Asia Pac. J. Math. 2025 **12**:54

AN EFFECT OF HURST EXPONENT ON PREDICTING THAI STOCK MARKET

MALINEE CHAIYA*, PASSAWAN NOPPAKAEW, THANAKORN PRINYASART

Department of Mathematics, Faculty of Science, Silpakorn University, Thailand

*Corresponding author: chaiya_m@su.ac.th

Received April. 7, 2025

Abstract. Any time series can be classified by Hurst exponent. If Hurst exponent is greater than 0.5, then the time series has a persistent behavior. While it has anti-persistent behavior when Hurst exponent is less than 0.5. In this paper, we use Hurst exponent to determine the behavior of Thai stock market in different periods of time. We found that the periods with larger Hurst exponent have higher accuracy when predicted by Neural Networks.

2020 Mathematics Subject Classification. 91G15.

Key words and phrases. Hurst exponent; neural networks; Thai stock market; time series.

1. Introduction

Predicting the direction of stock prices is one of the topics that attracts significant interest from investors and researchers. The more precise the forecasting in unpredictable environments, the more effective the market strategies.

The Efficient Market Hypothesis (EMH) was proposed independently by Paul A. Samuelson and Eugene F. Fama in the 1960s. It is a concept in financial economics which states that the stock prices or assets in the market fully reflects all available information at that time. In [4], Fama presented compelling empirical evidence demonstrating that stock prices exhibit a random walk pattern. On the other hand, Lo and MacKinlay [5] examined the random walk hypothesis for weekly stock market return by comparing variance estimators from different types of frequency data. The results showed that the random walk hypothesis did not hold for all sample periods.

Identifying trends can improve the accuracy of stock price forecasting. Hurst exponent is a useful tool for indicating trends effectively. It was proposed by Harold Edwin Hurst [1] in the 1950s. He introduced Hurst exponent in his work in the area of hydrology while he was studying long-term

DOI: 10.28924/APJM/12-54

storage capacity of reservoirs and analyzing the Nile River's water flow patterns. The Hurst exponent method was later introduced and explored in other fields by Mandelbrot and Ness [2].

The Hurst exponent is a statistical tool used to assess the long-term memory of a time series. Understanding this memory is valuable in financial markets, as financial returns often deviate from the behavior of random walks. It helps to determine whether a time series (such as stock prices) is trending, mean-reverting, or exhibiting a random walk. The Hurst exponent H ranges between 0 and 1. If $0 < H < 0.5$, then the time series is mean reverting. If $H = 0.5$, then the time series is a random walk. If $0.5 < H < 1$, then the time series is trending. The strength of trend increases as H approaches 1.

Mitra [3] estimated the Hurst exponent of stock indices and examined the relationships between Hurst exponent and the predictability of financial time series. The study showed that Hurst exponent and the returns of stock indices from a trading rule are correlated and hence, Hurst exponent can be used as a measure to detect the trend in technical analysis.

Neural networks are widely deployed in modern finance to forecast the direction of classic financial time series data. The more precise the forecasting in unpredictable environments, the more effective the market strategies. In the past decades, there have been many researches that implement Neural networks to predict stock future closed prices. There are many research articles that utilize neural networks for predicting financial time series data. In [6], Sako et al. used neural networks to predict stock indices and currency exchange rates.

Noorbakhsh and Shaygani [7] applied five models, 2 hybrid models and three single models, in neural networks to predict stock prices of five companies listed on the Tehran Stock Exchange. The results demonstrated that the hybrid model which is CNN-LSTM model achieved higher accuracy than the other models.

Qian and Rasheed [15] investigated the effect of the Hurst exponent on classifying series of financial data from different periods of time. The data of Dow-Jones daily return was used in this research. By using backpropagation Neural Networks, the result showed that the time series data with large Hurst exponent can be predicted more accurately than those series with Hurst exponent close to 0.50.

The Random Walk Hypothesis suggests that log returns of stock prices follow a random walk. Each log return is independent of the previous log returns and the mean and variance remain constant over time. If log returns truly follow a random walk, it would be difficult to consistently predict future stock prices based on the past price movements. Moreover, it would be ineffective to use Technical analysis to identify patterns in the price charts.

In this work, we would like to see the effect of Hurst exponent in predicting the daily log return of stock indices. Here, we used the SET index data from January 2, 1997 to December 30, 2024. We divided our work into two parts. The first part utilized Hurst exponent to understand trends of the daily returns. Then we divided the data into two groups, one with large Hurst exponent, and the other

with Hurst exponent close to 0.5. The second part utilized Neural networks to predict daily returns of the assets in each group and compare the accuracy of the prediction. We found that the larger Hurst exponent, the smaller the error obtained.

