Expert Systems With Applications 186 (2021) 115716

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process

Kyung Keun Yun [a], Sang Won Yoon [a,b], Daehan Won [a,*]

[a] Department of Systems Science and Industrial Engineering, State University of New York at Binghamton, Binghamton, NY 13902, United States
[b] Faculty of Commerce, Waseda University, Tokyo 169-8050, Japan

ARTICLE INFO

Keywords:
Genetic algorithm
XGBoost feature selection
Technical indicators
Blessing of dimensionality
Curse of dimensionality
Feature set expansion
Optimal feature set

ABSTRACT

The stock market has performed one of the most important functions in a laissez-faire economic system by gathering people, companies, and flows of money for several centuries. There have been numerous studies on the stock market among researchers to predict stock prices, and a growing number of studies employed machine learning or deep learning techniques on the stock market predictions with the advent of big data and the rapid development of artificial intelligence techniques. However, making accurate predictions of stock price direction remains difficult because stock prices are inherently complex, nonlinear, nonstationary, and sometimes too irrational to be predictable. Despite the wealth of information, previous prediction systems often overlooked key indicators and the importance of feature engineering. This study proposes a hybrid GA-XGBoost prediction system with an enhanced feature engineering process consisting of feature set expansion, data preparation, and optimal feature set selection using the hybrid GA-XGBoost algorithm. This study experimentally verifies the importance of feature engineering process in stock price direction prediction by comparing obtained feature sets to original dataset as well as improving prediction performance to outperform benchmark models. Specifically, the most significant accuracy increment comes from feature expansion that adds 67 technical indicators to the original historical stock price data. This study also produces a parsimonious optimal feature set using the GA-XGBoost algorithm that can achieve the desired performance with substantially fewer features. Consequently, this study empirically proves that a successful prediction performance largely depends on a deliberate combination of feature engineering processes with a baseline learning model to make a good balance and harmony between the curse of dimensionality and the blessing of dimensionality.

1. Introduction

As one of the most important financial marketplaces in a laissez-faire economic system, the stock market joins people and companies with flows of money. Because of its easy accessibility and high profitability, the stock market has always attracted investors seeking profitable returns and companies needing capital to fund their businesses. The stock market has experienced dramatic expansion and growth with the emergence of personal computers and the rapid proliferation of electronic trading platforms since the late 20th century. Electronic trading systems and the development of computing technology made stock trading more efficient and produced a massive amount of stock trading data and information. Consequently, big data of stock prices have become a hot topic for data scientists and researchers who attempt to solve the secret mechanism of stock price movement.

However, it is difficult to build a function or system that can predict the movement of stock prices with a great degree of accuracy, because stock prices are volatile, complex, nonlinear, noisy, and sometimes too irrational to be predictable (Chen & Hao, 2017; Chung & Shin, 2018; Wang et al., 2018; Basak et al., 2019; Henrique et al., 2019). Stock prices are affected in convoluted manners by various external factors such as political, economic, or social news and even public sentiment. Nevertheless, there have been numerous studies on the behavior of forecasting stock prices among researchers from multiple academic fields as well as investors and traders (Chen & Hao, 2017; Chung & Shin, 2018; Yu & Yan, 2020).

There are two conventional approaches to predicting stock prices and the movement (Shah et al., 2019): fundamental analysis and

* Corresponding author at: Department of Systems Science and Industrial Engineering, Binghamton University, 4400 Vestal Parkway E, Binghamton, NY 13902, United States.
E-mail addresses: kyun4@binghamton.edu (K.K. Yun), yoons@binghamton.edu (S.W. Yoon), dhwon@binghamton.edu (D. Won).

https://doi.org/10.1016/j.eswa.2021.115716
Received 9 March 2021; Received in revised form 18 July 2021; Accepted 2 August 2021
Available online 13 August 2021

technical or chartist analysis. Fundamental analysis assumes that any individual stock has an intrinsic value or equilibrium price that depends on the earning potential of the stock. The earning potential of the stock is determined by such fundamental factors as quality of management, periodic earnings, the outlook for the industry, and the economy. Hence, a fundamental analysis uses financial methodologies to determine if the actual price of the stock is above or below its intrinsic value and make a prediction of its future price (Murphy, 1999; Nti et al., 2020). Meanwhile, technical or chartist analysis assumes that successive stock price changes in individual stock are interrelated, and they tend to repeat the past patterns of price behavior. Technical stock analysis uses statistical or mathematical methods based on historical price changes and trading volumes to identify patterns, trends, and strength within the price data and to predict the direction of price changes (Lin, 2011).

Accurate prediction of a stock price direction makes it possible for investors and traders to decide an exact moment to sell or buy, and so it would help build an optimal investment strategy to decrease the risk of investment. Besides, understanding the direction of a stock price is imperative for investors to beat the market and make profits (Qiu & Song, 2016; Ding & Qin, 2020). In this study, we propose a stock price direction prediction system to improve the accuracy of technical analysis. There are several reasons for using technical analysis in this study. First, fundamental analysis based on intrinsic value or equilibrium price depends on qualitative measurement to a great degree (Murphy, 1999). For instance, the quality of management, the outlook for the industry, the cost of a new business model, the value of the company's brand, etc. are important factors in the intrinsic value of a stock but they are not easily assessed for quantitative technical analysis. Second, fundamental factors that determine the intrinsic value of a stock do not reflect constantly changing stock prices. A company's financial statements such as the balance sheet, income statement, cash flow statement, and financial ratios that are required to calculate the intrinsic value of a stock are usually prepared for such long-term periods as monthly, quarterly, or annually. That long-term financial information is not appropriate to predict a stock price's momentary movements that happen frequently and whimsically. Third, fundamental analysis based on a company's financial statements is vulnerable to various intentional or unintentional interventions, errors, or fraud by humans (Chen & Hao, 2017; Song et al., 2019).

There are multiple methodologies to study the stock market and predict the behavior of stock price movement. Before the flourishing of AI algorithms, statistical models such as regression analysis and time series models such as autoregressive integrated moving average (ARIMA), and its variations: autoregressive conditionally heteroscedastic (ARCH) and generalized autoregressive conditional heteroskedasticity (GARCH) were popular to study stock prices (Chung & Shin, 2018; Henrique et al., 2019; Shah et al., 2019). However, these statistical models do not completely capture the complexity of the stock price data, because stock data are nonlinear, nonstationary, and even seasonally volatile, and so underlying assumptions of these methods are rarely satisfied.

With the advent of big data and the rapid development of artificial intelligence (AI) techniques during recent decades, many studies have employed machine learning (ML) and deep learning (DL) techniques to study the stock market. ML is a subset of AI techniques that enables computer systems to learn from previous experience and improve their behavior for a given task (Chollet, 2018). ML techniques include AdaBoost (AB), random forest (RF), support vector machines (SVM), decision trees, k-nearest neighbour (KNN), neural networks (NN), etc. (Strader et al., 2017; Henrique et al., 2019; Shah et al., 2019). DL is a subset of NN or artificial neural networks (ANN). They are usually described as a collection of connected units, called artificial neurons, organized in layers. DL models include deep neural network (DNN), convolutional neural network (CNN), recurrent neural networks (RNN), generative adversarial networks (GAN), auto-encoder (AE), and many more. Those ML and DL methods attracted a great deal of attention in the past decade for their outstanding performance in processing sequential data including time series because they do not require prespecified data assumptions such as linearity, stationarity, homoscedasticity, or normality (Chen & Hao, 2017; Chung & Shin, 2018; Wang et al., 2018; Basak et al., 2019; Henrique et al., 2019). They can deal complex massive time-series stock price datasets more efficiently than previous methods. Unlike conventional statistical analysis methods that produce answers only according to input rules or algorithms, ML and DL methods train themselves to learn the rules and structure for given data and then produce the new prediction rules (Chollet, 2018). As a result, ML and DL models dramatically improved the accuracy of predictions made from the complex stock price dataset in recent years.

Even with the outstanding successes of the ML and DL algorithms, challenges remain in building an accurate stock price direction prediction model because the framework of the algorithms consists of sequential steps of workflow: data preprocessing, feature engineering, optimization, prediction, and model validation and evaluation. However, we are witnessing a growing number of stock prediction studies adopting ML or DL that excessively center on the main prediction algorithm, and often overlook the importance of feature engineering. As a result, previous studies without appropriate feature engineering and optimization often fail to achieve a good accuracy or underperform conventional stock price prediction models. Even the newest version of deep learning, which removes most feature engineering steps by extracting useful features directly from raw data, requires some feature engineering including data preprocessing because good feature engineering enables prediction algorithms to solve problems more elegantly while using fewer resources with far less data (Chollet, 2018).

We propose a stock price direction prediction system focusing on an enhanced feature engineering process. The feature engineering in this study decomposes the prediction process into three subprocesses: (1) Feature set expansion by generating technical indicators, (2) data preparation and normalization, (3) Optimal feature selection using a hybrid genetic extreme gradient boosting (GA-XGBoost) algorithm. This top-down approach gains insights into each compositional step of the feature engineering process and identifies the importance and function of each step in predicting a stock price direction.

Extreme gradient boosting (XGBoost) has been widely recognized as one of the two most winning algorithms in numerous machine learning competitions since its initial release in 2014. Just as deep learning (DL) algorithms perform well for perceptual problems such as image classification, XGBoost is outstanding for shallow learning problems where structured data is available (Chollet, 2018). Among the 29 top-three winning solutions of Kaggle's competitions in 2015, 17 used XGBoost as a sole predictor or ensembles with DL. Every top 10 winning team in the 2015 KDD Cup competition used XGBoost. The strength of XGBoost is its scalability and great accuracy in all scenarios. The scalability of XGBoost is due to the way the algorithm processes sparse data. It uses a weighted quantile sketch procedure for handling instance weights in approximate tree learning with parallel and distributed computing (Chen & Guestrin, 2016). By integrating scalability with gradient boosting, XGBoost can maximize the prediction performance and processing speed by exploiting every available hardware resource. Even with the successes and practical popularity of XGBoost in machine learning algorithms, few studies used the XGBoost method for stock market predictions (Nobre & Neves, 2019). Being motivated by this, we integrate XGBoost as the main predictor with a genetic algorithm (GA) as the optimizer for feature selection. GA is a well-known evolutionary optimization algorithm for finding near-optimal solutions to large search space problems (Chung & Shin, 2018; Chung & Shin, 2020).

The distinctive, beneficial points of our approach are as follows. First, good performance. The proposed prediction system outperforms benchmark deep learning models in terms of computational cost-effectiveness. Even simple feed-forward deep neural networks require hundreds of nodes and edges to route a single input component such as a coefficient in a vectorial unit through most network units, and as a

Table 1
Recent stock price prediction studies employing AI techniques.

Authors	Dataset	Output	Input Features	Feature Engineering	Feature Dimension	Prediction Methods	Evaluation Metrics
(Patel et al., 2015)	4 Stock prices & Indices	Price, Direction	Technical	Generation	(2474, 10)	ANN, RF, SVM, NB	ACC, F1
(Ballings et al., 2015)	European stock prices	Direction	Financial, Economic, Technical	Generation	(5767, 87)	RF, KF, AdaBoost,	AUC
(Qiu & Song, 2016)	Nikkei stock index	Direction	Technical	Generation	(1707, 13), (1707, 9)	GA + ANN	ACC
(Chen & Hao, 2017)	2 Chinese stock indices	Price	Historical, Technical	Generation, Selection	(1500, 13)	FW-SVM, FW-KNN	MAPE, RMSE
(Chung & Shin, 2018)	Korean stock price index	Price	Historical, Technical	Generation	(4203,10)	GA + LSTM	MSE, MAE, MAPE
(Song et al., 2019)	Korean stock price index	Price increase rate	Historical, Technical	Filtering, Generation	(12783,715)	DNN	ACC
(Wang et al., 2018)	Chinese stock index	Direction	Historical, Technical	Generation	(970, 62)	Deep Learning, EC	ACC, AUC, Recall, F1
(Basak et al., 2019)	10 US & Indian stocks	Direction	Historical, Technical	Generation	(10700, 8)	RF, XGBoost	ACC, AUC, Recall, F1
(Naik & Mohan, 2019)	Indian stock index	Price	Historical, Technical	Generation, Selection	(10 years, 22)	ANN	MAE, RMSE
(Ntakaris et al., 2019)	5 Nordic & 2 US stocks	Direction	Economic, Technical LOB features	Generation, Extraction	(13 Million events, 87)	MLP, CNN, LSTM	F1, RMSE
(Yu & Yan, 2020)	6 Stock indices	Price	Historical	Extension	(2518, m)	LSTM	DA, MRSE, MAPE, CORR
(Chung & Shin, 2020)	Korean stock price index	Price	Historical, Technical	Generation	(4203, 12)	GA + CNN	ACC
(Ampomah et al., 2020)	22 Stock prices	Direction	Historical, Technical	Generation, Extraction	(3774, 45)	EC	ACC, AUC, Recall, F1
(Ding & Qin, 2020)	2 Chinese stock prices	Price	Historical, Technical	Generation	(6112, 7)	LSTM + DRNN	MSE, MAE
(Nabipour et al., 2020a)	4 Iranian stock prices	Price	Technical	Generation	(10 years, 10)	Tree models, NN	MAPE, MAE, RMSE, MSE

result, the computational graph of DL is fundamentally complex and dense (Peter et al., 2017). On the other hand, XGBoost and other tree-based learning algorithms route each sample along a single path from the root to a leaf, which is a small subset of nodes in a decision tree, and so it makes the computational cost of XGBoost cheaper than DL algorithms. An expensive computational cost of numerical simulation can be a major issue in predicting stock price movement where an instantaneous, accurate decision is always required to respond to a stock's price volatility. Available computational resources are continuously increasing but do not always translate into shortening of the design cycle because of the growing demand for better accuracy and the need to simulate larger and more complex systems (Koziel et al., 2014). Accurate simulation of a complex system such as the stock market may be as long as several hours, days, or even weeks, which often makes conventional methods impractical or prohibitive. This fact increases the importance of lower computational costs in stock price prediction.

Secondly, the proposed GA-XGBoost algorithm yields better interpretability than DL models. The GA-XGBoost algorithm can select important features as it repeatedly examines the data and learns relationship and keep the features without losing interpretability as a feature selection. The algorithm depicts the underlying mechanism between input features and the target outcome based on the importance of optimal features. However, DL models are notorious for their interpretability problem because of their black-box structure (Montavon et al., 2018). DL models reduce the dimensionality of feature space by extracting transformed features. Feature extraction, which finds new features by combining original features, usually forms a new feature space with low dimensionality, but loses the interpretability and understandability of the original features.

