



## PREDICTING STOCK MARKET INDICES USING CLASSIFICATION TOOLS



 Minjae Park<sup>1\*</sup>  
 Mi Lim Lee<sup>2</sup>  
 Jinpyo Lee<sup>3</sup>

<sup>1,2,3</sup> College of Business Administration, Hongik University, Seoul, Korea

<sup>1</sup>Email: [mjpark@hongik.ac.kr](mailto:mjpark@hongik.ac.kr) Tel: 8223201720

<sup>2</sup>Email: [milee@hongik.ac.kr](mailto:milee@hongik.ac.kr) Tel: 8223201755

<sup>3</sup>Email: [jinyo.lee@hongik.ac.kr](mailto:jinyo.lee@hongik.ac.kr) Tel: 8223201734



(+ Corresponding author)

### ABSTRACT

#### Article History

Received: 9 November 2018

Revised: 12 December 2018

Accepted: 21 January 2019

Published: 6 March 2019

#### Keywords

AdaBoost

Classifiers

Cost and profit

Data mining

Forecasting

Maintenance policy

Quality management

Standard & poor's 500.

#### JEL Classification:

C38; C53; F37; G17.

Increasing interest has been shown in the use of classifiers to extract informative patterns from time series data generated by monitoring financial phenomena. This paper investigates data mining and pattern recognition methods in forecasting the movement of the Standard & Poor's 500 index. We use functional forms of varying classifiers to predict financial time series data and to evaluate the performance of different classifiers. By using the time series ARIMA model, we forecast the Standard & Poor's 500 index. Additionally, with the AdaBoost algorithm and its extensions, we compare the classifying accuracy rates of bagging and boosting models with several classifiers, such as support vector machines,  $k$ -nearest neighbor, the probabilistic neural network, and the classification and regression tree. Results indicate that the boosting classifier with real AdaBoost (exponential loss) best forecast the Standard & Poor's 500 index movements. This result should be relevant to firms that want to predict the stock prices.

**Contribution/Originality:** The paper contributes to the understanding of how daily changes in the stock index and international stock indices can affect the S&P 500 index movement on a daily basis. Pattern recognition and classification methods of data mining are used to predict S&P 500 index movement.

## 1. INTRODUCTION

For several decades, both researchers and non-academics have been interested in forecasting stock prices. Stock prices, however, are not easy to predict accurately because they do not follow any model, process, or distribution. In this study, we attempt to predict stock prices using various methodologies by classifying the S&P 500 index for the last 20 years using several data mining tools. Surprisingly, research regarding predicting stock prices by classifying the stock index via data mining tools is scant. For constructing classifications, we take into account various index funds from several countries that influence the S&P 500. Next, we classify the S&P 500 depending on whether the selected indices increase or decrease the S&P 500 index compared with the previous day's closing index. We then use several data mining tools, including support vector machines (SVM),  $k$ -nearest neighbor ( $k$ NN), probabilistic neural network (PNN), the classification and regression tree (CART), and AdaBoost to evaluate the accuracy of the time series model's predictions.

Numerous studies have employed data mining tools, and review papers have summarized the effectiveness of classifiers, such as machine learning (Blum and Langley, 1997) neural networks (Warner and Misra, 1996; Zhang, 2000) and instance-based learning algorithms (Wilson and Martinez, 2000). The direction of change predictability has been evaluated via supervised learning techniques, machine learning algorithms, or classifier induction techniques (Apté and Hong, 1994; Kim, 2003). Many researchers (Aha *et al.*, 1991; Aha, 1992; Wilson and Martinez, 2000; Okamoto and Yugami, 2003) have investigated classification predictions using instance-based learning algorithms. Stock prediction has motivated many researchers to use classifiers (Chenoweth and Obradović, 1996; Hellstrom and Holmstrom, 1998; Rodriguez *et al.*, 2006; Qian and Rasheed, 2007). Some researchers have used neural networks for stock prediction (Chen *et al.*, 2003) and for financial forecasting (Malliaris, 1994; Leung *et al.*, 2000) whereas other researchers (Chenoweth and Obradović, 1996; Tsaih *et al.*, 1998; Qian and Rasheed, 2007) have used neural networks and a decision rule. Rodriguez *et al.* (2006) used the machine learning algorithm called AdaBoost for stock prediction, whereas bagging (Yang and Lee, 2004; Inoue and Kilian, 2008) has been used for financial forecasting and classifying stock patterns (Zhanggui *et al.*, 1999; Zeng *et al.*, 2001; Ting *et al.*, 2006). The combined approaches of more than two methods can be used for stock prediction neural network, kNN, naïve BC and genetic algorithm (Zemke, 1999) neural network, kNN, and decision tree (Qian and Rasheed, 2007). Zemke (1999) compared neural network, k-nearest neighbor, naïve Bayesian classifier, and genetic algorithm, developing classification rules for their prediction accuracies on stock exchange index data. The method yielding the best result, the nearest neighbor, is then refined and incorporated into a simple trading system that achieves returns above the index growth. Qian and Rasheed (2007) used the Hurst exponent to select a period with enhanced predictability to investigate the predictability of the Dow Jones Industrial Average index using artificial neural network, decision tree, and k-nearest neighbor. For the data mining tools, CART (Feldman and Gross, 2005; Razi and Athappilly, 2005; Moreira *et al.*, 2006) Bayes classifiers (Shin and Kil, 1998; Tsaih *et al.*, 1998; Pop, 2006) and prediction using SVM (Ince and Trafalis, 2004; Moreira *et al.*, 2006) have been used for financial market prediction. For example, Warner and Misra (1996) compared regression analysis and neural networks, Razi and Athappilly (2005) compared neural networks, and Ince and Trafalis (2004) compared CART, neural networks, and SVM. Additionally, the Hurst Exponent and financial market predictability have been investigated (Corazza and Malliaris, 2002). In this paper, we investigate stock prediction using various classifiers, predict the ups and downs of the S&P 500 index for 10 years, and identify the superior classifiers. Of the 10 years examined, the training data account for 9 years, and the testing data account for 1 year. A five-fold cross validation is also conducted for the study.