2. Hurst Exponents

In this work, we use the R/S analysis to calculate the Hurst exponent of the time series. For an integer $N \geq 5$, let $X_1, X_2, \ldots, X_{2^N}$ be a time series of length 2^N. For each $n \in \{2^4, 2^5, \ldots, 2^N\}$ and $k \in \left\{0, 1, \ldots, \frac{2^N}{n} - 1\right\}$, let

$$m_{n,k} = \frac{1}{n} \sum_{i=1}^{n} X_{kn+i}$$

and for $t \in \{1, 2, \ldots, n\}$ let

$$Y_{n,k,t} = X_{kn+t} - m_{n,k} \quad \text{and} \quad Z_{n,k,t} = \sum_{i=1}^{t} Y_{n,k,i}.$$

Next we calculate $R(n,k)$ and $S(n,k)$ as follows:

$$R(n,k) = \max_{t \in \{1,2,\ldots,n\}} (Z_{n,k,t}) - \min_{t \in \{1,2,\ldots,n\}} (Z_{n,k,t})$$

and

$$S(n,k) = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} Y_{n,k,i}^2}.$$

Then define

$$\frac{R}{S}(n) = \frac{n}{2^N} \sum_{i=1}^{\frac{2^N}{n}} \frac{R(n,i-1)}{S(n,i-1)}.$$

Plot $\log \frac{R}{S}(n)$ as a function of $\log n$. After fitting a straight line to the data, the Hurst exponent, denoted by H, is the slope of the line. Figure 1 shows an example of the straight line fitted with the data from 08/02/2007 to 22/04/2011. The Hurst exponent in this example is about 0.6583.

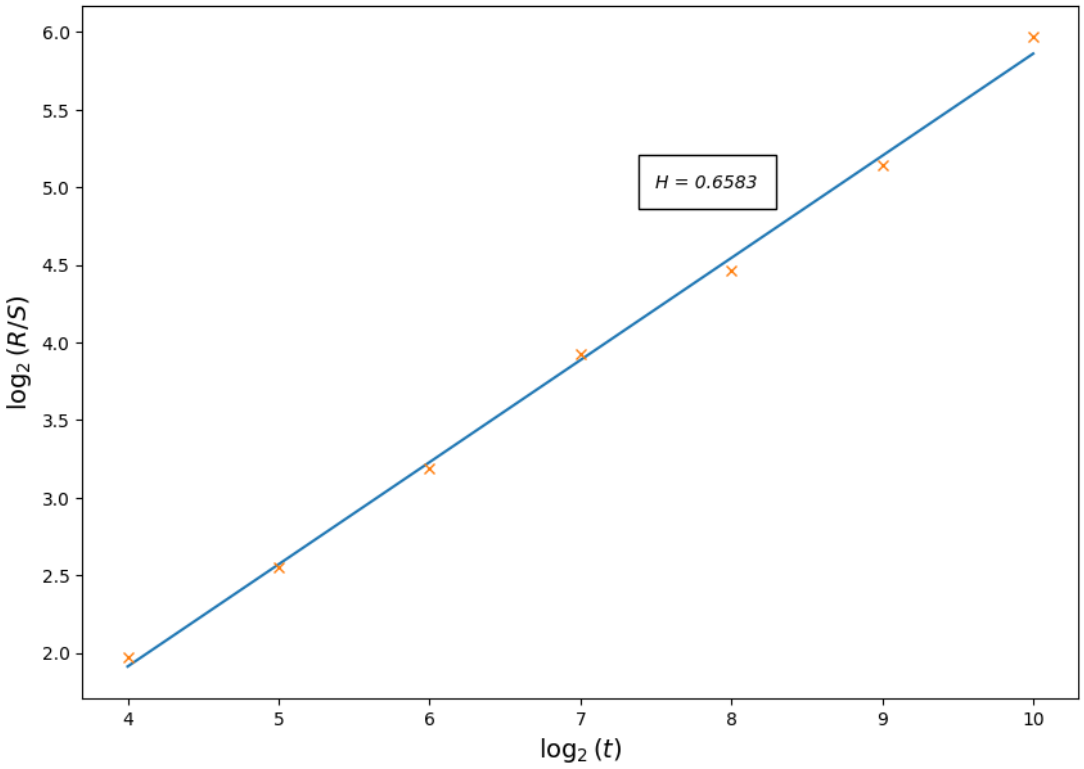

FIGURE 1. The straight line fitted with the data from 08/02/2007 to 22/04/2011

In this research, we analyze the log returns of the SET index over the period from January 2, 1997, to November 30, 2024. For each rolling window of 1024 trading days, we computed the Hurst exponent as shown in Figure 2. In the next section, we will show that there are underlying structures in rolling windows with high Hurst exponent.