Third, the proposed prediction system increases prediction accuracy by a great degree by using the enhanced three-stage feature engineering process. This study empirically demonstrates that feature space expansion contributes the most in improvement of prediction accuracy of stock price direction. Expanding the feature space by generating related technical indicators relieves the computational stress of increasing prediction accuracy and achieves better performance than previous stock prediction studies.

The rest of this paper is organized as follows. Section 2 describes recent related works. Section 3 explains the proposed prediction system. Section 4 provides experimental results and discussion, and Section 5 concludes this study.

2. Literature review

In the past decade, there have been an increasing number of stock price prediction studies that use artificial intelligence (AI) techniques. There are several reasons for this intense trend of adopting AI techniques on stock price forecasting. First, stock price data are inherently nonlinear, nonnormal, and nonstationary (Ballings et al., 2015; Chen & Hao, 2017; Chung & Shin, 2018; Wang et al., 2018; Basak et al., 2019; Henrique et al., 2019). Conventional statistical methodologies that use linear statistical models and time-series techniques do not achieve satisfactory results when applied to complex stock price data. Examples of the models that struggle with the nature of the data include autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), autoregressive conditional heteroscedasticity (ARCH). Meanwhile, AI models better use predictors to make accurate prediction targets whether the relationships are linear or nonlinear without the inconvenience of checking underlying statistical assumptions. Furthermore, AI learning models equipped with a solid ability to process big data, for instance, graphics processing units (GPUs), can learn the complex nonlinear relationships quickly and are achieving tremendous successes in various fields such as image classification, speech recognition, text-to-speech conversion (Chollet, 2018; Chung & Shin, 2020), and even in time series financial data analysis.

Table 1 describes recent studies that employed AI techniques for stock price prediction. Among the various AI techniques adopted for stock price prediction, deep learning (DL) techniques have undergone rapid development with remarkable financial time series prediction performance in the past few years. DL is a learning structure that uses

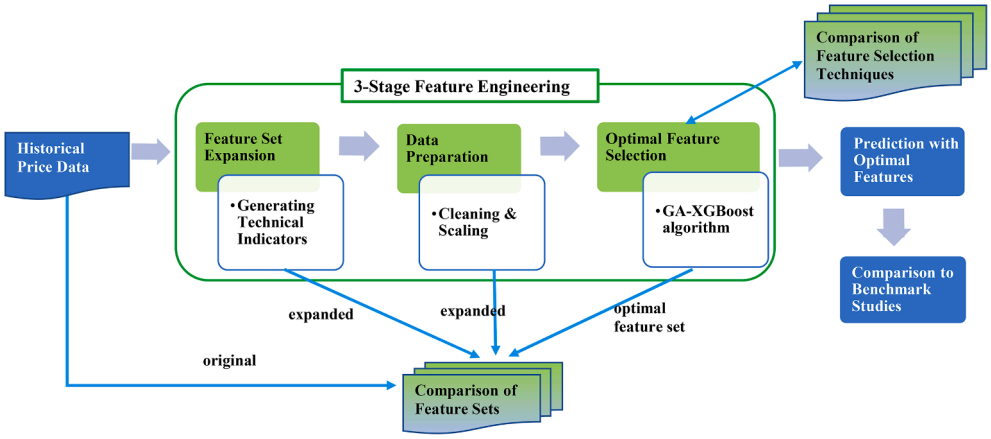

Fig. 1. GA-XGBoost stock price direction prediction system.

Neural Networks (NN) or Artificial Neural Networks (ANN) in which successive hidden layers representing data are stacked on top of each other between the input and output (Chollet, 2018). DL models can be classified into three major types. They are Feedforward Neural Network (FFNN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) (Chollet, 2018; Henrique et al., 2019; Shah et al., 2019). Among DL model variations, Long Short-Term Memory (LSTM) networks, a specific subtype of RNN, have risen to prominence in the stock market prediction studies because of their capability of handling the vanishing gradient problem of long-term time-series data (Chung & Shin, 2018; Yu & Yan, 2020). Chung and Shin (2018), Ntakaris et al. (2019), Ding and Qin (2020), and Yu and Yan (2020) used a type of LSTM to predict stock prices and indices.

In addition to the recently popular DL models above, various ML models remain in great demand for stock price prediction as shown in Table 1. The popular ML models are support vector machine (SVM), Naïve Bayes learning, k-nearest neighbour, and tree-based learning algorithms such as decision trees, extra trees, random forest, and gradient boosting machine (GBM) (Strader et al., 2017; Henrique et al., 2019; Shah et al., 2019). Ballings et al.(2015) and Ampomah et al.(2020) used tree-based ensemble classifiers to predict stock price direction. Chen and Hao (2017) adopted feature weighted SVM with non-linear Gaussian kernel function and information gain to assign different importance to each feature and then predict future stock market indices using a weighted K-NN algorithm.

However, using DL models or ML models solely counting on one predictor does not guarantee the best performance in time series forecasting. Makridakis et al.(2018) evaluated the performance of ML and DL models such as decision trees, multilayer perceptron, and LSTM. They compared them to the performance of conventional time series forecasting methods such as ARIMA, ETS, and Theta using a large subset of 1045 monthly time series used in the M3 Competition. The results showed that ML and DL models underperformed traditional statistical models in terms of accuracy and computation requirements (Makridakis et al., 2018).

Some studies focus on feature engineering, a well-known approach to improve model performance in machine learning using AI techniques. Chen and Hao (2017), Wang et al. (2018), Chung and Shin (2018), Chung and Shin (2020), and Naik and Mohan (2019) generated technical indicators and used them with historical price values. Meanwhile, Patel et al. (2015) and Song et al. (2019) generated technical indicators from the base historical price data and used them as input features without historical prices. With the expansion of feature space by generating technical indicators, some researchers use feature extraction or feature selection to prevent the curse of dimensionality along with the generation of technical indicators. Chen and Hao (2017) used feature selection, but Ntakaris et al.(2019) and Ampomah et al.(2020) used feature extraction after generating an expanded feature set. Many studies have used an expanded input feature space with technical indicators. However, the scope and the number of technical indicators to be included in stock price prediction do not have established protocols yet to the best of our knowledge. A large dataset with many features causes the problem of irrelevant and redundant information, which may significantly degrade the performance of learning algorithms (Yu & Liu, 2003). Paradoxically, when a dataset has many dimensions, some problems can be solved simply and straightforwardly because of the blessing of dimensionality (Kainen, 1997). Hence, 'The blessing of dimensionality' and 'the curse of dimensionality' are two sides of a coin (Gorban & Tyukin, 2018). Being motivated by this, we propose a prediction system with an enhanced feature engineering process that can benefit from the blessing of dimensionality, while preventing the curse of dimensionality.

3. Proposed methodology

The main goal of this study is to build a stock price direction prediction system with an enhanced 3-stage feature engineering process to achieve a great degree of prediction accuracy. Fig. 1 shows the flowchart of the proposed XGBoost stock price direction prediction system. Feature engineering is a well-known approach to improve model performance in the machine learning field. It is a pre-process of constructing input features from raw data based on domain knowledge. Feature engineering includes feature generation, feature transformation, feature extraction, feature selection and feature evaluation and is often data type-specific and application dependent (Dong & Liu, 2018; Li et al., 2018). Feature engineering for a stock price prediction study generally starts with generating additional features because stock price data usually contains only historical prices (Open, Lowest, Highest, and Close) and trading volume, which are combined with additional handcrafted features, to substantially boost model performance. A logical basis of this performance-improving feature set expansion phenomena may be 'the blessing of dimensionality', which describes the phenomena of measure concentration that drastically simplifies some large-dimension machine learning problems (Gorban & Tyukin, 2018). Therefore, we engineer sequentially enhanced features through (1) feature set expansion by generating technical indicators, (2) data preparation including data cleaning and normalization (3) optimal feature set selection by using the hybrid GA-XGBoost algorithm.

3.1. Description of stock price direction prediction

Predicting stock price direction is a binary classification problem in which Y'_{t+1}, the predicted direction of the closing price of day $t+1$, is predicted for X_t, a given set of input features of day t, and then compared

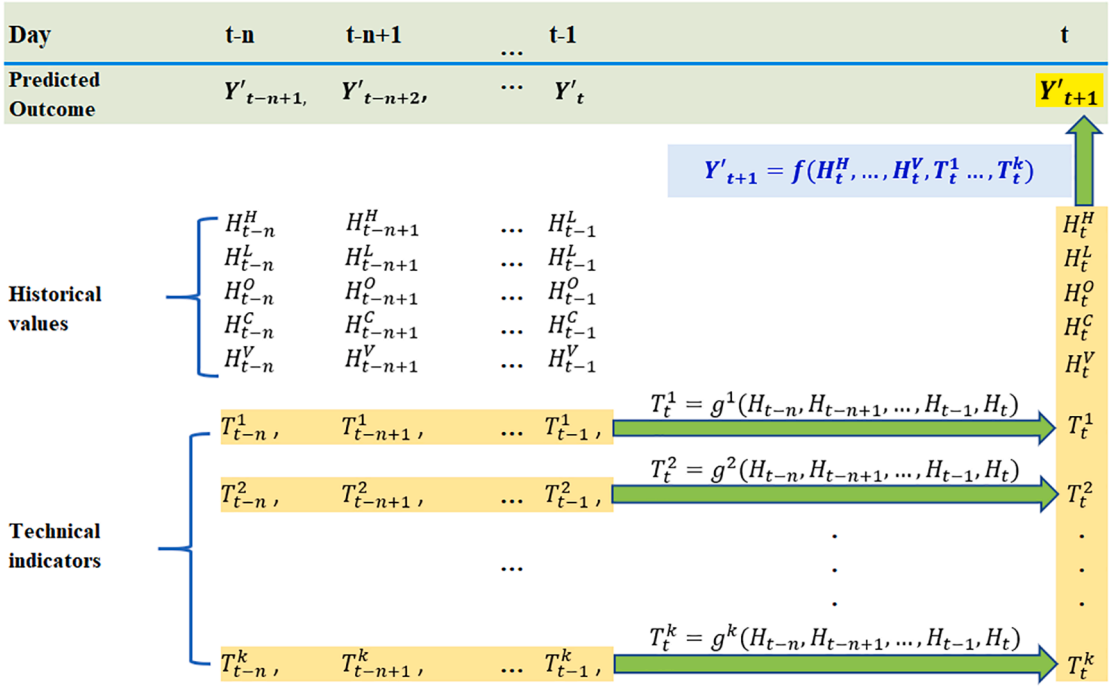

Fig. 2. Illustration of predicting stock price direction.

to the target outcome Y_{t+1} of a binary value 0 or 1. The target outcome in this study is defined as:

$$Y_{t+1} = \begin{cases} 1, & if\ C_t < C_{t+1} \\ 0, & if\ C_t \geq C_{t+1} \end{cases} \quad (1)$$

where C_t, C_{t+1} are the closing price of day t and $t+1$, respectively. So, $Y_{t+1} = 1$, if C_{t+1} moves upward from C_t, and $Y_{t+1} = 0$, otherwise. The predicted outcome Y'_{t+1} is a function of the set of input features X_t. That is,

$$Y'_{t+1} = f(X_t) \quad (2)$$

where f is a nonlinear function that maps a set of input features X_t at day t to binary outcome values $\{0, 1\}$ at day $t+1$ as shown in Fig. 2.

A stock price time series X_t is a Markov process of order n that is affected by n previous states of prices, which can be expressed as:

$$p(x_t|x_{t-1}, x_{t-2}, \cdots, x_1) = p(x_t|x_{t-1}, x_{t-2}, \cdots, x_{t-n}) for\ t > n \quad (3)$$

where $p(A|B)$ represents a conditional probability of event A given event B. This stock price time series X_t is different from the classic Markov processes in which

$$p(x_t|x_{t-1}, x_{t-2}, \cdots, x_1) = p(x_t|x_{t-1}) \quad (4)$$

given the past, present, and future states are all independent. As a result, the next state in the classic Markov process is affected only by the current state, and not by the sequence of previous states. However, in a stock price time series following a Markov process with memory n, the current daily stock prices depend on the past n states stock prices. From the perspective of the Markovian property of stock price data, stock price time series consisting of 5 input features: 'High', 'Low', 'Open', 'Close', and 'Volume' to a specific single day t are limited to predict the next day's stock price direction. Hence, we generate additional technical indicators from these five basic historical price values to take advantage of the past n states price information as shown in Fig. 2.

The original historical values: $H_t^H, H_t^L, H_t^O, H_t^C, H_t^V$ represent the current day$t$'s price values: 'High', 'Low', 'Open', 'Close', and 'Volume', respectively, and k technical indicators $T_t^1, T_t^2, \cdots, T_t^k$ represent the value of nonlinear functions of the past n days' historical values $H_{t-n}, H_{t-n+1}, \cdots, H_{t-1}, H_t$ from day t. Consequently, the predicted outcome of day $t+1$ can be produced as a nonlinear function f of input features of day t which are comprised of historical values $H_t^H, H_t^L, H_t^O, H_t^C, H_t^V$ and k technical indicators, $T_t^1, T_t^2, \cdots, T_t^k$ that reflect the historical information of the days: $t-n, t-n+1, \cdots, t-1, t$, and so the Eq. (2) can be replaced by the following Eq. (5).

$$Y'_{t+1} = f\left(H_t^H, H_t^L, H_t^O, H_t^C, H_t^V, T_t^1, T_t^2, \cdots, T_t^k\right) \quad (5)$$

Table 2
Commonly used technical indicators in stock price prediction studies.

