and test sets is better than the ANN model without Brownian motion hence it gives better predictions on a data set it has not seen before. Also figure 1, 4 and 5 show that ANN performs better than Brownian motion.

Conclusion

From the above analysis we can conclude that adding Brownian motion predicted responses as input feature to ANN model will result to a better performance in stock market price predictions. Also it can serve as a regularization term since the addition reduced the performance of the train set and increased the performance of the test set.

Future Work

Further work can be done by adding more than one predicted responses from mathematical models. One could even consider totally replacing the whole input features with the ones generated by mathematical models.

References

Areerak, T. (2014). Mathematical Model of Stock Prices via a Fractional Brownian Motion with Adaptive Parameters. *ISRN Applied Mathematics*, 1-7.

Chakravorty, T. (2018, June 21). *How Machine Learning Works: An Overview*. Retrieved from The New Stack: https://thenewstack.io/how-machine-learning-works-an-overview/

Moghaddam, A. H. (2016). Stock market index prediction using artificial neural network. *Journal of Economics, Finance and Administrative Science*, 89-93.

Olden, M. (2016). *Predicting Stocks with Machine Learning.* Oslo: Department of Informatics, University of Oslo.

Ramezanian, M. R. (2011). Combination Neural Network and Financial Indices for Stock Price Prediction. *Journal of Applied Sciences*, 3429-3435.

Wang, Y. (2014). Stock price direction prediction by directly using prices data: an empirical study on the KOSPI and HSI. *Int. J. Business Intelligence and Data Mining*, 45–160.