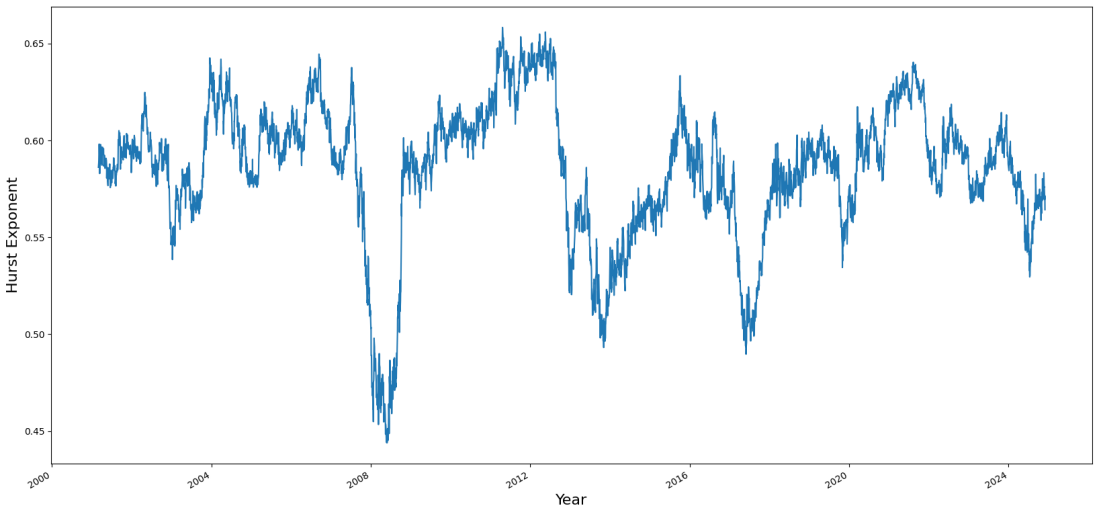

FIGURE 2. Rolling Hurst Exponent of the daily return of SET index from 02/01/1997 to 30/12/2024

3. Hurst exponent of standard Gaussian time series

For standard Gaussian time series, Anis and Lloyd [19] gave a formula to estimate the expectation of $\frac{R}{S}(t)$ as follows:

$$E\left(\frac{R}{S}(t)\right) \approx \frac{\Gamma\left(\frac{t-1}{2}\right)}{\sqrt{\pi}\cdot\Gamma\left(\frac{t}{2}\right)} \sum_{i=1}^{t-1} \sqrt{\frac{t-i}{i}}. \tag{1}$$

We approximate the expectation of $\frac{R}{S}(t)$ for $t \in \{2^4, 2^5, \ldots, 2^{10}\}$ and apply linear regression at significance level $\alpha = 0.05$. Results are provided in Table 1.

$\log_2(t)$	$\log_2\left(E\left(\frac{R}{S}(t)\right)\right)$
4	2.012758404
5	2.600624317
6	3.164394909
7	3.710466641
8	4.243598131
9	4.767327524
10	5.284267119
Regression slope (H)	$0.543826278 \pm 0.01395367$

Table 1. The Hurst exponent calculated by using formula (1)

To confirm the result, we used Monte Carlo simulations to estimate the Hurst exponent of standard Gaussian time series. Specifically, we generated 10,000 standard Gaussian time series of length 1024 and calculated the average and standard deviation of their Hurst exponents. This process was repeated 10 times, and we took the mean of the averages and standard deviations across all repetitions. With 95% confidence, we determined that the Hurst exponent of a standard Gaussian time series falls between $0.545528 - (1.96 \times 0.048709) = 0.450058$ and $0.545528 + (1.96 \times 0.048709) = 0.640998$.

	Simulated Hurst Exponent	Standard deviation
1	0.546583	0.049243
2	0.545754	0.048507
3	0.545521	0.049578
4	0.545464	0.048378
5	0.545623	0.048401
6	0.544929	0.048624
7	0.545539	0.048322
8	0.545610	0.048438
9	0.545097	0.048808
10	0.545164	0.048790
Mean	**0.5455284**	**0.0487089**
Std.	**0.000454749**	

TABLE 2. Hurst exponent calculated by using Monte Carlo methods

To confirm the presence of underlying structures in periods with high Hurst exponents, we randomly selected a period with a Hurst exponent greater than 0.645. For this period, we shuffled the data order 500 times and then calculated the Hurst exponent of the shuffled data. We repeated this process for 10 times and recorded the result in Table 3. Then we calculated the average of the Hurst exponents.

	Hurst exponent after shuffling	Standard deviation
1	0.549713	0.051802
2	0.552230	0.046106
3	0.548959	0.049081
4	0.547096	0.046547
5	0.548084	0.047139
6	0.549542	0.048883
7	0.549175	0.048774
8	0.550917	0.048263
9	0.548429	0.050287
10	0.547314	0.048827
Mean	**0.549146**	**0.048571**

TABLE 3. The Hurst exponent of shuffled data

After shuffling, the Hurst exponent is close to 0.545528, which corresponds to the Hurst exponent of a standard Gaussian time series. This result indicates that the original periods contained meaningful structures that were eliminated by shuffling.

4. Neural networks

Currently, neural networks offer best solutions for many complex problems, especially the problems concerning with pattern recognition or prediction. It is then often used as a tool in forecasting time series data. A neural network is a computing system composing of interconnected simple processing nodes, called neurons or perceptrons, in a layered structure. Each node computes a weighted sum of its inputs from the previous layer and passes the result, transformed by its activation function, to connected nodes in the next layer. The classical feedforward neural network will let the information flow in only one direction, from inputs to outputs with no feedback loop, and the input of $i+1$th layer is the output of the ith layer. In particular, a feedforward neural network with a single hidden layer is considered to be able to approximate any continuous function [10]. The learning process of a neural network involves iterative adjustments of weights and biases to minimize the errors between its predicted outputs and the actual values. There are many learning algorithms in neural networks. One of the widely used algorithms is backpropagation. A feedforward neural network trained by backpropagation algorithm is called the feedforward backpropagation neural network. The backpropagation algorithm employs gradient descent to locate a local minimum of the error function. It determines the gradient, or partial derivative, of the error with respect to each weight. Moving in the opposite direction of these gradients, known as the steepest descent direction, leads to the fastest reduction in error. The algorithm updates the weights by following this steepest descent path. Many variations of backpropagation were introduced to optimize the direction and step size for improving the performance of the neural network (see [9, 11–14, 16] for more details). Once training is complete, the network can be used to predict new data.