Technical indicators	Formulas
Momentum	$C_t - C_{t-14}$
Relative strength index (RSI)	$100 - \frac{100}{1 + (\sum_{t=0}^{n-1} Up_{t-1}/14)/(\sum_{t=0}^{n-1} Dw_{t-1}/14)}$
Commodity channel index (CCI)	$\frac{M_t - m_t}{d_t \times 0.015} \times 100$
Stochastic %K	$\frac{C_t - LL_{t-14}}{HH_{t-14} - LL_{t-14}} \times 100$
Larry William's (LW) %R	$\frac{H_{14} - C_t}{H_{14} - L_{14}} \times 100$
Moving average convergence divergence (MACD)	$EMA_{12} - EMA_{26}$
Accumulation/Distribution (A/D) oscillator	$\frac{H_t - C_{t-1}}{H_t - L_t} \times 100$

Commodity channel index (CCI)	$\frac{M_t - m_t}{d_t \times 0.015} \times 100$
Stochastic %K	$\frac{C_t - LL_{t-14}}{HH_{t-14} - LL_{t-14}} \times 100$
Larry William's (LW) %R	$\frac{H_{14} - C_t}{H_{14} - L_{14}} \times 100$
Moving average convergence divergence (MACD)	$EMA_{12} - EMA_{26}$
Accumulation/Distribution (A/D) oscillator	$\frac{H_t - C_{t-1}}{H_t - L_t} \times 100$

Note: C_t: the closing price at day t. Up_t: the upward price change at day t. Dw_t: the downward price change at day t. M_t : typicalpriceatday$t$ $= (H_t + L_t + C_t)/3$. m_t : 14 −daymovingaverage $= (\sum_{i=1}^{14} M_{t-i+1})/14$. d_t : 14days'meandeviation $= (\sum_{i=1}^{14} |M_{t-i+1} - m_t|)/14$. LL_t: the lowest low price in the last t days. HH_t: the highest high price in the last t days. Dw_t: the downward price change at day t. H_t: the highest price at day t. L_t: the lowest price at day t. EMA_t: 14-day exponential moving average at day $t = C_t \times 2/14 + EMA_{t-1} \times (1 - 2/(14+1))$.

where $H_t^H, H_t^L, H_t^O, H_t^C, H_t^V$ are 'High', 'Low', 'Open', 'Close', 'Volume', and $T_t^1, T_t^2, \cdots, T_t^k$ are k technical indicators of day t. Here, day t's j-th technical indicator is

$$T_t^j = g^j\left(H_{t-n}, H_{t-n+1}, \cdots, H_{t-1}, H_t\right) \tag{6}$$

where g^j is a nonlinear function of the past n days' historical values: $H_{t-n}, H_{t-n+1}, \cdots, H_{t-1}, H_t$ from day t.

3.2. Feature set expansion by generating technical indicators

The first step of the proposed 3-stage feature engineering is expanding the feature set by generating technical indicators. There are two reasons for using technical indicators. The first reason is to integrate the indication of past states' stock prices with the current state's price to predict the next day's stock price direction based on the Markovian n-states memory property as described in 3.1. The second is to use technical indicators to expand the input feature set to capture 'the blessing of dimensionality'. When a dataset is high dimensional, some problems can be solved simply and straightforwardly because of 'the blessing of dimensionality' (Kainen, 1997). By using the high dimensionality blessing phenomenon, this study expands the feature set by generating technical indicators. The scope and number of technical indicators to be included in predicting stock price direction remains to be further researched, but most previous studies employing technical indicators for stock price prediction commonly used the following technical indicators as shown in Table 2 (Qiu & Song, 2016; Wang et al., 2018; Basak et al., 2019; Chung & Shin, 2018; Chung & Shin, 2020). For this reason, we create two expanded feature sets. The expanded feature set 1 adopts 7 popular technical indicators from the literature as shown in Table 2 and is created for comparison purposes.

A brief explanation of 7 commonly used technical indicators from the literature is provided below.

- Momentum indicator: it measures the difference between the closing prices at day t and day t-14 to determine the rise (strength) or fall (weakness) of a stock's price.
- Relative strength index (RSI): it determines whether the stock is overbought or oversold. RSI ranges from 0 to 100. RSI indicates that the stock is overbought as it gets closer to 100, and indicates the stock is oversold as it gets closer to 0.
- Commodity channel index (CCI): it measures the difference between the current price and the historical average price. When the CCI is above zero, it indicates the price is above the historical average. Conversely, when the CCI is below zero, the price is below the historical average.
- Stochastic %K: it measures the position of a stock's closing price of a stock in the high- and the low-price range over a given period as a stochastic oscillator. It ranges from 0 to 100. If the value is 80 or above, it indicates the overbought range, and if the value is 20 or lower, then it is considered oversold.
- Larry William's (LW) %R: it measures the level of a market's closing price with the highest price for 14 days. It ranges from −100 to 0. When its value is above −20, it is a sell signal and when its value is below −80, it is a buy signal.
- Moving average convergence divergence (MACD): an indicator that compares two moving averages of prices by subtracting the 26-day Exponential Moving Average (EMA) from the 12-day EMA. It sends signals when it crosses above (to buy) or below (to sell) its signal line. The speed of crossing the signal line implies whether a market is overbought or oversold.
- Accumulation/Distribution (A/D) oscillator: a cumulative indicator that uses volume and price to assess whether a stock is being accumulated or distributed. A rising A/D oscillator line implies a rising price trend, and a falling A/D line implies a downtrend of the stock price.

The expanded feature set 2 intends to maximally harness the blessing of dimensionality and so generates 67 technical indicators out of the original 5 input features and they are described in Table A.1 in Appendix A. All technical indicators in this study are computed using the functions provided by TA-Lib library (TA-Lib, 202)). TA-Lib is widely used by trading software developers seeking to perform technical analysis of financial market data. All generated technical indicators belong to one of 6 function groups of indicators, which are selected by reviewing expert knowledge and previous studies (Qiu & Song, 2016; Wang et al., 2018; Basak et al., 2019; Chung & Shin, 2018; Chung & Shin, 2020; Nabipour et al., 2020a). They are Overlap Studies, Momentum Indicators, Volume Indicators, Volatility Indicators, Price Transform, and Cycle Indicators as shown in Table A.1.

3.3. Data preparation

Along with the feature space expansion using technical indicators, the data pre-processing for the following major prediction step is necessary. Data pre-processing employs data cleaning, data transformation, and data normalization in this study. For the data cleaning step, unwanted and unnecessary features are removed from the expanded feature set. The adjusted closing price is removed before the expansion of the feature set because the closing prices contains that information, making it redundant. A continuous feature, the closing price, is removed after the binary target outcome is created. Data transformation as a pre-processing step is required for the GA-XGBoost prediction step. Machine learning models find appropriate representations for their input data. Transformations of the data make it more amenable to the task at hand, such as a classification task (Chollet, 2018). All input data values are transformed into a homogenous numerical array to be fed into the GA-XGBoost. Historical stock price data have continuous variables with different measurement units for volume and prices. Furthermore, some technical indicators added to the original historical data are rate measurements. Data normalization using the min–max formula in Eq. (7) is helpful to process different scaled features together. Specifically, the normalization of input feature values helps the gradient descent converge much faster (Chung & Shin, 2018) and preserves precisely all relationships in the data, and thereby, it does not introduce any bias (Devan & Khare, 2020).

$$x^{'} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{7}$$

3.4. Optimal feature selection by the GA-XGBoost algorithm

A high-dimension dataset may cause overfitting, unnecessary increasing memory consumption, expensive computational costs, and debasing performance to a learning algorithm (Li et al., 2018; Huang et al., 2020). To overcome problems caused by data with many dimensions, feature extraction and feature selection are popular dimensionality-reduction techniques. Feature extraction builds new features from combinations of original features but loses the interpretability and understandability of the originals. Feature selection keeps some important features without losing interpretability. The feature selection process removes irrelevant and redundant features, increases learning efficiency (Yu & Liu, 2003), and improves prediction accuracy. Feature selection methods can be classified into filter, wrapper, and embedded methods (Yu & Liu, 2003; Miao & Niu, 2016; Ramírez-Gallego et al., 2017; Li et al., 2018). Filter methods select features based on the statistical properties of each feature without using any learning algorithm. On the other hand, wrapper and embedded methods combine feature selection and a predetermined learning algorithm to select an optimal subset of features for the algorithm. Filter methods generally run noniterative computations on the dataset and are more computationally efficient than the wrapper or embedded methods. Meanwhile, wrapper or embedded methods form all possible combinations of

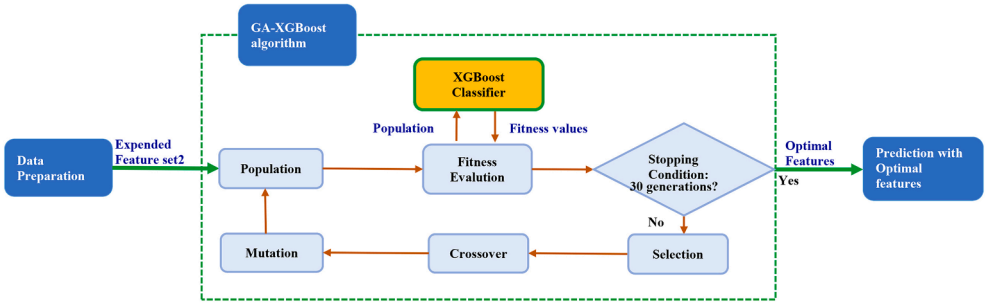

Fig. 3. The GA-XGBoost algorithm.

features and evaluate each subset of features for the best performance with the learning algorithm. Hence, wrapper or embedded methods can achieve a better performing feature selection. The wrapper method conforms to the main goal of this study, which is to propose a system to predict stock price direction better. The proposed hybrid GA-XGBoost algorithm is a wrapper-based feature selection method to select an optimal feature set for classification by removing irrelevant and redundant features.

A genetic algorithm is a search and optimization method that mimics the evolution mechanism of natural selection. It finds a near-optimal solution in multidimensional and large search spaces according to 'the survival of the fittest' principle. For that reason, GA has been widely used to select an optimal feature set (De Stefano et al., 2014; Chung & Shin, 2018). The process of a genetic algorithm begins with a randomly selected set of chromosomes. These chromosomes are representations of possible solutions to the problem. According to the attributes of the problem, units of each chromosome are encoded as bits, characters, or numbers. These units in each chromosome are called genes and the value of each gene is changed randomly through genetic algorithm operations during evolution. The set of chromosomes during a stage of evolution are called a population. In addition to the construction of a population and its chromosomes, another important condition of GA is a fitness function. A fitness function is used to evaluate the goodness or performance of each chromosome by the XGBoost algorithm. As shown in Fig. 3, Once the initial population is set up, GA proceeds to repeat a loop that is composed of fitness evaluation by XGBoost, stopping condition check, selection, crossover, and mutation until it reaches the predefined stopping criterion: 30 generations. There are several options for the stopping criterion of a GA. A predefined absolute number of generations, a predefined target fitness score, or a minimum fitness score change between generations are possible options for a stopping criterion of a GA. In this study, 30 generations of GA are used as a stopping condition to compare the performance of GA-XGBoost to other competitive ML feature selection techniques under the same condition in Section 4.4. The number 30 was selected empirically as we saw no significant accuracy improvement with more iterations in the compared ML classifiers. If the stopping condition is met, GA stops the iteration of a loop that is called a generation. If the stopping condition is not met, the GA selects some superior chromosomes according to their fitness values calculated by the fitness function. The selected chromosomes become the parents of a new population in the next generation, and some parts of the parents' chromosomes are swapped to produce offspring. This crossover operation helps to inherit the superior characteristics of the parents. The next process, called a mutation, selects some chromosomes, and randomly changes the genes. This mutation secures the diversity of the population and prevents the local optimum problem (Chung & Shin, 2020).

XGBoost stands for extreme gradient boosting (Chen & Guestrin, 2016). It has become one of the best ML algorithms with its speedy learning and exceptional accuracy. Chen and Guestrin (2016) described the XGBoost algorithm and their explanation is summarized as follows. The general tree ensemble model with a dataset with n examples and m features has an output that is the sum of independent trees' predictions as shown:

$$\widehat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{8}$$

where $F = \{f(x) = w_{q(x)}\}(q : R^m \rightarrow T, w \in R^T)$ is the space of regression trees. Here, $q$is the structure of each tree and $T$is the number of leaves in the tree. Each f_k corresponds to an independent tree structure $q$and leaf weights w. To learn the set of additive functions in the tree ensemble model, we minimize the objective function as shown:

$$L(\phi) = \sum_i l\left(\widehat{y}_i, y_i\right) + \sum_k \Omega(f_k) \tag{9}$$

where l is a differential convex loss function, and $\Omega(f) = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$ is a model complexity for regularization. To optimize the objective function of the general tree ensemble model above, the XGBoost model uses $\widehat{y}_i^{(t)}$as the prediction of the i-th instance at the t-th iteration, and adds f_i term to the objective function as shown:

$$L^{(t)} = \sum_i l\left(y_i, \widehat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{10}$$

and this objective function of XGBoost can be written by Taylor's second-order approximation as:

$$L^{(t)} \simeq \sum_i \left[l\left(y_i, \widehat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) \tag{11}$$

where g_i, h_i are first- and second-order gradient statistics on the loss function $l(y_i, \widehat{y}^{(t-1)})$. The objective function at step t can be simplified by removing the constant terms as:

$$L^{(t)} \simeq \sum_i \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) \tag{12}$$

For a fixed structure $q(x)$, the optimal weight is

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{13}$$

and the corresponding optimal value is given by

$$\overline{L}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{14}$$

This equation is used as a scoring function to measure the quality of a tree structure g. The score is like the impurity score for evaluating decision trees, except that it is derived for a wider range of objective functions.

As an integrated hybrid approach, the GA-XGBoost algorithm searches a near-optimal or optimal feature set by collecting and upgrading feature sets for better accuracy over generations. The optimal

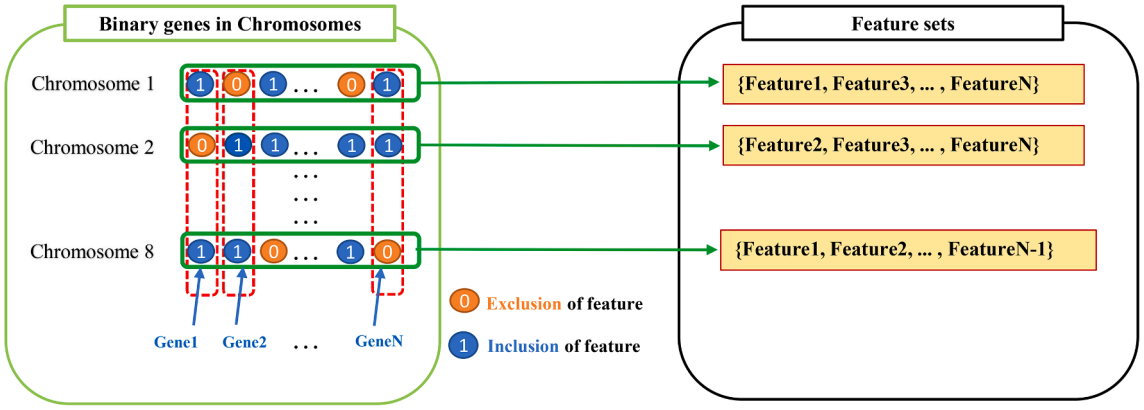

Fig. 4. Binary genes in Chromosomes of GA-XGBoost.