Several stock market data of eight countries were gathered from the Internet Time Series Data Library. Each stock index has a series of approximately 260 numbers per year, representing the price for the stock at the beginning of a trading day. The stock market data were derived from previously presented data (Franses and Van Dijk, 2000). Eight daily indices of stock markets were used, including Amsterdam (EOE), Frankfurt (DAX), Hong Kong (Hang Seng), London (FTSE100), New York (S&P 500), Paris (CAC40), Singapore (Singapore All Shares), and Tokyo (Nikkei) from January 1988 to December 1997. The data are divided into training data and testing data. Training data cover the period from January 1, 1988, to December 31, 1996, spread across 2,348 rows. Testing data cover the period from January 1, 1997, to December 31, 1997, spread across 261 rows. When binary classification for stock index data is performed, the classified variable is “1” if the stock index increased from the previous day; otherwise, it is “0.” Accordingly, the binary class data were obtained through the above process.

The structure of this paper is organized as follows: In Section 2, we show forecasts using the autoregressive integrated moving average (ARIMA) time series model. Section 3 presents the classification of data using learning methods. In Section 4, we explain the method of resampling using bootstrapping. Section 5 provides concluding remarks.

## 2. PROBLEM DESCRIPTION

Mining of financial data has proven to be extremely efficient and profitable. Increased interest has therefore been shown in using data mining techniques to extract patterns from time series data generated by monitoring various financial phenomena. Among the various types of time series data, the stock indices of several exchange markets were selected, and the ARIMA model was used to forecast the stock price. The ARIMA model and the integrated moving average (IMA) model are examples of nonstationary models. Here we consider the model  $\varphi(B)\tilde{z}_t = \theta(B)a_t$  where  $\varphi(B)$  is a nonstationary autoregressive operator such that  $d$  of the roots of  $\varphi(B) = 0$  are unity, and the remainder lies outside of the unit circle. We can then express  $\varphi(B)\tilde{z}_t = \theta(B)a_t$  in the form of  $\phi(B)\tilde{z}_t = \phi(B)(1-B)^d \tilde{z}_t = \theta(B)a_t$ , where  $\phi(B)$  is a stationary autoregressive operator. Because  $\nabla^d \tilde{z}_t = \nabla^d z_t$ , for  $d \geq 1$ , where  $\nabla = 1-B$  is the difference operator, we can write the model as  $\phi(B)\nabla^d z_t = \theta(B)a_t$ . Equivalently, the process is defined by the two equations:  $\phi(B)w_t = \theta(B)a_t$  and  $w_t = \nabla^d z_t$ , where  $\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \dots - \theta_q B^q$ ,

$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \dots - \phi_p B^p$ . This is known as the autoregressive integrated moving average (ARIMA) process

$(p, d, q)$ . IMA models form a special case of the ARIMA models and are similar to the model given by

$\phi(B)\nabla^d z_t = \theta(B)a_t$ . They can be obtained by dropping the autoregressive part of an ARIMA  $(p, d, q)$  and are given by  $\nabla^d z_t = \theta(B)a_t$ . This class of models are known as integrated moving average (IMA) processes.

Nevertheless, an underlying assumption in the ARIMA model is that the series contains several patterns. Hence, cluster data such as temperature, population trends, and so on can be predicted. The stock market though is inherently unpredictable. Using several classifiers by R program and Matlab, we carefully investigate the stock index. Then, using several classifying methodologies, we train and test the several indices. We then compare the classifiers in terms of accuracy rate by changing several functions and parameters. We use support vector machines, boosting and bagging, probabilistic neural network (PNN),  $k$ -nearest neighbor (KNN), and the classification and regression tree (CART). Both boosting and bagging have been successfully used in machine learning to improve the performance of classification algorithms. To investigate the predictability of daily financial time series directional movement, we use the data mining technique. For boosting, we use the AdaBoost algorithm, and CART is chosen as a weak classifier. In addition, the use of CART as an individual classifier demonstrates that boosting and bagging are better than CART is in terms of accuracy rate. In summary, methodologies are designed to evaluate the performance of the techniques and models. First, using the time series ARIMA model, S&P 500 is forecast and compared with the true values. Second, a comparison is performed of the accuracy rates of the directional success of bagging and boosting models to the SVM,  $k$ NN, and, especially, CART.

This process is executed 10 times for each dataset to arrive at a prediction of the accuracy rates' maximum, minimum, mean, standard deviation, and coefficient of variation. These stock predictions do not have an independent variable. The absence of an independent variable constitutes a substantial difference from how we conduct our work.

## 3. CLASSIFYING USING LEARNING METHODS

Several classifiers are used to predict the S&P 500. We conducted time series analyses and predicted S&P 500 indices. We therefore used several classifiers, performed binary classification for the changed S&P 500 index, and compared their accuracy rates.

### 3.1. Support Vector Machine (SVM)

To train and test the data, we use the Support Vector Machine (SVM) function in the R package “kernlab” (Karatzoglou *et al.*, 2004). In the R package “kernlab,” we use the code “ksvm,” which represents “kernel support vector machines.” SVMs are excellent tools for classification. We only use four options for kernel function: radial basis function (RBF), polynomial, linear, and sigmoid. For the final project, we add three further optional functions: Laplacian kernel, Bessel kernel, and analysis of variance (ANOVA) RFB kernel. Each equation for the kernel functions is presented below:

$$\text{The Gaussian RBF } k(x, x') = \exp(-\sigma \|x - x'\|^2)$$

$$\text{The Polynomial } k(x, x') = (\text{scale} \langle x, x' \rangle + \text{offset})^{\text{degree}}$$

$$\text{The Linear } k(x, x') = \langle x, x' \rangle$$

$$\text{The Sigmoid } k(x, x') = \tanh(\text{scale} \langle x, x' \rangle + \text{offset})$$

$$\text{The Laplacian } k(x, x') = \exp(-\sigma \|x - x'\|)$$

$$\text{The Bessel } k(x, x') = \exp(-\text{Bessel}_{(v+1)}^n \sigma \|x - x'\|^2)$$

$$\text{The ANOVA RBF } k(x, x') = \sum_{1 \leq i_1, \dots, i_d \leq N} \prod_{d=1}^D k(x_{i_d}, x'_{i_d})$$

Table 1 presents the accuracy rates for different functions using various parameters. In terms of accuracy rate, polynomial, linear, and sigmoid functions outperform others.

Table-1. Accuracy rates using SVM with various parameters.