4.1. **Data preparation.**

Using a rolling window of 1024 trading days, we computed the Hurst exponent for SET index daily return data spanning from 02/01/1997 to 30/12/2024. The calculation resulted in a total of 5087 rolling periods. Of these, 164 periods had Hurst exponents exceeding 0.645, while 159 periods showed Hurst exponents within the range of 0.52 to 0.535. The histogram in Figure 3 shows the distribution of Hurst exponents across all periods.

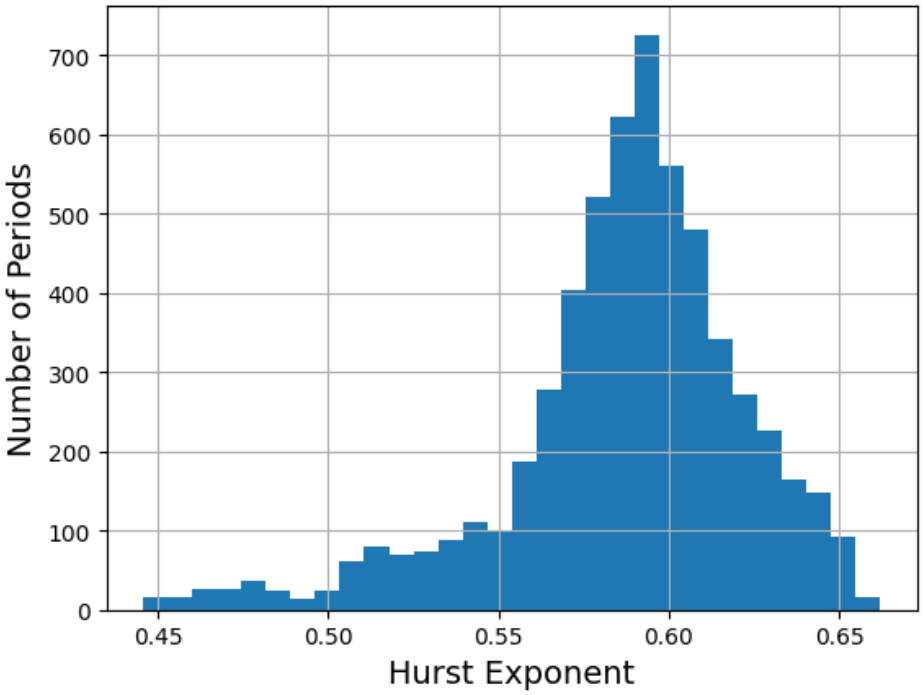

FIGURE 3. Histogram of all 5087 calculated Hurst exponents

4.2. **Defining the neural network structure.** Forecasting the future state of a time series is a complicate problem especially when the time series is non-linear or chaotic because it cannot be dealt with classical statistical methodology. To handle with this kind of time series, many tools are introduced (see [8,18]). However, the main idea of these tools is to firstly embed the problematic time series to higher-dimensional space which yield a new representation and then study the underlying dynamical structure from the new representation. This embedding is called time delayed embedding method and it is regularly used in the time series analysis and prediction. One of the delayed embedding techniques is introduced by Takens [17] in 1981. He proposed a theorem in which the main hypothesis is that, given a time series data $x_1, x_2, \ldots, x_i$, we can predict the value of x_{i+1} from the information contained in the time-delay vector $X_i = (x_i, x_{i+\tau}, x_{i+2\tau}, \ldots, x_{i+(d-1)\tau})$ if the values of d and τ are appropriate and $d \ll N$. In this way, we can represent the future point in the time series by points in the d-dimensional space. Here d is called the embedding dimension and τ is called the delay.

According to Takens' theorem, it requires two parameters which are the delay parameter τ, and the embedding dimension parameter d. An estimation of these parameters in one-dimensional time series can be done by calculating the auto-mutual information (AMI) function and the false nearest neighbor (FNN) function [18]. The delay and the embedding dimension are the first local minima of the AMI and FNN functions, respectively.

Figure 4 shows the graph of AMI function of the SET index data from 08/02/2007 to 22/04/2011 with Hurst exponent being equal to 0.6620. According to the figure, it estimates that the delay is equal to 1. We further used this information to calculate FNN and got the graph represented in Figure 5. The graph shows that the suggested embedding dimension is equal to 3. We calculated the AMI and FNN functions for our random 60 data sets. The delay is suggested to be 1. This coincides with the work in [15]. The embedding dimension is suggested to be 3. This coincide with the embedding dimension that gives the best prediction in [15]. However, we will also consider the embedding dimension 4 and 5.