Fig. 5. KOSPI historical closing price.

feature set maximizes prediction accuracy with a minimal number of features. The initial population has 8 chromosomes, and each chromosome contains the same number of binary genes as the number of relevant features as described in Fig. 4. Each binary gene has either 0 for the exclusion or 1 to include a corresponding feature in the selected feature set. After a generation is formed by the genetic algorithm's selection, crossover, and mutation operations, then each chromosome in the population is evaluated by the XGBoost classifier's predefined fitness function. The XGBoost classifier trains the training dataset with selected features, measures the classification accuracy using each feature set, and returns the accuracy with the feature set to the GA process.

4. Experimental results and discussion

4.1. Experimental environment and data description

All experiments in this study are performed in Python (version 3.8) on Windows 10 with Intel Core i7-10875, 2.30 GHZ processor and 32 GB RAM. We used 4203 trading days' worth of data regarding the Korea composite stock price index 200 (KOSPI) to predict the stock index's up or down movement direction on the next day, allowing us to intensively test our 3-stage feature engineering approach while simultaneously comparing our performance with a benchmark. This dataset was previously used in a stock price movement prediction study (Chung & Shin, 2020). They applied multi-channel convolutional neural networks (CNN) to predict the movement direction of the stock index of the following day with the use of the same 7 technical indicators as our expanded feature set 1 that is generated for comparison purposes. The five historical values: High Price (High), Low Price (Low), Opening Price (Open), Closing Price (Close), and Trading Volume (Volume) were downloaded from the Yahoo finance website (https://finance.yahoo.com) for the period: 1/4/2000 through 12/31/2016, which is 4203 trading days. Table 3 in Appendix B shows the summary statistics of KOSPI data for the period. Fig. 5 is the graph of the daily 'Closing' price of KOSPI data for 16 years.

4.2. Feature set expansion by generating technical indicators

From the original 5 features of historical prices: 'High', 'Low', 'Open', 'Close', 'Volume', two expanded datasets are created by generating technical indicators as described in Section 3. As shown in Table 2 in Section 3.2, the expanded feature set 1 adds 7 commonly used technical indicators in previous stock price prediction studies for comparison purposes. Among 7 technical indicators added to the expanded feature set 1, commodity channel index (CCI), moving average convergence/divergence (MACD), momentum (MOM), relative strength index (RSI), stochastic %K (STOK), Larry Williams' %R (WILLR) are momentum indicators and only A/D oscillator (ADOSC) is a volume indicator. The expanded feature set 2 adds 67 technical indicators created from the original 4 input features to realize the blessing of dimensionality. Table A.1 in Appendix A shows the full list of the extended feature set. The categories of 67 technical indicators and the number of features belonging to each category are Overlap studies (11), Momentum indicators (40), Volatility indicators (3), Volume indicators (2), Price Transform Indicators (4), Cycle indicators (7). As described in Section 3, original historical values represent the current day's prices X_t, and technical indicators retrospect the past 14 days price information from day t in this study. Hence, a predicted outcome of day $t+1$ is a nonlinear function of input features of day t, which comprises the historical prices of day $t$and the technical indicators of the period of days $t-14, t-13, \cdots, t-1, t$.

4.3. Data cleaning and normalization

After expanding the feature set, the feature 'Close' is replaced by the binary output feature 'Prediction'. The target outcome of the proposed model is the movement direction of the stock price from the previous day to the current day. So, the binary outcome feature 'Prediction' with values 0 and 1 is created. As a result of removing 78 null valued trading days out of 4203 trading days, 71 input features and 4125 trading days remained in the expanded feature set 2. The entire dataset is divided into a training dataset (first 80%, 3300 trading days) and a holdout dataset (last 20%, 825 trading days). The holdout dataset is reserved for the evaluation of the final optimal feature set and to compare performance comparison to the benchmark. The training dataset is split into the first 85% and the last 15% of 4125 trading days. The first 85% is used to train for learning the GA-XGBoost algorithm and the last 15% is used for

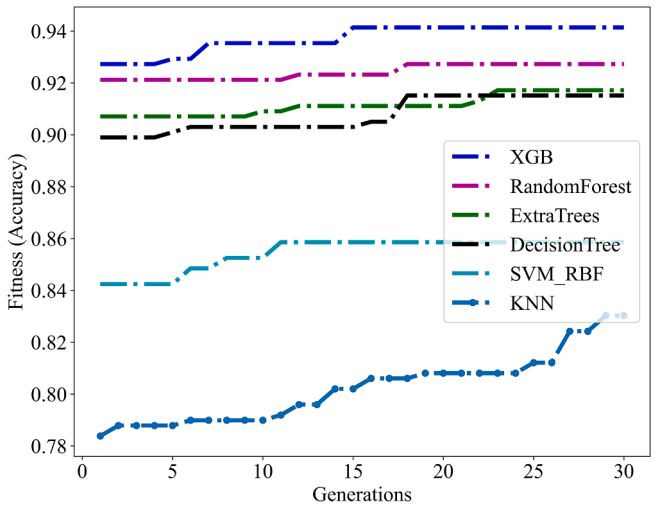

Fig. 6. Accuracy comparison of feature selection techniques.

Table 6
Performance comparison of feature sets.

Feature Sets	Number of Input features	Mean Accuracy	Standard Deviation
Original set	4	0.6049	0.0180
Expanded set 1	11	0.7519	0.0087
Expanded set 2	71	0.9328	0.0038
Optimal set	33	0.9315	0.0059

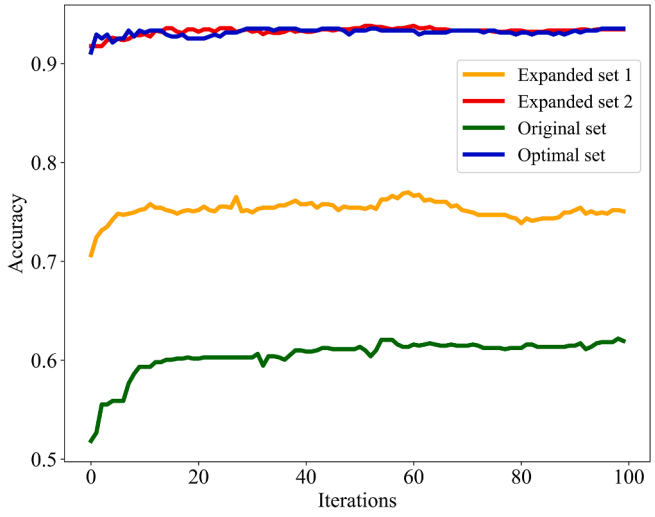

Fig. 7. Accuracy comparison of feature sets.

validating the model. All values are normalized to covert the original values into the range of $[0, 1]$ by the equation (7).

4.4. Optimal feature set selection by the GA-XGBoost algorithm

The expanded set 2 of 71 input features is input to select an optimal feature set by GA-XGBoost. As described in Section 3, a population is composed of 8 chromosomes and each chromosome has 71 genes representing a feature in the reduced feature set. That is, the inclusion of each feature is represented by gene value 1 and the exclusion of the feature is represented by gene value 0 in a chromosome as shown in Fig. 4, and each gene value is randomly assigned by the GA through its operations. The crossover rate is 0.5, the mutation rate is 0.375, and 30 generations are used for this experiment. We compared the performance of the GA-XGBoost algorithm to the performance of GA-based hybridized nonlinear machine learning (ML) classifiers. They are Extra Trees (ET), Decision Tree (DT), Random Forest (RF), SVM with RBF kernel (SVM_RBF), K-Nearest Neighbour (KNN) classifiers. The first 85% (2805 trading days) of the training dataset of 3300 trading days is used for training only and the next 15% (495 trading days) of the training dataset is used for validation purposes in the GA-XGBoost algorithm. The experimental results of XGBoost and other hybridized classifiers are summarized in Table 4 in Appendix B.

After 30 generations of GA, the XGBoost shows the best classification accuracy of 94.14%. RF has the second-best accuracy of 92.73% and ET algorithm follows with 91.72%. XGBoost selects 33 features, and RF and ET select 30 and 34 features for the optimal feature set, respectively. In the comparison of the training times of the 6 feature selection techniques, XGBoost follows DT and KNN but has a faster time to train than ET, RF, and SVM_RBF. KNN and SVM_RBF classifiers have a fairly short time to train but their accuracies are significantly worse than other classifiers. ET and RF classifiers have much longer training time with lower classification accuracy than XGBoost. Fig. 6 draws the comparison of classification accuracies of 6 feature selection techniques. XGBoost, shortened as XGB, remarkably outperforms other hybridized classifiers from the beginning to the end of 30 generations. The experimental results of hybridized optimal feature selection methods based on a genetic algorithm verify that XGBoost is the best classifier for predicting stock price direction with its optimal feature sets. Table 4 in Appendix B and Fig. 6 are good evidence to select the XGBoost classifier as the best feature selection algorithm based on the accuracy, the number of optimal features and processing time.

To check the statistical differences in the accuracy of 6 classifiers, Tukey's HSD (honestly significant difference) test is performed. All 15 pairwise mean differences of classification accuracy are tested at the 5% level of significance. As shown in Table 5 in Appendix B, XGBoost classifier has a significantly better mean classification accuracy than the other non-linear classifiers. As a result, the null hypotheses that the mean classification accuracy of XGBoost is equal to any other classifier's mean accuracy are all rejected with p-values of 0.001. In Table 5 in Appendix B, all feature selection techniques' mean classification accuracies are significantly different from each other at $\alpha = 0.05$ except for the pair of Extra Trees and Decision Tree.

4.5. Comparison of feature sets

As the result of applying the proposed 3-stage feature engineering process to the original historical price data (Original set), the expanded feature set 2 (Expanded set 2) and optimal feature set (Optimal set) and the expanded feature set 1 (Expanded set 1) was created to compare with the results of previous studies. As an ensemble learning model, XGBoost tests each feature set with 100 iterations of the individual base learner, the decision tree model. The mean classification accuracy and standard error of each feature set are obtained from testing the holdout dataset 825 trading days (the last 20% of 4125 valid trading days) and summarized in Table 6. The expanded set 1 incudes original 4 historical values and 7 commonly used technical indicators in previous stock price studies (Qiu & Song, 2016; Wang et al., 2018; Chung & Shin, 2018; Chung & Shin, 2020; Basak et al., 2019; Nabipour et al., 2020a) and achieves 75.19% prediction accuracy. The expanded set 2 added 67 additional technical indicators to the original set and drastically increases classification accuracy to 93.28%. The optimal set has a slightly lower accuracy of 93.15% than the expanded set 2, but the number of input features is significantly decreased from 71 to 33 as shown in Table 6.

Fig. 7 visualizes the prediction accuracies of each feature set over 100 iterations of boosting base learning in XGBoost. The graphs of the expanded feature set 2 and the optimal feature set do not show any clear difference. The accuracy of the expanded set 1 with 11 input features consisting of 4 historical price values and 7 technical indicators is much worse than the other three feature sets.

Table 8
Test performance metrics on KOSPI data.

Performance Metrics (%)				
Accuracy	Precision	Recall	F1	AUC
93.82	93.75	94.41	94.08	93.79

Table 9
Confusion matrix on KOSPI data.

		Target outcome		
		Down	Up	Sum
Predicted outcome	Down	369	27	396
	Up	24	405	429
	Sum	393	432	825

Tukey HSD (honestly significant difference) test results in Table 7 in Appendix B support the findings of the above results. The mean prediction accuracy of optimal feature set and expanded feature set 2 are not statistically different at 5% significance level. However, all other pairwise feature sets are significantly different in mean prediction accuracy.

4.6. Performance metrics and feature importance of optimal feature set

For the final test of the GA-XGBoost prediction system with 33 optimal features, the reserved holdout dataset (the last 825 trading days) is used. The performance of the proposed prediction system is measured by the following metrics.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{16}$$

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

$$AUC = \int_0^1 \frac{TP}{TP + FN} d\frac{FP}{FP + TN} \tag{19}$$

where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives.

The prediction accuracy for our proposed model with 33 optimal input features reaches 93.82% as shown in Table 8. The final evaluation also achieves precision 93.75%, F1 score 94.08%, recall 94.41%, and AUC 93.79% on KOSPI stock index data.

The confusion matrix of the predicted outcome and target outcome on the holdout dataset is described in Table 9.

We checked the feature importance of the optimal feature set. Fig. 8 ranks all input features by their important scores. 16 of the 33 selected important features are momentum indicators in category 2, and 2 features are original historical values: 'Volume' (trading volume) and 'Low' (lowest price). The most important features in optimal feature set are '2Stok' (the highest stochastic relative strength index) in the last 14 days and '2Stod' (the lowest stochastic relative strength index) in the last 14 days with feature importance scores 225 and 193, respectively as shown in Fig. 8. Stochastic relative strength index (SRSI) is a momentum indicator in category 2 that ranges between 0 and 1 and measures overbought or oversold of stocks in a predefined period (Murphy, 1999; Vanstone & Hahn, 2010). The higher SRSI value is considered overbought, and the lower SRSI value is considered oversold. The next important feature is '2Bop' (the balance of power) with a score of 123

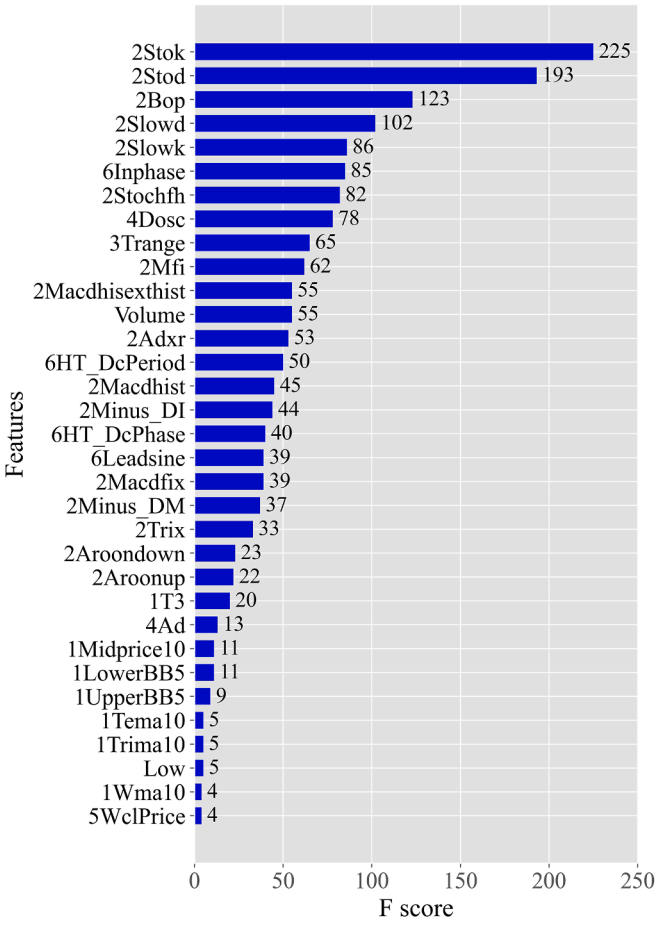

Fig. 8. Feature importance of optimal feature set.

and it is another momentum indicator in category 2. This '2Bop' indicates the direction of a trend measuring the strength of buyers against sellers by assessing the ability of each side to drive prices to an extreme level (Boxer, 2014).