Support Vector Machines						
RBF	Parameters	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$\sigma = 4$	$\sigma = 5$
	Accuracy Rate	0.51341	0.52107	0.52490	0.52874	0.52874
Polynomial	Parameters	Degree = 1	Degree = 2	Degree = 3		
	Accuracy Rate	0.54023	0.42912	0.52107		
Linear	Parameters	No parameter is needed for Linear Function				
	Accuracy Rate	0.54023				
Sigmoid	Parameters	(1,1)	(1,2)	(2,1)	(2,2)	(2,3)
	Accuracy Rate	0.54023	0.54023	0.54023	0.54023	0.54023
Laplacian	Parameters	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$\sigma = 4$	$\sigma = 5$
	Accuracy Rate	0.517241	0.51724	0.51724	0.52107	0.53640
Bessel	Parameters	(1,1,1)	(1,1,2)	(1,1,3)	(1,1,4)	(1,1,5)
	Accuracy Rate	0.501916	0.45977	0.51724	0.51724	0.50958
ANOVA RBF	Parameters	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
	Accuracy Rate	0.45977	0.45977	0.45594	0.45977	0.45977

Source: Authors calculation from R software and Matlab software.

### 3.2. k-Nearest Neighbor Classifier

We used Matlab to find the *k*-nearest neighbor. According to the results of the analysis, no significant differences were shown depending on *k*. The percentage of classification error is the lowest (below 50%) when the *k*-nearest neighbor classifier is 9. The confusion matrix is given by  $\begin{pmatrix} 81 & 60 \\ 70 & 50 \end{pmatrix}$ , and the accuracy rate is 0.5019.

### 3.3. Probabilistic Neural Network (PNN)

In this subsection, we used the probabilistic neural network (PNN) as a classifier. We changed the  $\sigma$  parameters from 0.010 to 0.020. The accuracy rate is the highest when  $\sigma$  is 0.018. As  $\sigma$  increases from 0.010, the accuracy rate continually increases until  $\sigma$  reaches 0.018; the accuracy rate then declines as  $\sigma$  increases. The sensitivity and reliability also peak when  $\sigma$  is 0.018. No clear pattern in specificity is shown.

Table-2. Accuracy rates using PNN with different values of  $\sigma$ .

$\sigma$	Confusion Matrix		(a)Accuracy	(b)Sensitivity	(c)Reliability	(d)Specificity
0.010	39	102	0.452107	0.276596	0.4875	0.658333
	41	79				
0.011	52	89	0.471264	0.368794	0.514851	0.591667
	49	71				
0.012	59	82	0.478927	0.41844	0.522124	0.55
	54	66				
0.013	69	72	0.482759	0.489362	0.522727	0.475
	63	57				
0.014	80	61	0.509579	0.567376	0.544218	0.441667
	67	53				
0.015	85	56	0.51341	0.602837	0.544872	0.408333
	71	49				
0.016	86	55	0.524904	0.609929	0.554839	0.425
	69	51				
0.017	86	55	0.532567	0.609929	0.562092	0.441667
	67	53				
<b>0.018</b>	87	54	<b>0.536398</b>	0.617021	0.564935	0.441667
	67	53				
0.019	85	56	0.517241	0.602837	0.548387	0.416667
	70	50				
0.020	90	51	0.505747	0.638298	0.535714	0.35
	78	42				

Source: Authors calculation from R software and Matlab software.

Using sensitivity and specificity, we obtain the receiver operating characteristic (ROC) curve for the probabilistic NN function shown in Figure 1.

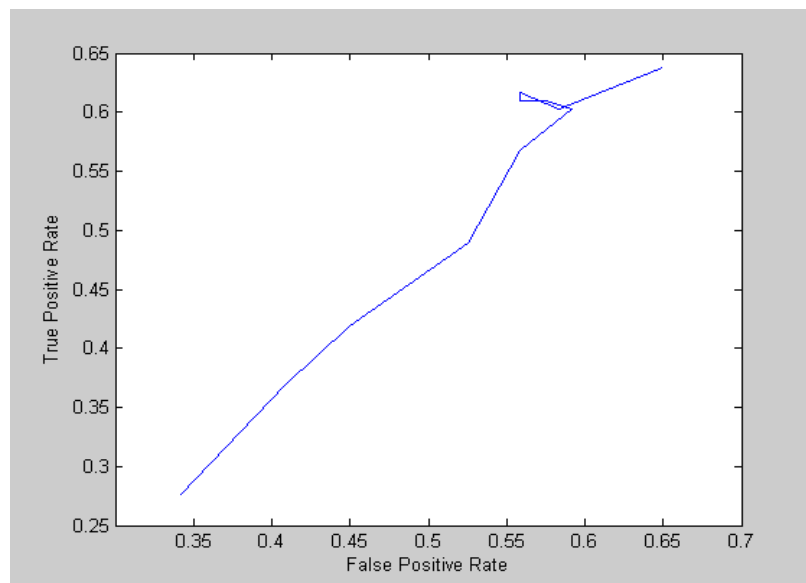


Figure-1. ROC curve for probabilistic NN function.

Source: Authors' calculation from R software and Matlab software.

### 3.4. Classification and Regression Tree (CART)

Using the function “rpart,” we built a large regression tree and investigated the optimal magnitude for a range of complexity parameters. The results are presented in Table 3.

Table-3. Classification and regression tree.

	CP	nsplit	rel.error	x.error	xstd.
1	0.03794	0	1.0000	1.0000	0.0656
2	0.03252	4	0.8293	1.2358	0.0648
3	0.02439	5	0.7968	1.2276	0.0649
4	0.01626	11	0.6504	1.1951	0.0651
5	0.01220	15	0.5854	1.1951	0.0651
6	0.01000	17	0.5610	1.1951	0.0651

Source: Authors' calculation from R software and Matlab software.

We observe that the error is minimized when the complexity parameter (CP) is approximately 0.03794. When the CP is 0.03794, the minimized error is 1.00. By using another plot function, we can easily compare the complexity parameters shown in Figure 2.