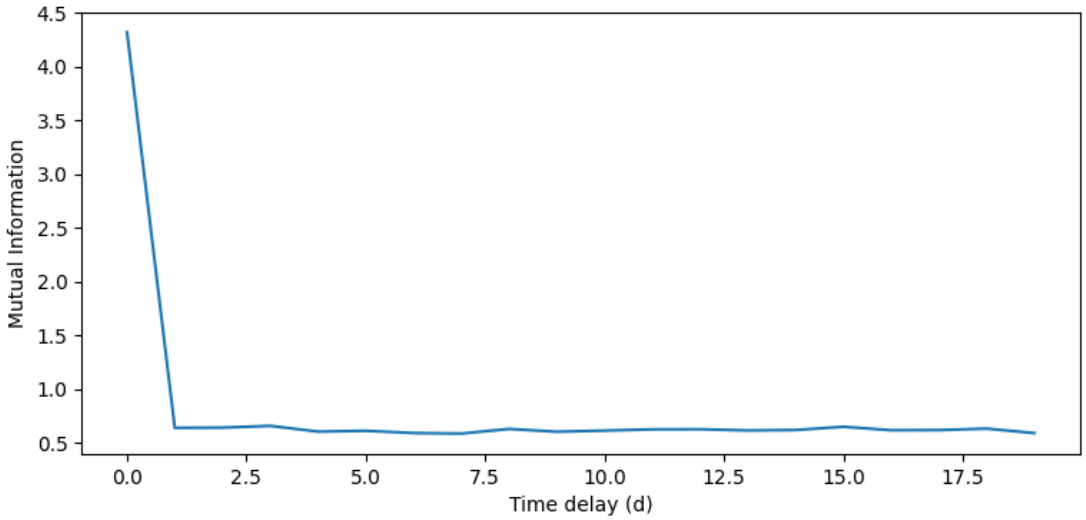

FIGURE 4. The auto-mutual information of the log return of the SET index from 08/02/2007 to 22/04/2011

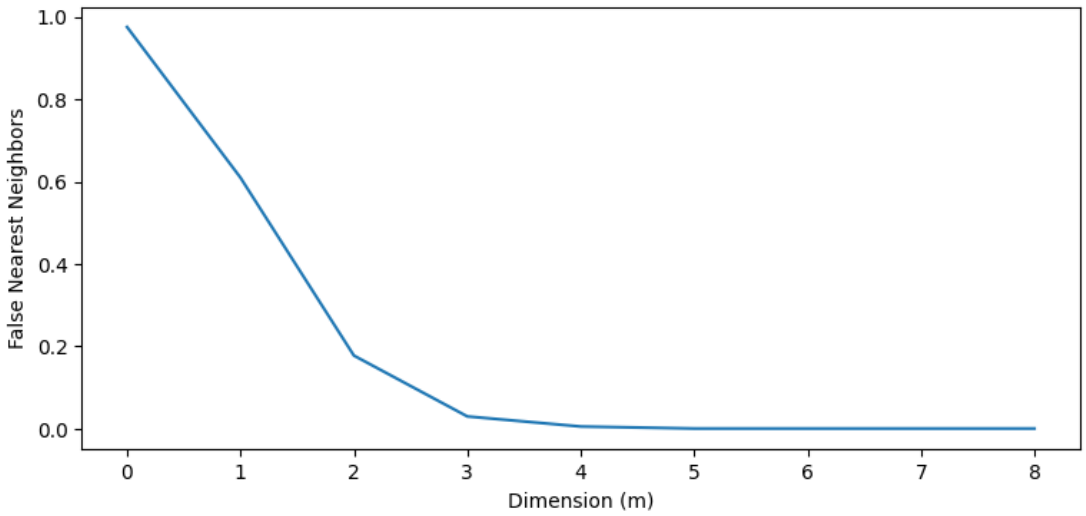

FIGURE 5. The false nearest neighbor of the log return of the SET index from 08/02/2007 to 22/04/2011 when $\tau = 1$

As aforementioned, most practical applications utilize networks with a single hidden layer [10] because multi-hidden-layer networks offer no significant advantage over their single-layer counterparts. In this research, we designed a neural network with a single hidden layer, using the suggested number of input nodes and one output node. The network was built using Keras's Sequential model and optimized with the Adaptive Moment Estimation (Adam) algorithm. The hidden layer employed the ReLU activation function, while the output layer used a linear activation function. We normalize the data before feeding the data to the models by using the Python StandardScaler function which is a technique for feature scaling in the preprocessing stage before training a model in neural networks. After normalizing, each feature in the dataset is rescaled such that the mean is adjusted to 0 and the standard deviation is adjusted to 1 by using the following equation:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma},$$

where X is the original value in the data set, μ is the mean of the feature, and σ is the standard deviation of the feature.

To mitigate overfitting, the dataset was divided into three subsets: training, validation, and testing. The training set was used to update the network's weights through error backpropagation, while the validation set was employed to halt training when the error on the validation set began to rise. The testing set was used to evaluate the network's predictive accuracy. Specifically, 60% of the data were allocated for training, 20% for validation, and the remaining 20% for testing. This approach ensures greater reliability in the network's predictive performance, as the testing data directly precede the forecasting period.