4.7. Comparison to benchmark studies

The performance metrics of the proposed GA-XGBoost prediction system are compared to recent 10 benchmark studies on the prediction of the stock price direction. The results are presented in Table 10 in Appendix B. Among the 10 comparisons, 8 comparisons are executed on the same datasets used in the benchmark studies. The prediction period of the GA-XGBoost system is extended from 1 day to 7 days, 15 days, or 30 days to verify the predictive power of the proposed GA-XGBoost algorithm. The proposed GA-XGBoost prediction system outperforms 7 of the 8 previous studies on the same datasets (Patel et al., 2015; Qiu & Song, 2016; Song et al, 2019; Ampomah et al., 2020; Chung & Shin, 2020; Long et al., 2020; Yang et al., 2020) with all three prediction periods. For the Shanghai Stock Exchange (SSE) index dataset for the period of 2012 through 2015, GA-XGBoost slightly underperforms the benchmark (Wang et al., 2018) on the following day's stock price direction prediction. However, GA-XGBoost's performance metrics for the 15-day period are significantly improved from the 1-day prediction performance, and so they are parallel to the benchmark result.

Eight of 10 benchmark studies adopt DL techniques such as ANN, CNN, RNN, LSTM, and BiLSTM; the other 2 studies used ML techniques: tree-based Ensemble Learning (Ampomah et al., 2020) and Naïve Bayes classifier (Patel et al., 2015). The last two benchmark studies used RNN (Nabipour et al., 2020b) and LSTM with Feature Extension (FE), Recursive Feature Elimination (RFE) and Principal Component Analysis

Table 11
Default hyperparameters of DL models.

Epochs	Batch size	Learning rate	Dense layers	Dense nodes	Activation function
100	50	1.0E-05	1	20	Relu

(PCA) (Shen & Shafiq, 2020), respectively. Especially, (Shen & Shafig, 2020) achieve remarkable performances: 93% accuracy and 96% precision and recall on 3558 Chinese stocks. However, because of the difficulty of accessing the same datasets of the 2 benchmark studies (Nabipour et al., 2020b; Shen & Shafiq, 2020), the mean performance metrics of all GA-XGBoost practices over the benchmark studies are calculated and compared.

4.8. Comparison to deep learning models

4.8.1. Performance and computational cost comparison to deep learning models

As subtypes of RNN that can capture the past information from the previous nodes, LSTM and hybrid LSTM network models are among the most popular deep learning algorithms for stock price prediction in the recent literature (Li & Bastos, 2020). Whereas RNN cannot process extended time-span information that is more than a few time steps, LSTM can determine the amount of information to be retained from node to node by the forgetting gate. Because of this unique structure and function, LSTM can capture long-term past information of more than 1000-time steps (Chung & Shin, 2018). Hence, it does not cause a vanishing gradient problem that quickly loses the long-time dependent information from the past nodes when the number of time steps increases. As a result, LSTM models produce outstanding predictability with nonlinear and complex stock price times data (Chung & Shin, 2018; Yu & Yan, 2020; Hao & Gao, 2020; Ding & Qin, 2020).

The prediction performance and computational cost of a model are crucial factors in stock price prediction. To beat the market and make profits depends on an accurate prediction of stock prices and a prompt decision to respond to a stock price's ephemeral movement and to capture the exact moment to sell or buy. For this reason, the computational time is a better option than computational space or energy in measuring the computational cost of an algorithm.

To compare the performance and computational cost of GA-XGBoost to DL models, 3 LSTM models and 3 Bidirectional LSTM (BiLSTM) models are constructed as representative DL models and compared for prediction accuracy and computational time in two steps. First, DL models with default hyperparameters are compared to the XGBoost algorithm. The default hyperparameters are provided in Table 11.

Second, DL models with optimized hyperparameters using Bayesian optimization are compared with the performance of the GA-XGBoost algorithm.

There are various hyperparameter optimization techniques, e.g., grid search, random search, and evolutionary optimization algorithm. In contrast to other methods, Bayesian optimization (BO) provides mathematically grounded tools to reason about model uncertainty (Gal & Ghahramani, 2016) and an adaptive paradigm to sample the design space more efficiently for identifying the global optimum (Zhang et al, 2020). For this reason, we adopted BO to optimize the hyperparameters of DL models. In our experiment, we used the 71 input features of the expanded feature set 2. Out of 4125 trading days, the first 80% (3300 trading days) is used for training and validation and the next 20% (825 trading days) is set aside for the final test. The first 85% (2805 trading days) of 3300 trading days is used for training only and the next 15% (495 trading days) is used for validation. Training and validation

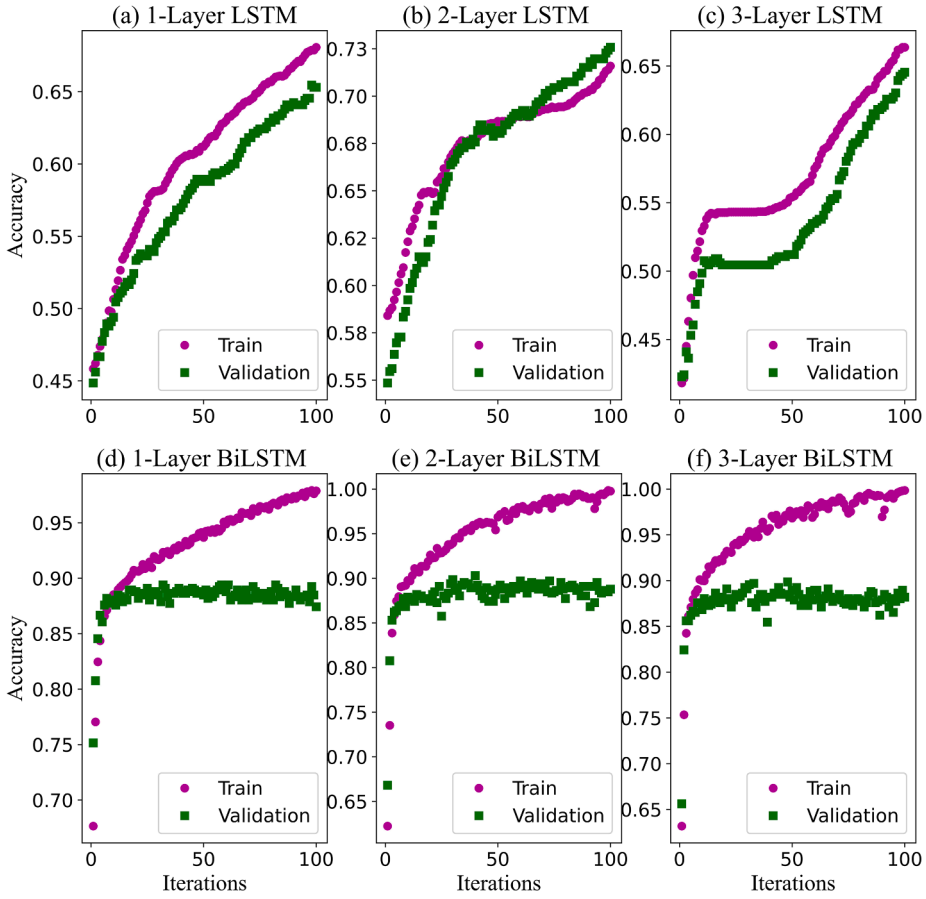

Fig. 9. Training and valid accuracy of 6 DL models with default hyperparameters.

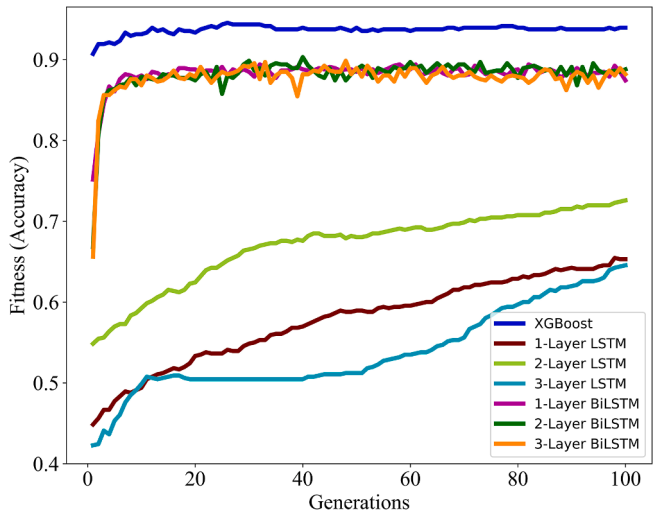

Fig. 10. Validation accuracy comparison to DL models with default hyperparameters.

accuracies of 100 epochs are depicted in Fig. 9. LSTM models and BiLSTM models have different patterns and degrees of accuracy. The 2-layer LSTM model has the best validation accuracy among LSTM models, but training and validation accuracies intersect each other. Three BiLSTM models have higher validation accuracies than LSTM models, but they tend to overfit as the number of epochs increases.

Fig. 10 shows the direct comparison of validation accuracy of XGBoost to that of 6 DL models over 100 epochs. The number of LSTM and BiLSTM layers does not affect the model's performance. LSTM models have improving accuracies as the number of epochs increases. BiLSTM models quickly reach higher accuracies than 3 LSTM models. Nevertheless, all DL models underperform XGBoost in terms of classification accuracy.

Table 12 in Appendix B shows how the performance of the XGBoost algorithm with default hyperparameters compares with 6 DL models. XGBoost's validation accuracy of 93.68% and test accuracy of 93.82% are significantly better than all 6 DL models' accuracies. The processing time, which measures the computational cost of an algorithm, for 100 epochs by XGBoost is 1.01 s, substantially faster than that of 6 DL models. Tukey's HSD test is performed to check the statistical differences in the mean validation accuracy, and the result is shown in Table 13 in Appendix B. All 21 pairwise mean differences of validation accuracy are tested at the 5% level of significance. The null hypothesis that the mean validation accuracy of XGBoost is equal to any other mean accuracy of 6 DL models is rejected with p-values of 0.001. The results shown in Table 13 in Appendix B prove that mean accuracy differences among 3 BiLSTM models are not significant, but the mean accuracy differences among 3 LSTM models are significant. Conclusively, the XGBoost algorithm substantially outperforms all DL models with default hyperparameters.

Although the DL model is popular because of its successful learning ability on stock price time-series data, the model's performance greatly depends on hyperparameters that control the behavior of the model's learning process. Hence, hyperparameter optimization, which configures the optimal values of hyperparameters such as learning rate, number of nodes, and number of layers, etc., is necessary to produce the best performance of a DL model. To improve the performance for the competing LSTM and BiLSTM models, Bayesian Optimization (BO) is used to tune the hyperparameters: learning rate, the number of dense layers, the number of dense nodes, and activation function of dense

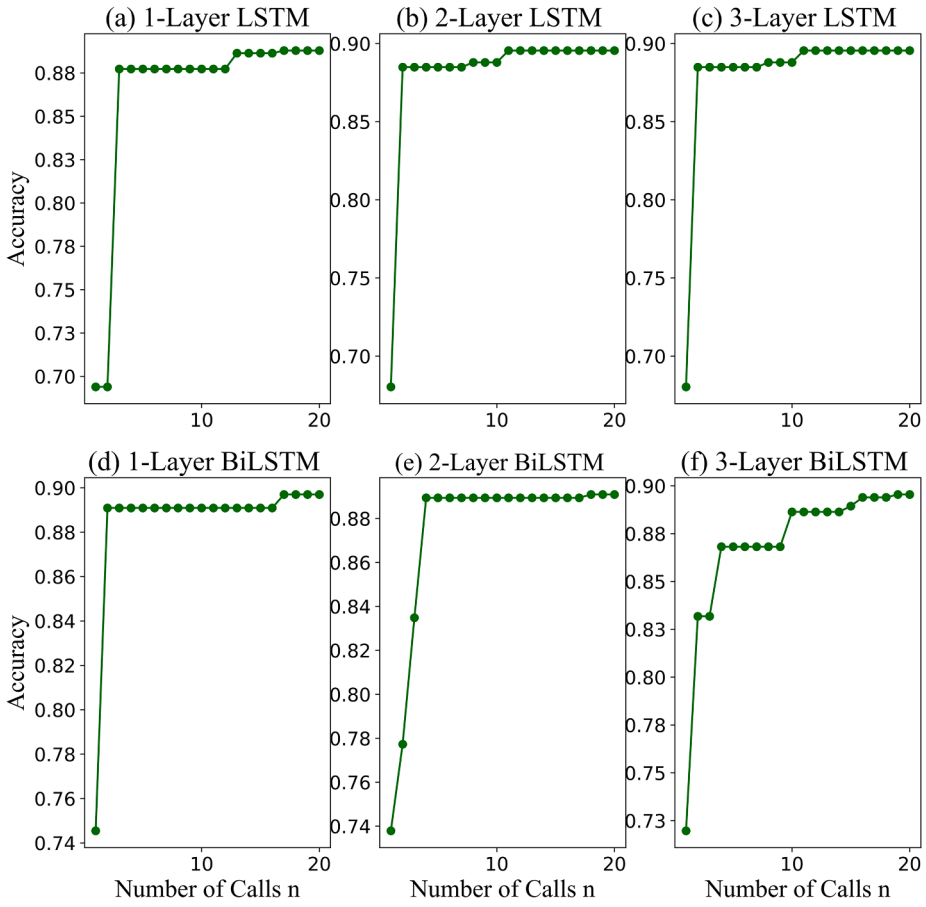

Fig. 11. Convergence plots of Bayesian optimization (BO) of DL models.