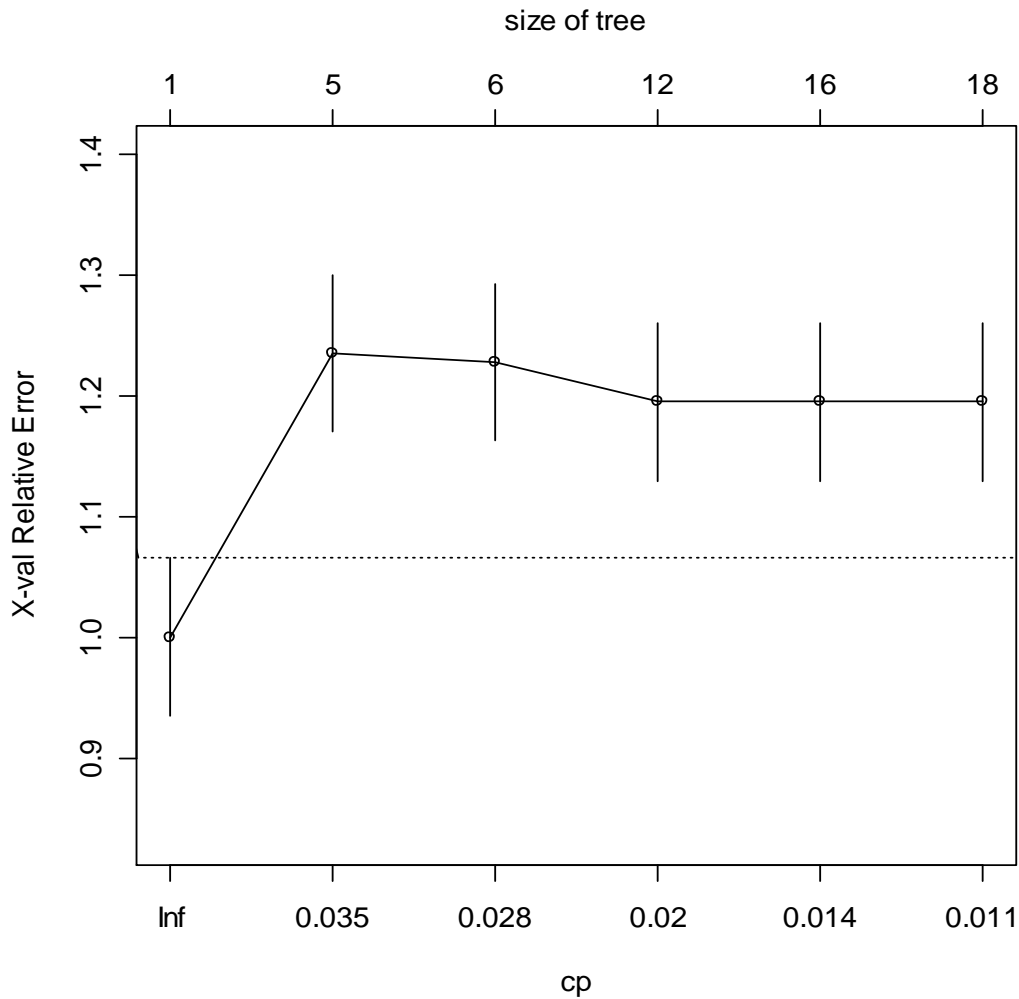


Figure-2. Plot of complexity parameter.

Source: Authors' calculation from R software and Matlab software.

We then selected the best CP and drew the decision tree classifiers shown in Figure 3. Finally, we could predict the classified variables of the test data with the training data using CART.

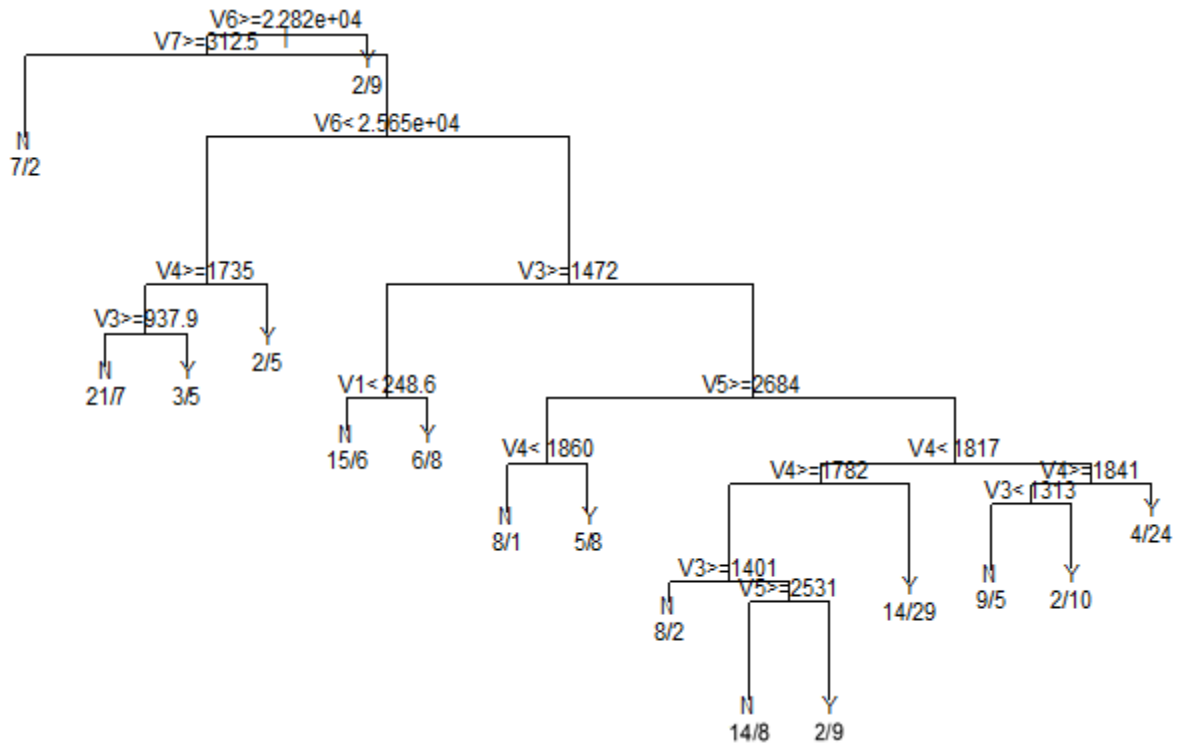


Figure-3. Decision Tree.

Source: Authors' calculation from R software and Matlab software.