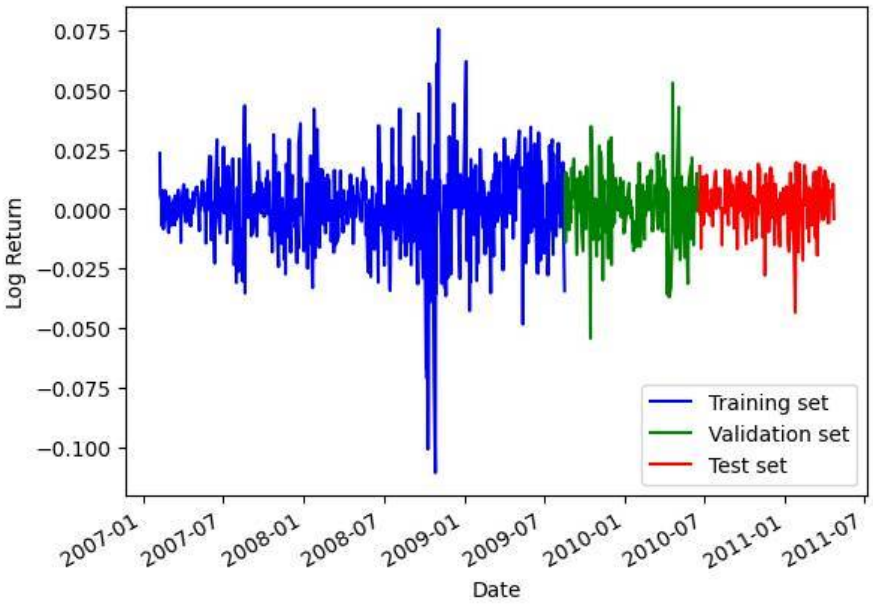

FIGURE 6. Log return data from 02/01/1997, to 30/12/2024, divided into three datasets

4.3. **Neural network construction and forecasting.** A practical guideline for deciding the hidden nodes is that the total degrees of freedom in the network should be 1.5 times the square root of the total data count. This leads to the following equation:

$$(\#\text{ input nodes}+1)(\#\text{ hidden nodes})+(\#\text{ hidden nodes}+1)(\#\text{ output nodes})=1.5\sqrt{\#\text{ data}}.$$

We observe that for dimensions 3, 4, and 5, the number of hidden nodes should be 10, 8, and 7, respectively. For each of these dimensions, five network configurations are tested, with hidden node counts varying slightly around the recommended number. For instance, for dimension 3, the configurations include 8, 9, 10, 11, and 12 hidden nodes. We randomly select five periods for training each network, running the training 100 times per network, and recording the lowest NRMSE value. NRMSE is defined as:

$$\text{NRMSE}=\frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(O_i-T_i)^2}}{\max_{i\le n}O_i-\min_{i\le n}O_i}. \tag{2}$$

In (2), O_i is an output value and T_i is a target value, where $i\in\{1,2,\ldots,n\}$. After the training, we obtained the following results shown in Table 4 to Table 6:

	Dimension 3, Hidden nodes				
	8	9	10	11	12
1	0.121892	0.120617	0.119863	0.158328	0.123228
2	0.156266	0.119484	0.120696	0.163260	0.152804
3	0.163524	0.124464	0.129016	0.151973	0.155709
4	0.175731	0.152956	0.131743	0.153010	0.128893
5	0.119783	0.126728	0.123473	0.108648	0.124700
6	0.151146	0.146742	0.150026	0.154598	0.131969
7	0.125452	0.151689	0.132426	0.166046	0.153307
8	0.164703	0.120787	0.094381	0.190640	0.150365
9	0.130878	0.151674	0.124516	0.149566	0.120247
10	0.155145	0.157450	0.095528	0.164757	0.152698
Mean	0.146452	0.137259	**0.122167**	0.156083	0.139392
Std	0.020204	0.015982	0.016721	0.020402	0.014705

TABLE 4. The error for dimension 3

	Dimension 4, Hidden nodes				
	6	7	8	9	10
1	0.120601	0.122211	0.127069	0.160776	0.121840
2	0.131547	0.124977	0.161715	0.124139	0.120200
3	0.154574	0.129369	0.123473	0.132766	0.135982
4	0.126811	0.125079	0.126356	0.125040	0.123974
5	0.121472	0.123937	0.117585	0.159125	0.123316
6	0.095290	0.136187	0.153872	0.162635	0.138452
7	0.154438	0.152314	0.168702	0.126563	0.140016
8	0.161447	0.125318	0.100115	0.165518	0.122031
9	0.143804	0.104974	0.157160	0.152270	0.169056
10	0.152052	0.119589	0.102829	0.128290	0.152027
Mean	0.136204	**0.126396**	0.133888	0.143712	0.134689
Std	0.020711	0.012065	0.024743	0.017698	0.016005