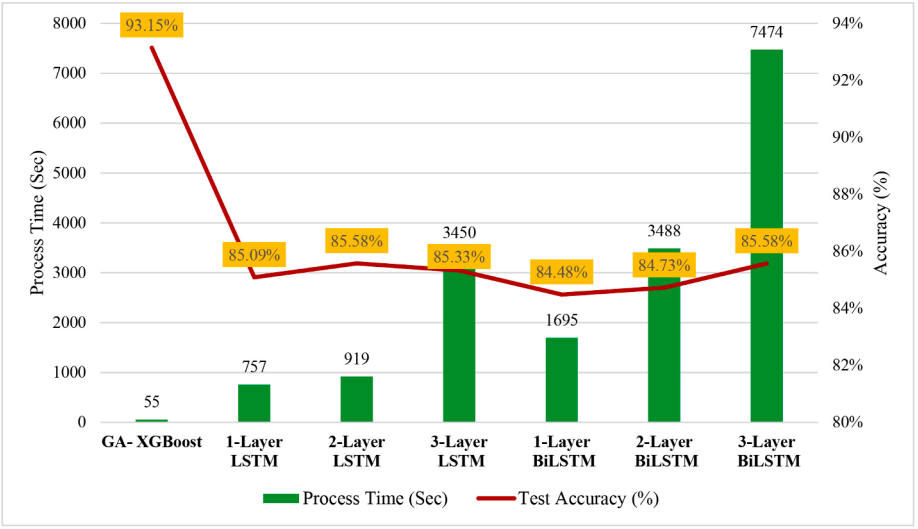

Fig. 12. Performance comparison to DL models with optimized hyperparameters.

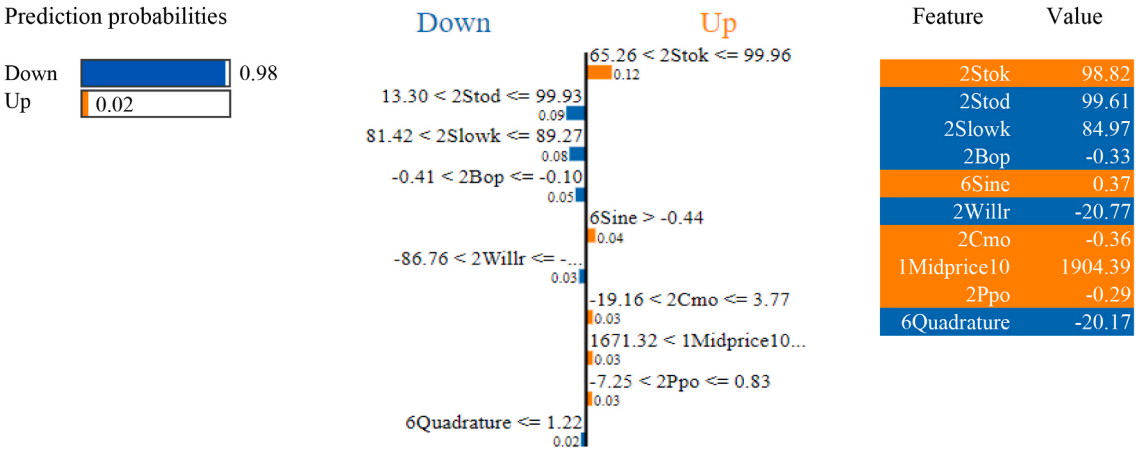

Fig. 13. LIME instance interpretation plot of the XGBoost's prediction result.

layers. The algorithm of BO is described as follows (Guo et al., 2019):

Algorithm 1 : The Bayesian optimization algorithm.

Input: Initial dataset $D_0 = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ with stop condition K fixed number of iterations, where x_n is a decision vector, and y_n is an output observation.
Output: x_t such that $x_t = \arg\max a(x)$
For k = 1, 2, …, to K, do:
1. Utilize the data set D_{t-1} to train a Gaussian process model of $f(x)$, and obtain a posterior distribution over functions: $p(f|D_t) = \frac{p(D_t|f)p(f)}{p(D_t)}$, where $p(f)$ and $p(D_t)$ are a priori probability distribution of f and D_t, and $p(D_t|f)$ is the likelihood function of f.
2. Maximize the acquisition function $a(x)$ which based on $p(f|D_t)$ to find the new point x_t such that $x_t = \arg\max a(x)$
3. Calculate the value of the Gaussian process model of the selected point $x_t : f(x_t)$
4. Update the data set: $D_t = D_{t-1} \cup \{x_t, y_t\}$

End

Fig. 11 shows the convergence plots of 6 DL models in the Bayesian optimization process. For 20 calls of the optimization processes, all DL models have convergent accuracies but could not reach an accuracy of 90%.

The result of the Bayesian parameter optimization is summarized in Table 14 in Appendix B. Validation accuracies and test accuracies of DL models are significantly improved from the default hyperparameter models. Especially, 3 LSTM models' parameter optimization made substantial accuracy increments: 20.37, 14.43, and 15.88 percentage points for each LSTM model. However, the BO process increases the number of dense layers of each DL model to achieve better accuracy, which leads to an extremely long processing time of each model as shown in Table 14 in Appendix B. This time-consuming optimization process represents the notoriously expensive computational cost of DL models (Koziel et al., 2014; Maji & Mullins, 2018).

Finally, the prediction accuracy and processing time of each DL model with optimized hyperparameters are measured on the test dataset and compared to that of GA-XGBoost in Fig. 12. GA-XGBoost outperforms by a substantial margin all LSTM and BiLSTM models in accuracy and processing time.

4.8.2. *Interpretability comparison to deep learning models*

Despite the remarkable performances, most ML and DL algorithms lack transparency and interpretability about their training process and prediction result because of their black-box nature learning process. Accordingly, model interpretability is an important factor in evaluating GA-XGBoost and its competing DL models along with performance and computational time. Interpretability of a model provides a transparent explanation about the prediction result and a basis for a constructive application in the following step. Here, a critical criterion for interpretability is that a model must provide a qualitative explanation of the relationship between the input features and the response (Ribeiro et al., 2016; Jalali et al., 2020). In the recent research about the

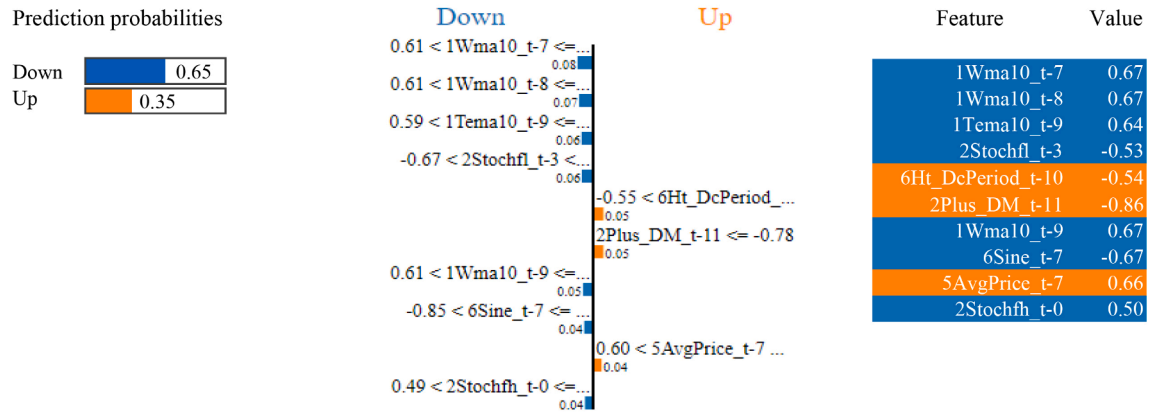

Fig. 14. LIME instance interpretation plot of LSTM model's prediction result.

Fig. 15. SHAP instance interpretation plot of the XGBoost's prediction result.

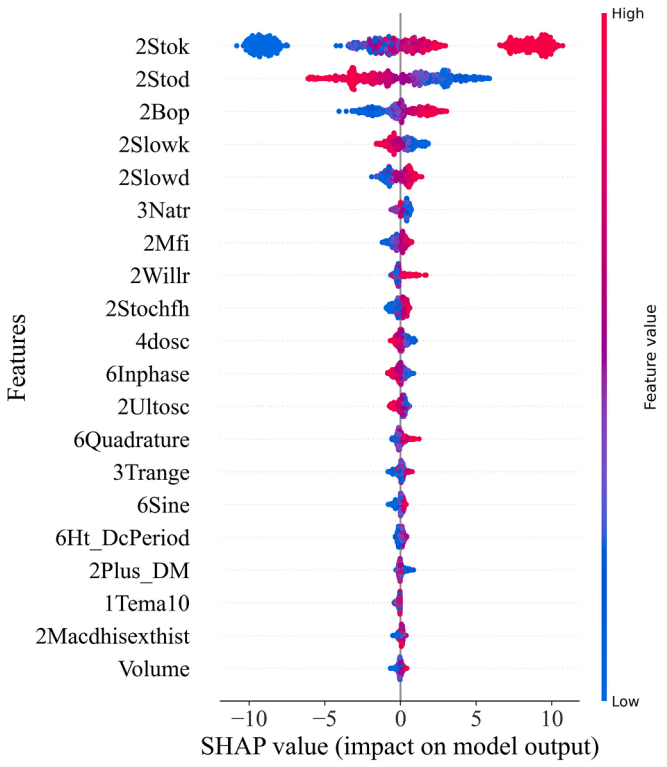

Fig. 16. SHAP feature importance plot of the XGBoost prediction result.

interpretability of AI models, known as Explainable Artificial Intelligence (XAI), various studies commonly focus on the relationship between input features and the model's output by measuring the importance of a feature or set of features. Several techniques measure the importance of features for a model's interpretability. Local Interpretable Model-Agnostic Explanations (LIME) proposed by Ribeiro et al. (2016) and Shapley Additive Explanations (SHAP) proposed by Lundberg and Lee (2016) are representative tools in XAI (Jalali et al., 2020).

The interpretability of XGBoost and a DL model is compared using the LIME and SHAP techniques. The default parameter XGBoost is evaluated against a single layer LSTM model with 32 hidden units, a rectified linear activation function (Relu), and a 0.2 dropout rate. A sigmoid activation function is used for the final output layer. Fig. 13 shows the LIME explanation of XGBoost's prediction result of the 345th instance which is randomly selected from the test dataset. LIME implements a local surrogate interpretable model such as Lasso or Decision Tree to explain individual sample's prediction of a given black-box ML or DL model, rather than providing a global explanation of the output of a model.

XGBoost predicts there is a 98% probability that the 345th instance will move downward and a 2% probability the movement will be upward. The actual outcome was downward. The important features behind upward direction are 'Stok', 'Sine', 'Cmo', 'Midprice', and 'Ppo'. The important features related to downward direction are 'Stod', 'Slowk', 'Bop', 'Willr', and 'Quadrature'. Each feature's contribution to this prediction is ranked in the order of importance in the middle bar plot. The rightmost feature-value plot in Fig. 13 shows the actual value of each feature in the test dataset.

Meanwhile, LIME's explanation on the same 345th instance's prediction result by the vanilla LSTM model is quite different in prediction probability and important features selection as shown in Fig. 14. Downward direction with 65% or upward direction with 35% is the predicted result. Furthermore, the set of features affecting this predicted movement is different from the result of XGBoost.

While LIME implements an interpretation of an individual instance's prediction based on a local surrogate model, SHAP provides local and global explanations for its underlying model based on Shapley values, which describes a way to distribute the total gains to all features, assuming they all collaborate (Lundberg & Lee, 2016). Fig. 15 shows the SHAP important features for the same 345th instance's prediction made by XGBoost. This SHAP force plot explains how important features can affect the prediction probability 0.99 of the downside direction. The features 'Slowd', 'Sine', 'Inphase', 'Stok', and 'Quadrature' in blue lead the output to downward direction and 'Stod', 'Slowk', and 'Bop' in red lead the output to upward direction.

Fig. 16 shows the global feature importance of XGBoost model based on Shapley values of 20 features. This SHAP feature importance plot shows a similar result to Fig. 8, which ranks important features based on their F-1 scores in the XGBoost learning algorithm. As described in

Section 4.6, the most important features in an optimal feature set are '2Stok' (the highest stochastic relative strength index) in the last 14 days and '2Stod' (the lowest stochastic relative strength index) in the last 14 days again. The next important feature is '2Bop' (the balance of power) which is the same as in Fig. 8 in Section 4.6.

In summary, XGBoost's interpretability of a randomly selected instance measured by LIME and SHAP is consistent and robust, as shown in Table 15 in Appendix B. The prediction probabilities are 0.98 in LIME and 0.99 in SHAP, respectively. The important features selected by both techniques are also akin. The five highlighted features out of 8 are in common. However, LIME's interpretation of the same instance's prediction result by the LSTM model is different from XGBoost. The prediction probability of downward movement is merely 0.65, and that the affecting features of the prediction are different from the features selected by the XGBoost model. LSTM model even selects the same feature at different time steps as important features. This different result can be ascribed to the reconstruction of input data shape in 3-dimension to be fed into tensor operations of DL models (Chollet, 2018).

LIME can provide only local interpretation for a specific instance, but SHAP can be used for global interpretation of a model output as well as the local interpretation of an instance (Jalali et al., 2020). In Table 16 in Appendix B, the SHAP global feature importance of the XGBoost model is similar to the findings in Fig. 8 in Section 4.6, as shown in the leftmost column. The top 13 out of 20 important features of XGBoost are selected by SHAP global feature importance criterion again and that the sum of F-1 scores of these 13 important features is more than 80% of the total F-1 score of the top 20 features. This result implies that the robustness of XGBoost's interpretability is supported by SHAP techniques. However, SHAP's explainer does not support LSTM models trained on 3-dimensional stock price data (Jalali et al., 2020). Because of this technical limitation, the interpretability of LSTM models could not be assessed.