### 3.5. Boosting with CART

Boosting has proven to be an effective method of improving the performance of weak classifiers, both theoretically and empirically. The underlying concept of boosting is the combination of weak classifiers to form an ensemble, the performance of which is significantly improved. In our study, we chose the weak classifiers as CART. In addition, adaptive boosting (AdaBoost) proves to be a practical implementation of the boosting ensemble method. Freund and Schapire (1995) developed the well-known AdaBoost.M1 algorithm, which generates a sequentially weighted set of weak classifiers that are combined to form an overall strong classifier. Boosting is a technique that improves the performance of machine learning algorithms. The boosting technique has been successfully used in machine learning to improve the performance of classification algorithms. The purpose of boosting is to increase the strength of a weak learning algorithm. Generally, a weak learning algorithm should be superior to random guessing. For a binary classifier, the weak learning hypothesis achieves 50% accuracy. Boosting trains a weak learner a number of times, using a reweighted version of the original training set. Boosting trains the first weak learner with equal weight on all the data points in the training set and then trains all the other weak learners based on the updated weight. The data points incorrectly classified by the previous weak learner are given a heavier weight, and the correctly classified data points are given a lighter weight.

Accordingly, the next classifier will attempt to fix the errors made by the previous learner. Several boosting algorithms are available, including AdaBoost, AdaBoost.M1, AdaBoost.M2, and AdaBoost.R. AdaBoost is used for binary classification problems, and AdaBoost.R is used for regression. In each step of the sequence, Adaboost attempts to find an optimal classifier according to the current distribution of weights on the observations. If an observation is incorrectly classified using the current distribution of weights, the observations under the current distribution of weights will then receive less weight in the next iteration. In Friedman et al. (2000) three extensions of the AdaBoost algorithm were proposed: Gentle AdaBoost, Logit AdaBoost, and Real AdaBoost. Among them, we use Gentle and Real AdaBoost.

3.6. Boosting Extensions

The Real AdaBoost algorithm attempts to minimize the expectation of  $e^{(-yF(x))}$  similarly to Discrete AdaBoost. In general, Friedman *et al.* (2000) proposed a further extension called the Gentle AdaBoost algorithm, which minimizes the exponential loss function of AdaBoost through a sequence of Newton steps. For Real AdaBoost and Gentle Adaboost, the weak classifier is CART. This is because we determined the accuracy rate of CART as an individual classifier and because the most popular weak learners employed by boosting algorithms are classification and regression trees. The objective is to construct a classification rule that can accurately predict the class labels of objects for which only the attribute vector is observed.

Discrete AdaBoost

The discrete AdaBoost implements the original AdaBoost using both exponential and logistic loss functions for classification problems. When the exponential method is used, the confusion matrix is given by  $\begin{pmatrix} 545 & 569 \\ 306 & 928 \end{pmatrix}$ , and the train error rate is 0.373. After training and testing the data, we obtain the training and testing error plot shown in Figure 4. Additionally, the pairs function produces a visualization of the pairwise relationships between a subset of variables in the dataset shown in Figure 5. The upper panels represent the true class labels as a color for each pairwise relationship, whereas the lower panels give the predicted class for each observation.

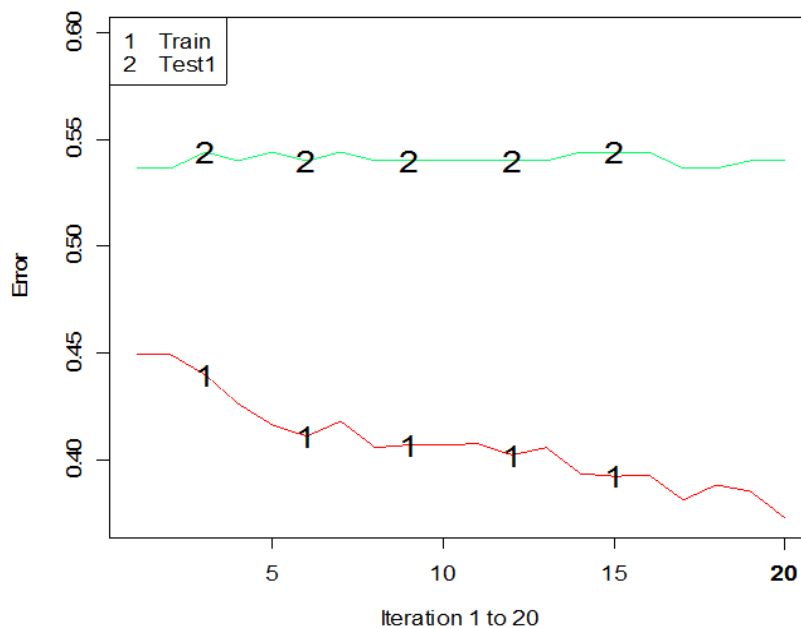


Figure-4. Training and Testing Error.  
Source: Authors' calculation from R software and Matlab software.