TABLE 5. The error for dimension 4

	Dimension 5, Hidden nodes				
	5	6	7	8	9
1	0.132331	0.132331	0.159188	0.135140	0.170846
2	0.120823	0.120823	0.119370	0.124525	0.126940
3	0.121063	0.121063	0.160163	0.122652	0.122106
4	0.122690	0.122690	0.123148	0.125123	0.161274
5	0.117412	0.117412	0.129360	0.167059	0.161859
6	0.152520	0.152520	0.102262	0.168188	0.193977
7	0.159476	0.159476	0.154130	0.164050	0.148780
8	0.149949	0.149949	0.147213	0.154635	0.103922
9	0.136422	0.136422	0.146666	0.169422	0.152576
10	0.195455	0.195455	0.156997	0.159797	0.144139
Mean	0.140814	0.140814	**0.139850**	0.149059	0.148642
Std	0.024293	0.024293	0.020018	0.019837	0.026020

TABLE 6. The error for dimension 5

From Table 4 to Table 6, the optimal numbers of hidden nodes that resulted in the lowest average NRMSE for dimensions 3, 4, and 5 were 10, 7, and 7, respectively. Therefore, we selected networks with 10 hidden nodes for dimension 3, and 7 hidden nodes for dimensions 4 and 5 for the prediction task. Each network is set to be trained for 100 times and stop when the error in the validation set is increased. The minimum NRMSE is recorded. Table 7 shows NRMSE of our initial 50 samples from each group in each dimension.

	Dim = 3, Hidden = 11		Dim = 4, Hidden = 9		Dim = 5, Hidden = 7	
	Group 1	Group 2	Group 1	Group 2	Group 1	Group 2
1	0.128114	0.153203	0.122787	0.15248	0.122101	0.103352
2	0.123587	0.134926	0.122033	0.148094	0.124285	0.15307
3	0.125109	0.162508	0.123396	0.154251	0.126454	0.156869
4	0.122656	0.151139	0.124164	0.128019	0.132101	0.142092
5	0.121223	0.152134	0.127045	0.137381	0.119477	0.130878
6	0.119839	0.122057	0.160541	0.149577	0.124361	0.094874
7	0.134074	0.124365	0.159086	0.151293	0.120607	0.15307
8	0.120442	0.119885	0.124336	0.192595	0.118094	0.130664
9	0.161918	0.152596	0.15496	0.127351	0.126617	0.162113
10	0.161575	0.145918	0.12391	0.168361	0.122995	0.164169
11	0.123671	0.138578	0.159751	0.119913	0.157416	0.137864
12	0.128224	0.153765	0.162159	0.163962	0.128504	0.154382
13	0.157832	0.141353	0.129839	0.16904	0.154494	0.152744
14	0.121263	0.193128	0.183224	0.103071	0.121312	0.168753
15	0.158886	0.142334	0.130289	0.130296	0.12295	0.12062
16	0.121707	0.098852	0.125233	0.151683	0.123392	0.15016
17	0.161817	0.152747	0.157905	0.126412	0.12362	0.137572
18	0.155691	0.167353	0.159093	0.154234	0.155496	0.096145
19	0.121641	0.15272	0.125694	0.133125	0.163718	0.168864
20	0.120102	0.09431	0.156585	0.152689	0.130536	0.172223
21	0.125526	0.150192	0.127547	0.152158	0.129325	0.127309
22	0.118626	0.120418	0.121885	0.17095	0.125694	0.157241
23	0.125306	0.152463	0.120535	0.154091	0.124113	0.139754
24	0.13388	0.1464	0.163863	0.146348	0.118295	0.148325
25	0.128134	0.131116	0.159781	0.168475	0.156445	0.153525
26	0.117478	0.141827	0.159155	0.15148	0.123285	0.140452
27	0.132596	0.19249	0.12294	0.195189	0.122351	0.15032
28	0.123724	0.103499	0.155092	0.151711	0.162428	0.16404
29	0.126506	0.153291	0.118197	0.16218	0.153218	0.153277
30	0.125641	0.126366	0.157004	0.121123	0.157686	0.193069
31	0.123929	0.160364	0.164213	0.15452	0.123342	0.157091
32	0.128764	0.152183	0.12093	0.131891	0.159376	0.137198
33	0.172749	0.098614	0.159912	0.095674	0.129699	0.150951
34	0.127014	0.126579	0.128798	0.137611	0.145019	0.14674
35	0.161198	0.165377	0.119594	0.094429	0.127321	0.188349
36	0.159744	0.144582	0.125355	0.168027	0.126055	0.14648
37	0.128516	0.148503	0.124163	0.152731	0.128036	0.168069
38	0.15572	0.154183	0.122542	0.10048	0.126162	0.165219
39	0.12493	0.15288	0.121851	0.128571	0.123813	0.144357
40	0.122896	0.099447	0.119146	0.156209	0.120492	0.154209
41	0.119968	0.152441	0.160524	0.16174	0.166884	0.154672
42	0.157391	0.129421	0.12342	0.159426	0.159517	0.157019
43	0.155012	0.163585	0.118773	0.156724	0.117398	0.155261
44	0.124067	0.162659	0.122902	0.151628	0.124158	0.193598
45	0.15544	0.164935	0.176201	0.168889	0.119128	0.151014
46	0.175848	0.09462	0.128601	0.15061	0.160742	0.103567
47	0.123861	0.16731	0.127799	0.15556	0.12379	0.154245
48	0.124446	0.162216	0.124121	0.153011	0.121246	0.157016
49	0.133942	0.143894	0.165913	0.147035	0.152445	0.163653
50	0.126269	0.146374	0.13221	0.14604	0.129625	0.102784
Mean	0.13496984	0.143242	0.13849994	0.14716676	0.13351236	0.14858564
Std	0.016954199	0.022689224	0.019141513	0.021295698	0.015675002	0.021862984

TABLE 7. NRMSE for two groups

We used an unpaired t-test to determine if there was a significant difference between the means of the two groups. The t-statistics for dimensions 3, 4, and 5 were -2.065143, -2.140244, and -3.962003, respectively, with p-values of 0.041549, 0.034817, and 0.000141. Since all the p-values are lower than 0.05, we reject the null hypothesis. We can conclude that time series with a higher Hurst exponent tend to be predicted more accurately.