We explored and compared the interpretability of the XGBoost and LSTM model using model-agnostic techniques: LIME and SHAP. Model-agnostic interpretation methods treat a model as a black-box and create an interpretable surrogate model to produce a local approximation of the relationship between an instance and the prediction or calculate the global contribution of each feature to all outputs of the model (Jeyakumar et al., 2020; Jalali et al., 2020). In terms of the definition of interpretability by which users should have the ability to understand and reason about the model output, DL models are notorious for their drawbacks in transparency and functionality in contrast to machine learning methods (Chakraborty et al., 2017; Shi et al., 2019). Specifically, model functionality, providing semantically meaningful post-hoc explanations of what a model has done, is measured by identifying the important features contributing the most toward an output in this section. As a result, it is experimentally verified that the proposed GA-XGBoost algorithm yields better interpretability than a representative LSTM model.

5. Conclusion

Accurate prediction of stock price direction is pivotal information for all people who are interested in the stock market. So there have been numerous studies with various stock price movement predictions, but it is not easy to achieve accurate price direction predictions. According to the survey on the stock market prediction studies for the period from 1993 through 2017, more than 80 percent of the 90 studies used ML and DL models (Strader et al., 2017; Henrique et al., 2019). The share of studies using deep learning models as their baseline methodology has been continually increasing, and now reaching approximately 50 percent. Deep learning, as a specific subfield of machine learning, has achieved complete breakthroughs in various perceptual problems (Chollet, 2018). Specifically, LSTM networks are the top-notch deep learning analysis tool for stock price prediction in the recent literature because they produce satisfactory predictability in a nonlinear complex financial time series problem. Yet, limited objective evidence is available regarding ML or DL models' relative performance as a standard financial time series prediction tool (Makridakis et al., 2018).

This study proposes a stock price direction prediction system using the GA-XGBoost algorithm with an enhanced 3-stage feature engineering process. Compared to some benchmark studies in the literature, the proposed prediction model achieves better performances with more flexibility as the prediction period can be changed arbitrarily. This study shows the successful performance in stock price direction prediction is largely from the proposed feature engineering process, which is composed of feature set expansion, data preparation, and optimal feature set selection. Particularly, the most significant accuracy increment comes from the feature expansion by adding 67 technical indicators to the basic historical stock price data. The prediction accuracy of the expanded feature set 2 of KOSPI dataset is 93.28% and it is increased by 32.79 percentage points from the original feature set of accuracy 60.49% and increased by 18.09 percentage points from the expanded feature set 1 of accuracy 75.19%, which we used to reproduce a benchmark study adopting commonly used technical indicators. Therefore, this finding makes good experimental evidence of the blessing of dimensionality phenomenon (Kainen, 1997) through feature set expansion. Another achievement of this study is producing a parsimonious optimal feature set to relieve the curse of dimensionality. The optimal feature set of KOSPI data in this study has only 33 features, which are substantially fewer than 71 features of the expanded feature set 2, while matching the desired prediction accuracy. In general, a parsimonious feature set is more desirable than a larger feature set because a model with a smaller number of features can perform better with lower computational cost, less overfitting, and easier interpretability. We empirically proved through this study that a successful stock price direction prediction performance does not depend on a sole prediction method. Instead, it is shown that a good prediction performance largely depends on feature engineering processes, being deliberately combined with a baseline learning algorithm, that can make a good balance and harmony between the curse of dimensionality and the blessing of dimensionality.

There are a few shortcomings. First, our experiments used only a predefined number of technical indicators, which is 67 in 6 categories. However, the number and category of technical indicators to be included in a stock prediction study should be studied further. Second, model optimization is another well-known method to improve model performance in machine learning and deep learning with feature engineering. Most ML and DL models can be improved through optimization of the hyperparameters of the prediction model. Some research showed that hyperparameters optimization improves the model performance using such optimizers as genetic algorithm or Bayesian optimization for stock price prediction (Chung & Shin, 2018; Chung & Shin, 2020). It would be worth adopting a hyperparameter optimization to improve prediction accuracy.

CRediT authorship contribution statement

Kyung Keun Yun: Conceptualization, Methodology, Software, Validation, Formal analysis, Data curation, Writing – original draft, Writing - review & editing, Visualization. **Sang Won Yoon:** Conceptualization, Investigation, Writing - review & editing, Resources, Supervision. **Daehan Won:** Conceptualization, Investigation, Writing - review & editing, Resources, Supervision, Project administration.

Declaration of Competing Interest

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

Appendix A

Table A1
Technical indicators used in the study.[1]

Category	Technical Indicators	Description
Category 0: Historical Prices	High	Highest Daily Price
	Low	Lowest Daily Price
	Open	Opening Price
	Close	Closing Price
	Volume	Trading Volume
Category 1:Overlap Studies Indicators	1BBUPPER	Bollinger Bands Upper
	1BBMIDDLE	Bollinger Bands Middle
	1BBLOWER	Bollinger Bands Lower
	1DEMA	Double Exponential Moving Average
	1MIDPOINT	Midpoint over period
	1MIDPRICE	Midpoint Price over period
	1SMA	Simple Moving Average
	1 T3	Triple Exponential Moving Average (T3)
	1TEMA	Triple Exponential Moving Average
	1TRIMA	Triangular Moving Average
	1WMA	Weighted Moving Average
Category 2: Momentum Indicators	2ADX	Average Directional Movement Index
	2ADXR	Average Directional Movement Index Rating
	2APO	Absolute Price Oscillator
	2AROONDOWN	Aroon Down
	2AROONUP	Aroon Up
	2AROONOSC	Aroon Oscillator
	2BOP	Balance of Power
	2CCI	Commodity Channel Index
	2CMO	Chande Momentum Oscillator
	2DX	Directional Movement Index
	2MACD	Moving Average Convergence/Divergence
	2MACDSIG	Moving Average Convergence/Divergence
	2MACDHIST	Moving Average Convergence/Divergence
	2MACDEXT	MACD with controllable MA type
	2MACDEXTSIG	MACD with controllable MA type
	2MACDEXTHIST	MACD with controllable MA type
	2MACDFIX	Moving Average Convergence/Divergence Fix 12/26
	2MACDFIXSIG	Moving Average Convergence/Divergence Fix 12/26
	2MACDFIXHIST	Moving Average Convergence/Divergence Fix 12/26
	2MFI	Money Flow Index
	2MINUS_DI	Minus Directional Indicator
	2MINUS_DM	Minus Directional Movement
	2MOM	Momentum
	2PLUS_DI	Plus Directional Indicator
	2PLUS_DM	Plus Directional Movement
	2PPO	Percentage Price Oscillator
	2ROC	Rate of change: ((price/prevPrice)-1)*100
	2ROCP	Rate of change Percentage: (price-prevPrice)/prevPrice
	2ROCR	Rate of change ratio: (price/prevPrice)
	2ROCR100	Rate of change ratio 100 scale: (price/prevPrice)*100
	2RSI	Relative Strength Index
	2SLOWK	Stochastic %K High
	2SLOWD	Stochastic %K Low
	2STOCHFH	Stochastic Fast High

(*continued on next page*)

Table A1 (*continued*)

Category	Technical Indicators	Description
	2STOCHFL	Stochastic Fast Low
	2STOK	Stochastic Relative Strength Index High
	2STOD	Stochastic Relative Strength Index Low
	2TRIX	1-day Rate-Of-Change (ROC) of a Triple Smooth EMA
	2ULTOSC	Ultimate Oscillator
	2WILLR	Williams' %R
Category 3: Volatility Indicators	3ATR	Average True Range
	3NATR	Normalized Average True Range
	3TRANGE	True Range
Category 4: Volume Indicators	4AD	Chaikin A/D Line
	4DOSC	Chaikin A/D Oscillator
Category 5: Price Transform	5AVGPRICE	Average Price
	5MEDPRICE	Median Price
	5TYPPRICE	Typical Price
	5WCLPRICE	Weighted Close Price
Category 6: Cycle Indicators	6HT_DCPERIOD	Hilbert Transform - Dominant Cycle Period
	6HT_DCPHASE	Hilbert Transform - Dominant Cycle Phase
	6INPHASE	Hilbert Transform - Phasor Components Inpahse
	6QUADRATURE	Hilbert Transform - Phasor Components Quadrature
	6SINE	Hilbert Transform - SineWave
	6LEADSINE	Hilbert Transform - LeadSineWave
	6HT_TRENDMODE	Hilbert Transform - Trend vs Cycle Mode

[1] https://mrjbq7.github.io/ta-lib.

Appendix B

Table 3
Summary statistics of KOSPI data.

Statistics	High	Low	Open	Close	Volume
count	4203	4203	4203	4203	4203
mean	1439.69	1419.90	1431.07	1430.34	421,262
std	541.67	539.82	541.31	540.86	193,012
min	472.31	463.54	466.57	468.76	3016
25%	868.83	852.97	862.05	861.06	302,800
50%	1588.73	1566.83	1581.60	1579.21	379,800
75%	1958.69	1939.53	1950.15	1948.43	487,150
max	2231.47	2202.92	2225.95	2228.96	2,379,300

Table 4
Performance comparison of feature selection techniques hybridized with GA. (XGBoost's performance metrics are in bold)

Metrics	**XGBoost**	Extra Trees	Decision Tree	Random Forest	SVM_RBF	KNN
Best Accuracy	**0.9414**	0.9172	0.9152	0.9273	0.8586	0.8303
Mean Best Accuracy	**0.9371**	0.9114	0.9078	0.9242	0.8546	0.8024
S.D. of Best Accuracy	**0.0053**	0.0039	0.0066	0.0027	0.0062	0.0130
Total Training Time	**55.0285**	174.2382	7.8683	368.2192	60.2230	19.1140
Mean Training Time	**1.8343**	5.8079	0.2623	12.2739	2.0074	0.6371
S.D. of Training Time	**0.1030**	0.1415	0.0212	0.5441	0.0388	0.0450
Number of Features	**33**	34	33	30	23	26

Table 5
Tukey HSD test of mean accuracy differences of feature selection techniques (α = 5%). (Bold font indicates statistical significance of mean accuracy difference between XGBoost and other techniques)

group1	group2	Mean difference	p-value	95% CI lower bound	95% CI upper bound	reject Ho
XGBoost	Extra Trees	−2.5657	0.0010	−3.1023	−2.0290	**True**
XGBoost	Decision Tree	−2.9293	0.0010	−3.4659	−2.3926	**True**
XGBoost	Random Forest	−1.2862	0.0010	−1.8229	−0.7495	**True**
XGBoost	SVM_RBF	−8.2492	0.0010	−8.7858	−7.7125	**True**
XGBoost	KNN	−13.4680	0.0010	−14.0047	−12.9314	**True**
Extra Trees	Decision Tree	−0.3636	0.3751	−0.9003	0.1730	False
Extra Trees	Random Forest	1.2795	0.0010	0.7428	1.8161	True
Extra Trees	SVM_RBF	−5.6835	0.0010	−6.2202	−5.1468	True
Extra Trees	KNN	−10.9024	0.0010	−11.4390	−10.3657	True
Decision Tree	Random Forest	1.6431	0.0010	1.1064	2.1798	True
Decision Tree	SVM_RBF	−5.3199	0.0010	−5.8565	−4.7832	True
Decision Tree	KNN	−10.5387	0.0010	−11.0754	−10.0021	True
Random Forest	SVM_RBF	−6.9630	0.0010	−7.4996	−6.4263	True
Random Forest	KNN	−12.1818	0.0010	−12.7185	−11.6452	True
SVM_RBF	KNN	−5.2189	0.0010	−5.7555	−4.6822	True

Table 7
Tukey HSD Test of mean accuracy differences of feature sets (α = 5%). (Bold font implies no significant difference of mean accuracy)

group1	group2	Mean difference	p-value	95% CI lower bound	95% CI upper bound	reject Ho
Expanded1	Expanded2	18.0881	0.0010	17.8718	18.3044	True
Expanded1	Optimal	17.9653	0.0010	17.7490	18.1816	True
Expanded2	Optimal	−0.1228	0.3768	−0.3391	0.0935	**False**

Table 10
Comparison of GA-XGB performance metrics to benchmark studies. (Bold value represents the highest in the metric)

Dataset	Authors	Methods	Performance Metrics (%)				
			Accuracy	Precision	Recall	F1	AUC
Infosys (INFY) stock (2003–2012)	(Patel et al., 2015)	Naïve-Bayes	90.19	N/A	N/A	90.50	N/A
	GA-XGB	1-day	91.19	95.06	88.17	91.49	91.43
		15-day	**93.20**	92.75	90.40	**91.56**	92.76
		30-day	93.15	92.59	87.72	90.09	91.93
Tokyo Stock Exchange (Nikkei) index (2007–2013)	(Qiu & Song, 2016)	GA-ANN	81.27	N/A	N/A	N/A	N/A
	GA-XGB	1-day	91.44	89.69	95.60	92.55	90.91
		15-day	**94.19**	97.29	94.30	95.77	94.12
		30-day	92.97	94.05	96.73	95.37	89.22
Shanghai Stock Exchange (SSE) index (2012–2015)	(Wang et al., 2018)	Deep + Ensemble learning	91.00	90.30	**93.30**	91.80	**94.90**
	GA-XGB	1-day	89.39	89.00	91.75	90.36	89.17
		15-day	**92.05**	93.26	91.21	**92.22**	92.07
		30-day	89.02	**93.55**	87.00	90.16	89.39
Korean stock price index (KOSPI) (2000–2016)	(Song et al., 2019)	Deep learning + 715 input features	84.80	N/A	N/A	N/A	N/A
	GA-XGB	1-day	**93.38**	93.90	94.08	93.99	93.31
		15-day	93.13	93.22	94.11	93.66	93.04
		30-day	90.15	90.67	91.01	90.84	90.09
KOSPI index (2000–2016)	(Chung & Shin, 2020)	Multi-channel CNN	73.74	N/A	N/A	N/A	N/A
	GA-XGB	1-day	**93.82**	93.75	94.41	94.08	93.79
		15-day	93.07	93.83	93.63	93.73	93.00
		30-day	87.67	86.94	91.03	88.94	87.34
Bank of America (BAC) stock (2005–2019)	(Ampomah et al., 2020)	Ensemble learning	84.63	84.29	85.82	85.05	**92.80**
	GA-XGB	1-day	92.03	89.82	**94.89**	92.29	92.01
		15-day	**92.81**	**93.57**	93.79	**93.68**	92.65
		30-day	90.46	91.67	92.09	91.88	90.13
CITIC security stock (2017–2018)	(Long et al., 2020)	CNN + BiLSTM	75.89	N/A	N/A	N/A	73.10
	GA-XGB	1-day	87.80	87.18	87.18	87.18	87.78
		7-day	91.46	93.33	84.85	88.89	90.38
		15-day	**93.90**	95.00	92.68	93.83	**93.90**
S&P 500 (2010–2017)	(Yang et al., 2020)	CNN + LSTM	63.16	62.81	62.49	62.50	N/A
	GA-XGB	1-day	90.96	93.20	90.14	91.65	91.05
		7-day	92.51	93.38	95.85	94.60	90.55
		15-day	**92.76**	**94.10**	**96.10**	**95.09**	89.95
Iranian stocks	(Nabipour et al., 2020b)	RNN	90.12	N/A	N/A	90.24	90.16
3558 Chinese Stocks	(Shen & Shafiq, 2020)	FE + RFE + PCA + LSTM	**93.00**	**96.00**	**96.00**	N/A	N/A
Mean performance of GA-XGB			91.67	92.40	92.50	92.42	91.14