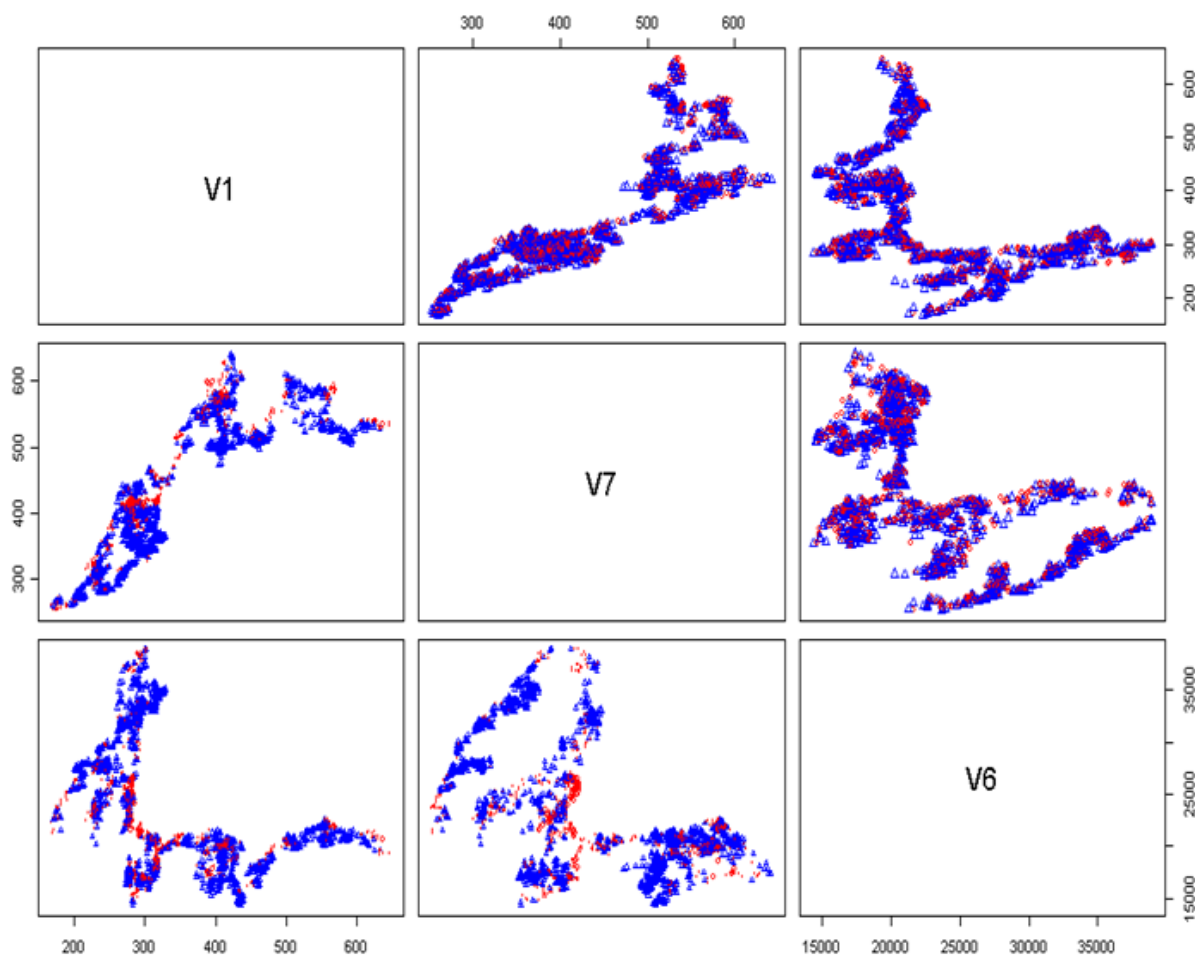


Figure-5. Pairs plot of the first three descriptors.

Source: Authors' calculation from R software and Matlab software.

*Real AdaBoost*

Real AdaBoost is the generalization of a discrete AdaBoost algorithm first introduced by Freund and Schapire (1995). Real AdaBoost should be treated as a basic hardcore-boosting algorithm. When the exponential method is used, the confusion matrix is given by  $\begin{pmatrix} 8 & 1106 \\ 0 & 1234 \end{pmatrix}$ , and the train error rate is 0.471.

*Gentle AdaBoost*

As a more robust and stable version of Real AdaBoost, Gentle AdaBoost has proven to be the most efficient boosting algorithm in practice. However, our experiments in the current project show that Real AdaBoost performs slightly better than Gentle AdaBoost on regular data does, although Gentle AdaBoost functions considerably better on noisy data and is significantly more resistant to outliers. For the training results, the accuracy rate is 0.802, and its kappa value is given by 0.601. From the test results, the accuracy rate is 0.452, and the kappa value is -0.059. When the exponential method is used, the confusion matrix is given by  $\begin{pmatrix} 834 & 280 \\ 255 & 979 \end{pmatrix}$ , and the train error rate is 0.228. Gentle AdaBoost is used to illustrate the variable importance function. Figure 6 shows the scores of the variable assessment for Gentle AdaBoost. This plot is helpful in identifying variables that are important for predicting the S&P 500.

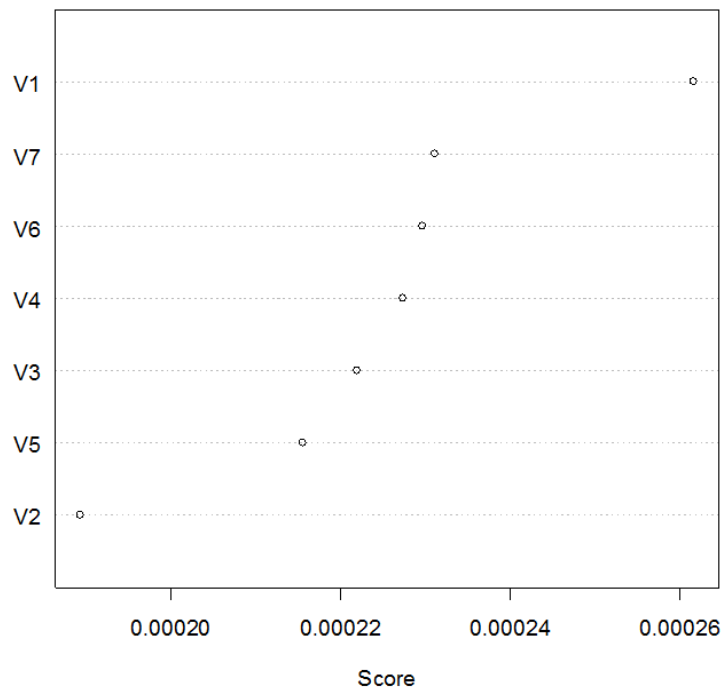


Figure-6. Variable Importance plot.

Source: Authors' calculation from R software and Matlab software.

Table-4. Variable Scores.

1	7	6	4	3	5	2
0.00026168	0.00023111	0.00022971	0.00022744	0.00022201	0.00021562	0.0001893

Source: Authors' calculation from R software and Matlab software.

The variable scores can be obtained directly without a plot, as shown in Table 4. Table 5 shows the accuracy rates of three AdaBoosts, where  $M$  indicates the number of iterations. As expected, the accuracy rate is generally better when the number of iterations is increased. The boosting algorithm in its general form can operate under an arbitrary loss function. In this study, we use both the exponential and logistic loss function. Table 5 summarizes the accuracy rates using AdaBoost algorithms.

Table-5. Accuracy Rates Using AdaBoost Algorithms.

	Exponential Loss				Logistic Loss			
	M = 1	M = 10	M = 20	M = 50	M = 1	M = 10	M = 20	M = 50
Discrete Adaboost	0.5520	0.5928	0.6286	0.6670	0.5324	0.5779	0.5788	0.5988
Real Adaboost	0.5451	0.6014	0.6537	0.6840	0.5511	0.5916	0.5894	0.6120
Gentle Adaboost	0.4744	0.4744	0.5443	0.5311	0.5622	0.4744	0.4744	0.4744

Source: Authors' calculation from R software and Matlab software.