5. Conclusion

In this study, we compare the effect of Hurst exponent in the prediction of SET index daily return by using ANN. We vary the number of the input nodes and the number of hidden nodes. We found that, in any daily return predictions, the accuracy is better in higher Hurst exponent periods than that in smaller Hurst exponent group. Therefore, the ANN model that is trained with high Hurst exponent data is more effective and give better trading results.

Authors' Contributions. All authors have read and approved the final version of the manuscript. The authors contributed equally to this work.

Conflicts of Interest. The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] H.E. Hurst, Long-Term Storage Capacity of Reservoirs, Trans. Amer. Soc. Civil Eng. 116 (1951), 770-799. https://doi.org/10.1061/TACEAT.0006518.

[2] B.B. Mandelbrot, J. Van Ness, Fractional Brownian Motions, Fractional Noises and Applications, SIAM Rev. 10 (1968), 422-437. https://www.jstor.org/stable/2027184.

[3] S.M. Mitra, Is Hurst Exponent Values Useful in Forecasting Financial Time Series?, Asian Soc. Sci. 8 (2012), 111-120. https://doi.org/10.5539/ass.v8n8p111.

[4] E.F. Fama, Random Walks in Stock Market Prices, Financ. Anal. J. 21 (1965), 55-59. https://www.jstor.org/stable/4469865.

[5] A.W. Lo, A.C. MacKinlay, Stock Market Prices Do Not Follow Random Walks: Evidence From a Simple Specification Test, Rev. Financ. Stud. 1 (1988), 41-66. https://www.jstor.org/stable/2962126.

[6] K. Sako, B.N. Mpinda, and P.C. Rodrigues, Neural Networks for Financial Time Series Forecasting, Entropy 24 (2022), 657. https://doi.org/10.3390/e24050657.

[7] A. Noorbakhsh, M. Shaygani, Deep Learning Approach for Stock Price Prediction: Comparing Single and Hybrid Models, Adv. Ind. Eng. 58 (2024), 237-249. https://doi.org/10.22059/aie.2024.372470.1889.

[8] P. Grassberger, T. Schreiber, C. Schafrath, Nonlinear Time Sequence Analysis, Int. J. Bifurc. Chaos 1 (1991), 521–547. https://doi.org/10.1142/s0218127491000403.

[9] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Comput. 9 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

[10] K. Hornik, M. Stinchcombe, H. White, Multilayer Feedforward Networks Are Universal Approximators, Neural Netw. 2 (1989), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

[11] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, in: Proceedings of the 32nd International Conference on Machine Learning, pp. 448–456, (2015). https://proceedings.mlr.press/v37/ioffe15.html.

[12] D. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, in: The 3rd International Conference for Learning Representations, San Diego, 2015. https://doi.org/10.48550/arXiv.1412.6980.

[13] Y.A. LeCun, L. Bottou, G.B. Orr, KR. Müller, Efficient BackProp, in: G. Montavon, G.B. Orr, K.R. Müller, (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg, 2012. https://doi.org/10.1007/978-3-642-35289-8_3.

[14] T. Lillicrap, D. Cownden, D. Tweed, C. Akerman, Random Synaptic Feedback Weights Support Error Backpropagation for Deep Learning, Nat. Commun 7 (2016), 13276. https://doi.org/10.1038/ncomms13276.

[15] B. Qian, K. Rasheed, Hurst Exponent and Financial Market Predictability, in: Proceedings of the Second IASTED Conference on Financial Engineering and Applications, pp. 203–209, (2004).

[16] M. Riedmiller, H. Braun, A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, in: IEEE International Conference on Neural Networks, IEEE, San Francisco. https://doi.org/10.1109/icnn.1993.298623.

[17] F. Takens, Detecting strange attractors in turbulence, in: D. Rand, L.S. Young, (eds) Dynamical Systems and Turbulence, Warwick 1980. Lecture Notes in Mathematics, vol 898. Springer, Berlin, Heidelberg, 1981. https://doi.org/10.1007/BFb0091924.

[18] A.S. Soofi, L. Cao, Modelling and Forecasting Financial Data: Techniques of Nonlinear Dynamics, Springer New York, 2002. https://doi.org/10.1007/978-1-4615-0931-8.

[19] A.A. Anis, E.H. Lloyd, The Expected Value of the Adjusted Rescaled Hurst Range of Independent Normal Summands, Biometrika. 63 (1976), 111-116. https://doi.org/10.2307/2335090.