### 3.7. Bagging

Bagging involves fitting the model, including all potential data points, to the original training set. Bootstrap samples with replacement of the original training set of a size up to that of the training set are generated. Some of the data points can appear more than once, whereas others do not appear. By averaging the resamples, bagging effectively removes the instability of the decision rule. Thus, the variance of the bagged prediction model is smaller than when only one classifier is fit into the original training set. By using bagging, overfitting can also be avoided. Nonetheless, our classified experiment using bagging shows an accuracy rate of 48.66%. When the exponential method is used, the confusion matrix is given by  $\begin{pmatrix} 29 & 43 \\ 91 & 98 \end{pmatrix}$ , and the train error rate is 0.51341. By feeding the same inputs for both the boosting and bagging procedures, as well as for SVM, kNN, PNN, and CART, we can compare

how each performed. The accuracy percentage is used to measure the performance. If the model predicts an increase and the index increases or remains steady, then it is correct; otherwise, if the index decreases, it is assumed to be incorrect. The accuracy of the percentage of stock and the index over the nine years are summarized in Table 6.

Table-6. Accuracy rate using different classifiers.

Compare the classifiers' Accuracy Rate			
Boosting	Exponential Loss	Discrete AdaBoost	0.6657
		Gentle AdaBoost	0.5443
		Real AdaBoost	0.6840
	Logistic Loss	Discrete AdaBoost	0.5988
		Gentle AdaBoost	0.5622
		Real AdaBoost	0.6120
Bagging			0.4866
Support Vector Machines		RBF	0.5287
		Polynomial	0.5402
		Linear	0.5402
		Sigmoid	0.5402
		Laplacian	0.5364
		Bessel	0.5172
		ANOVA RBF	0.4597
<i>k</i> -Nearest Neighbor			0.5019
Probabilistic Neural Network			0.5364
CART			0.5231

Source: Authors' calculation from R software and Matlab software.

Table 6 shows that AdaBoost's accuracy rates are superior to those of any other individual classifiers. This is because AdaBoost uses a sequence of simple weighted classifiers, each forced to learn a different aspect of the data, to generate a final, comprehensive classifier, which with high probability outperforms any individual classifier in terms of misclassification error rate. Real AdaBoost with exponential loss provides the best accuracy rate.

#### 4. DISCUSSION

This paper investigates a short-term prediction of stock prices, whereby we only attempt to provide a daily indication of whether the S&P 500 index would increase or decrease. Although a considerable amount of extant research has covered stock prediction, few studies have used dependent variables. We need to be careful when using dependent variables. Because both independent and dependent variables usually constitute daily data, if we were to predict the S&P 500 using independent variables, we would also need to establish an interval. For example, the independent variable data of one week would affect the S&P 500 index of the subsequent week. Otherwise, we would need to determine a narrower purpose for this paper, i.e., to compare the classifiers or to predict the stock price.

#### 5. CONCLUDING REMARKS

We classified the S&P 500 index using the indices used in several other countries and predicted the S&P 500 index using the ARIMA time series model. The classifiers are the SVM, *k*-nearest neighbor classifiers, PNN, bagging and boosting, several extensions of AdaBoost, and CART. We present the selected accuracy rates using the financial data shown in Table 7.

Table-7. Accuracy rates comparison using different classifiers.

Comparing the classifiers' accuracy rates			
Boosting	Exponential Loss	Real AdaBoost	0.6840
	Logistic Loss	Real AdaBoost	0.6120
Bagging			0.4866
Support Vector Machines		Polynomial	0.5402
k-Nearest Neighbor			0.5019
Probabilistic Neural Network			0.5364
CART			0.5231

Source: Authors' calculation from R software and Matlab software.

Table 7 shows that AdaBoost uses weak classifiers and that CART provides the highest rate of accuracy. We also practice resampling to allocate our investments efficiently using bootstrap. Future research topics could be related to the content explored in this study. Besides the stock indices of other countries, several other economic factors and indices can affect the S&P 500 index, such as oil prices, interest rates, exchange rates, and so on. Using these factors, we can predict the S&P 500 more accurately. However, it is not easy to obtain these time series accumulated data in the same format. In addition, financial data mining involves numerous financial issues, such as stock options, interest rate derivatives, and credit derivatives, whereas each index contains many individual stock prices. Hence, we can use clustering, principal component analysis, and Fisher discriminant functions to make data more controllable. Consequently, data mining in financial engineering topics will continue to present significant potential for growth.

**Funding:** This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2014S1A5A8012594).

**Competing Interests:** The authors declare that they have no competing interests.

**Contributors/Acknowledgement:** All authors contributed equally to the conception and design of the study.

## REFERENCES

- Aha, D.W., 1992. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36(2): 267-287. Available at: [https://doi.org/10.1016/0020-7373\(92\)90018-g](https://doi.org/10.1016/0020-7373(92)90018-g).
- Aha, D.W., D. Kibler and M.K. Albert, 1991. Instance-based learning algorithms. *Machine Learning*, 6(1): 37-66. Available at: <https://doi.org/10.1007/bf00153759>.
- Apté, C. and S.J. Hong, 1994. Predicting equity returns from securities data with minimal rule generation. *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*. AAAI Press. pp: 407-418.
- Blum, A.L. and P. Langley, 1997. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2): 245-271. Available at: [https://doi.org/10.1016/s0004-3702\(97\)00063-5](https://doi.org/10.1016/s0004-3702(97)00063-5).
- Chen, A.-S., M.T. Leung and H. Daouk, 2003. Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan stock index. *Computers & Operations Research*, 30(6): 901-923. Available at: [https://doi.org/10.1016/s0305-0548\(02\)00037-0](https://doi.org/10.1016/s0305-0548(02)00037-0).
- Chenoweth, T. and Z. Obradović, 1996. A multi-component nonlinear prediction system for the S&P 500 index. *Neurocomputing*, 10(3): 275-290. Available at: [https://doi.org/10.1016/0925-2312\(95\)00109-3](https://doi.org/10.1016/0925-2312(95)00109-3).
- Corazza, M. and A. Malliaris, 2002. Multi-fractality in foreign currency markets. *Multinational Finance Journal*, 6(2): 65-98. Available at: <https://doi.org/10.17578/6-2-1>.
- Feldman, D. and S. Gross, 2005. Mortgage default: Classification trees analysis. *The Journal of Real Estate Finance and Economics*, 30(4): 369-396. Available at: <https://doi.org/10.1007/s11146-005-7013-7>.
- Franses, P.H. and D. Van Dijk, 2000. *Non-linear time series models in empirical finance*. New York, U.S.A: Cambridge University Press. pp: 9-10.

- Freund, Y. and R.E. Schapire, 1995. A decision-theoretic generalization of on-line learning and an application to boosting. European Conference on Computational Learning Theory. Springer, Berlin, Heidelberg. pp: 23-37.
- Friedman, J., T. Hastie and R. Tibshirani, 2000. Additive logistic regression: A statistical view of boosting (With Discussion and a Rejoinder by the Authors). The Annals of Statistics, 28(2): 337-407. Available at: <https://doi.org/10.1214/aos/1016120463>.
- Hellstrom, T. and K. Holmstrom, 1998. Predicting the stock market. Research and Reports Opuscula ISRN HEV-BIB-OP 26-SE, Malardalen University, Vasteras, Sweden.
- Ince, H. and T. Trafalis, 2004. Kernel principal component analysis and support vector machines for stock price prediction. International Joint Conference on Neural Networks.
- Inoue, A. and L. Kilian, 2008. How useful is bagging in forecasting economic time series? A case study of US consumer price inflation. Journal of the American Statistical Association, 103(482): 511-522. Available at: <https://doi.org/10.1198/016214507000000473>.
- Karatzoglou, A., A. Smola, K. Hornik and A. Zeileis, 2004. Kernlab—an S4 package for kernel methods in R. Journal of Statistical Software, 11(9): 1-20. Available at: <https://doi.org/10.18637/jss.v011.i09>.
- Kim, K.-j., 2003. Financial time series forecasting using support vector machines. Neurocomputing, 55(1-2): 307-319. Available at: [https://doi.org/10.1016/s0925-2312\(03\)00372-2](https://doi.org/10.1016/s0925-2312(03)00372-2).
- Leung, M.T., H. Daouk and A.-S. Chen, 2000. Forecasting stock indices: A comparison of classification and level estimation models. International Journal of Forecasting, 16(2): 173-190. Available at: [https://doi.org/10.1016/s0169-2070\(99\)00048-5](https://doi.org/10.1016/s0169-2070(99)00048-5).
- Malliaris, M.E., 1994. Modeling the behavior of the S&P 500 index: A neural network approach. In Artificial Intelligence for Applications. Proceedings of the Tenth Conference on IEEE. pp: 86-90.
- Moreira, J.M., A.M. Jorge, C. Soares and D.J.F. Sousa, 2006. Improving SVM-Linear predictions using CART for example selection. Proceedings of the 16th International Conference on Foundations of Intelligent Systems Springer-Verlag. pp: 632-641.
- Okamoto, S. and N. Yugami, 2003. Effects of domain characteristics on instance-based learning algorithms. Theoretical Computer Science, 298(1): 207-233. Available at: [https://doi.org/10.1016/s0304-3975\(02\)00424-3](https://doi.org/10.1016/s0304-3975(02)00424-3).
- Pop, I., 2006. An approach of the Naive Bayes classifier for the document classification. General Mathematics, 14(4): 135-138.
- Qian, B. and K. Rasheed, 2007. Stock market prediction with multiple classifiers. Applied Intelligence, 26(1): 25-33. Available at: <https://doi.org/10.1007/s10489-006-0001-7>.
- Razi, M.A. and K. Athappilly, 2005. A comparative predictive analysis of neural networks (NNs), nonlinear regression and classification and regression tree (CART) models. Expert Systems with Applications, 29(1): 65-74. Available at: <https://doi.org/10.1016/j.eswa.2005.01.006>.
- Rodriguez, P., S. Rivero and J. Juan, 2006. Using machine learning algorithms to find patterns in stock prices, FEDEA Working Paper No. 2006-12. Available from <https://ssrn.com/abstract=893141> or <http://dx.doi.org/10.2139/ssrn.893141>.
- Shin, F.B. and D.H. Kil, 1998. Classification Cramer–Rao bounds on stock price prediction. Journal of Forecasting, 17(5-6): 389-399. Available at: [https://doi.org/10.1002/\(sici\)1099-131x\(199809\)17:5/6<389::aid-for703>3.0.co;2-n](https://doi.org/10.1002/(sici)1099-131x(199809)17:5/6<389::aid-for703>3.0.co;2-n).
- Ting, J., T.-c. Fu and F.-l. Chung, 2006. Mining of stock data: Intra-and inter-stock pattern associative classification. Threshold, 5(100): 5-99.
- Tsaih, R., Y. Hsu and C.C. Lai, 1998. Forecasting S&P 500 stock index futures with a hybrid AI system. Decision Support Systems, 23(2): 161-174. Available at: [https://doi.org/10.1016/s0167-9236\(98\)00028-1](https://doi.org/10.1016/s0167-9236(98)00028-1).
- Warner, B. and M. Misra, 1996. Understanding neural networks as statistical tools. The American Statistician, 50(4): 284-293. Available at: <https://doi.org/10.1080/00031305.1996.10473554>.
- Wilson, D.R. and T.R. Martinez, 2000. Reduction techniques for instance-based learning algorithms. Machine Learning, 38(3): 257-286.

- Yang, Y. and T.H. Lee, 2004. Bagging binary predictors for time series. In Econometric Society 2004 Far Eastern Meetings (No. 512). Econometric Society.
- Zemke, S., 1999. Nonlinear index prediction. *Physica A: Statistical Mechanics and its Applications*, 269(1): 177-183. Available at: [https://doi.org/10.1016/s0378-4371\(99\)00091-6](https://doi.org/10.1016/s0378-4371(99)00091-6).
- Zeng, Z., H. Yan and A.M. Fu, 2001. Time-series prediction based on pattern classification. *Artificial Intelligence in Engineering*, 15(1): 61-69. Available at: [https://doi.org/10.1016/s0954-1810\(00\)00026-1](https://doi.org/10.1016/s0954-1810(00)00026-1).
- Zhang, G.P., 2000. Neural networks for classification: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4): 451-462.
- Zhanggui, Z., H. Yan and A.M. Fu, 1999. New stock price prediction method based on pattern classification. *International Joint Conference on Neural Networks (IJCNN'99)*.

*Views and opinions expressed in this article are the views and opinions of the author(s). Asian Economic and Financial Review shall not be responsible or answerable for any loss, damage or liability etc. caused in relation to/arising out of the use of the content.*