Table 12
Performance comparison to DL models with default hyperparameters.(XGBoost's performance metrics are in bold)

Metrics	**XGBoost**	1-Layer LSTM	2-Layer LSTM	3-Layer LSTM	1-Layer BiLSTM	2-Layer BiLSTM	3-Layer BiLSTM
Validation Accuracy	**0.9368**	0.6530	0.7257	0.6454	0.8742	0.8879	0.8818
Test Accuracy	**0.9382**	0.6472	0.7115	0.6945	0.8436	0.8315	0.8279
Processing Time (Sec)	**1.01**	6.65	8.62	11.41	8.35	12.57	17.34

Table 13
Tukey HSD test of mean accuracy differences of DL models with default hyperparameters (α = 5%). (Bold font indicates statistical significance of mean accuracy difference between XGBoost and other techniques)

group1	group2	Mean difference	p-value	95% CI lower bound	95% CI upper bound	Reject Ho
XGBoost	LSTM1	−35.8702	0.001	−37.3864	−34.3541	**True**
XGBoost	LSTM2	−25.6566	0.001	−28.1727	−25.1404	**True**
XGBoost	LSTM3	−39.8414	0.001	−41.3576	−38.3253	**True**
XGBoost	BiLSTM1	−5.4429	0.001	−6.9591	−3.9268	**True**
XGBoost	BiLSTM2	−5.5414	0.001	−7.0576	−4.0253	**True**
XGBoost	BiLSTM3	−5.9884	0.001	−7.5045	−4.4722	**True**
LSTM1	LSTM2	9.2136	0.001	7.6975	10.7298	True
LSTM1	LSTM3	−3.9712	0.001	−5.4874	−2.4551	True
LSTM1	BiLSTM1	30.4273	0.001	28.9111	31.9434	True
LSTM1	BiLSTM2	30.3288	0.001	28.8126	31.8449	True
LSTM1	BiLSTM3	29.8818	0.001	28.3657	31.3980	True
LSTM2	LSTM3	−13.1848	0.001	−14.7010	−11.6687	True
LSTM2	BiLSTM1	21.2136	0.001	19.6975	22.7298	True
LSTM2	BiLSTM2	21.1152	0.001	19.5990	22.6313	True
LSTM2	BiLSTM3	20.6682	0.001	19.1520	22.1843	True
LSTM3	BiLSTM1	34.3985	0.001	32.8823	35.9146	True
LSTM3	BiLSTM2	34.3000	0.001	32.7838	35.8162	True
LSTM3	BiLSTM3	33.8530	0.001	32.3369	35.3692	True
BiLSTM1	BiLSTM2	−0.0985	0.900	−1.6146	1.4177	False
BiLSTM1	BiLSTM3	0.5455	0.900	−2.0616	0.9707	False
BiLSTM2	BiLSTM3	−0.4470	0.900	−1.9631	1.0692	False

Table 14
Bayesian optimization results of DL models.

Metrics	1-Layer LSTM	2-Layer LSTM	3-Layer LSTM	1-Layer BiLSTM	2-Layer BiLSTM	3-Layer BiLSTM
Learning Rate	1.06E-04	3.15E-04	1.60E-04	4.62E-05	1.39E-05	5.33E-04
Dense Layers	5	4	10	5	9	10
Dense Nodes	339	201	310	512	279	157
Activation Function	sigmoid	relu	relu	relu	relu	relu
Optimization Time	757(sec)	919(sec)	3450(sec)	1695(sec)	3488(sec)	7474(sec)
Validation Accuracy	0.8879	0.8955	0.8985	0.8970	0.8909	0.8955
Test Accuracy	0.8509	0.8558	0.8533	0.8448	0.8473	0.8558
Test Accuracy Increase by BO	0.2037	0.1443	0.1588	0.0306	0.0564	0.0279

Table 15
Local instance interpretability comparison. (Bold font indicates the same features selected by both techniques)

Interpretability Technique	LIME		SHAP
Learning Model	**XGBoost**	LSTM	**XGBoost**
Prediction Probability	P(Down) = 0.98	P(Down) = 0.65	P(Down) = 0.99
Important Features	**2Stok**	1WMA10_t-7	**2Stod**
	2Stod	1Wma10_t-8	2Slowd
	2Slowk	1Tema10_t-9	**2Slowk**
	2Bop	2Stochfl_t-3	**2Bop**
	6Sine	6Ht_DcPeriod_t-10	**6Sine**
	2Willr	2Plus_DM_t-11	6Inphase
	2Cmo	1Wma10_t-9	**2Stok**
	1Midprice10	6Sine_t-7	6Quadature

Table 16
Global interpretability comparison. (Bold font indicates the same features slected by both techniques)

Interpretability Technique	XGBoost	SHAP
Learning Model	**XGBoost** (**From** Fig. 8)	**XGBoost**
Important Features	**2Stok**	**2Stok**
	2Stod	**2Stod**
	2Bop	**2Bop**
	2Slowd	**2Slowk**
	2Slowk	**2Slowd**
	6Inphase	3Natr
	2Stochfh	**2Mfi**
	4Dosc	2Willr
	3Trange	**2Stochfh**
	2Mfi	**4Dosc**
	2Macdhisexhist	**6Inphase**
	Volume	2Ultosc
	2Adxr	6Quadrature
	6Ht_DcPeriod	**3Trange**
	2Macdhist	6Sine
	2Minus_DI	**6Ht_DcPeriod**
	6HT_DcPhase	2Plus_DM
	6Leadsine	1Tema10
	2Macdfix	**2Macdhisexhist**
	2Minus_DM	**Volume**

Appendix C

Abbreviations

ACC	Accuracy
AI	Artificial intelligence
ANN	Artificial neural network
ARCH	Autoregressive conditional heteroscedasticity
ARIMA	Autoregressive integrated moving average
AUC	Area under the ROC curve
BAC	Bank of America corporation
CNN	Convolutional neural network
CORR	Pearson correlation coefficient
DA	Direction accuracy
DNN	Deep neural network
EC	Ensemble classifier
FE	Feature extension
FW	Feature weighted
GA	Genetic algorithm
GARCH	Generalized autoregressive conditional heteroscedasticity
INFY	Infosys limited
KNN	K-nearest neighbour
KOSDAQ	Korean securities dealers automated quotations
KOSPI	Korea composite stock price index 200
LSTM	Long short-term memory
MAE	Mean error
MAPE	Mean absolute percent error
MI	Mutual information
MLP	Multilayer perceptron
MSE	Mean squared error
NB	Naïve Bayes classifier
NN	Neural network
PCA	Principal component analysis
PSO	Particle swarm optimization
Relu	Rectified linear activation function
RF	Random forest
RFE	Recursive feature elimination
RMSE	Root mean squared error
rRMSE	Relative root mean square
RNN	Recurrent neural network
SSE	Shanghai stock exchange
SVM	Support vector machine
SZSE	Shenzhen stock exchange
XGB	Extreme Gradient Boosting

References

Ampomah, E. K., Qin, Z., & Nyame, G. (2020). Evaluation of tree-based ensemble machine learning models in predicting stock price direction of movement. *Information, 11*(6), 332.

Ballings, M., Van den Poel, D., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications, 42* (20), 7046–7056.

Basak, S., Kar, S., Saha, S., Khaidem, L., & Dey, S. R. (2019). Predicting the direction of stock market prices using tree-based classifiers. *The North American Journal of Economics and Finance, 47*, 552–567.

Boxer, H. (2014). *Profitable day and swing trading: using price/volume surges and pattern recognition to catch big moves in the stock market*. John Wiley & Sons Incorporated.

Chakraborty, S., Tomsett, R., Raghavendra, R., Harborne, D., Alzantot, M., Cerutti, F., … Gurram, P. (2017). Interpretability of deep learning models: a survey of results. In *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation* (pp. 1–6). IEEE.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794).

Chen, Y., & Hao, Y. (2017). A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications, 80*, 340–355.

Chollet, F. (2018). *Deep learning with Python*. Manning Publications Co.

Chung, H., & Shin, K. (2018). Genetic algorithm-optimized long short-term memory network for stock market prediction. *Sustainability, 10*(10), 3765.

Chung, H., & Shin, K.-S. (2020). Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction. *Neural Computing and Applications, 32*(12), 7897–7914.

De Stefano, C., Fontanella, F., Marrocco, C., & Scotto di Freca, A. (2014). A GA-based feature selection approach with an application to handwritten character recognition. *Pattern Recognition Letters, 35*, 130–141.

Devan, P., & Khare, N. (2020). An efficient XGBoost–DNN-based classification model for network intrusion detection system. *Neural Computing and Applications, 32*(16), 12499–12514.

Ding, G., & Qin, L. (2020). Study on the prediction of stock price based on the associated network model of LSTM. *International Journal of Machine Learning and Cybernetics, 11* (6), 1307–1317.

Dong, G., & Liu, H. (2018). *Feature engineering for machine learning and data analytics*. CRC Press.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning* (pp. 1050–1059). PMLR.

Gorban, A. N., & Tyukin, I. Y. (2018). Blessing of dimensionality: Mathematical foundations of the statistical physics of data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 376*(2118), 20170237.

Guo, J., Yang, L., Bie, R., Yu, J., Gao, Y., Shen, Y., & Kos, A. (2019). An XGBoost-based physical fitness evaluation model using advanced feature selection and Bayesian hyper-parameter optimization for wearable running monitoring. *Computer Networks, 151*, 166–180.

Hao, Y., & Gao, Q. (2020). Predicting the trend of stock market index using the hybrid neural network based on multiple time scale feature learning. *Applied Sciences, 10* (11), 3961.

Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2019). Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications, 124*, 226–251.

Huang, Q., Xia, T., Sun, H., Yamada, M., & Chang, Y. (2020). Unsupervised nonlinear feature selection from high-dimensional signed networks. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*(04), 4182–4189.

Jalali, A., Schindler, A., Haslhofer, B., & Rauber, A. (2020). Machine Learning interpretability techniques for outage prediction: A comparative study. No. 1. In *PHM Society European Conference* (p. 10).

Jeyakumar, J. V., Noor, J., Cheng, Y. H., Garcia, L., & Srivastava, M. (2020). How can i explain this to you? An empirical study of deep neural network explanation methods. *Advances in Neural Information Processing Systems*.

Kainen, P. C. (1997). Utilizing Geometric anomalies of high dimension: when complexity makes computation easier. In M. Kárný, & K. Warwick (Eds.), *Computer Intensive Methods in Control and Signal Processing: The Curse of Dimensionality* (pp. 283–294). Birkhäuser.

Koziel, S., Leifsson, L., & Yang, X.-S. (Eds.). (2014). *Solving computationally expensive engineering problems: methods and applications*. Springer International Publishing.

Li, A. W., & Bastos, G. S. (2020). Stock market forecasting using deep learning and technical analysis: A systematic review. *IEEE Access, 8*, 185232–185242.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2018). Feature selection: A data perspective. *ACM Computing Surveys, 50*(6), 1–45.

Lin, X. (2011). Intelligent stock trading system based on improved technical analysis and Echo State Network. *Expert Systems with Applications, 38*(9), 11347–11354.

Long, J., Chen, Z., He, W., Wu, T., & Ren, J. (2020). An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market. *Applied Soft Computing, 91*, 106205. https://doi.org/10.1016/j.asoc.2020.106205

Lundberg, S. M., & Lee, S. (2016). A unified approach to interpreting model predictions. In Conference on Neural Information Processing Systems (NIPS 2017).

Maji, P., & Mullins, R. (2018). On the reduction of computational complexity of deep convolutional neural networks. *Entropy, 20*(4), 305.

Makridakis, S., Spiliotis, E., Assimakopoulos, V., & Hernandez Montoya, A. R. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE, 13*(3). e0194889.

Miao, J., & Niu, L. (2016). A survey on feature selection. *Procedia Computer Science, 91*, 919–926.

Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing, 73*, 1–15.

Murphy, J. J. (1999). Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications. Penguin.

Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E., & Shahab, S. (2020a). Deep Learning for Stock Market Prediction. *Entropy, 22*(8), 1aq–1aq.

Nabipour, M., Nayyeri, P., Jabani, H., Shahab, S., & Mosavi, A. (2020b). Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; A comparative analysis. *IEEE Access, 8*, 150199–150212.

Naik, N., & Mohan, B. R. (2019). Stock Price movements classification using machine and deep learning techniques-The case study of indian stock market. In J. Macintyre, L. Iliadis, I. Maglogiannis, & C. Jayne (Eds.), *Engineering Applications of Neural Networks* (pp. 445–452). Springer International Publishing.

Nobre, J., & Neves, R. F. (2019). Combining principal component analysis, discrete wavelet transform and XGBoost to trade in the financial markets. *Expert Systems with Applications, 125*, 181–194.

Ntakaris, A., Mirone, G., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2019). Feature engineering for mid-price prediction with deep learning. *IEEE Access, 7*, 82390–82412.

Nti, I. K., Adekoya, A. F., & Weyori, B. A. (2020). A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review, 53*(4), 3007–3057.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. *Expert Systems with Applications, 42*(1), 259–268.

Peter, S., Diego, F., Hamprecht, F. A., & Nadler, B. (2017). Cost efficient gradient boosting. *Advances in Neural Information Processing Systems, 30*, 1551–1561.

Qiu, M., & Song, Y. (2016). Predicting the direction of stock market index movement using an optimized artificial neural network model. *PLoS ONE, 11*(5). e0155133.

Ramírez-Gallego, S., Lastra, I., Martínez-Rego, D., Bolón-Canedo, V., Benítez, J. M., Herrera, F., & Alonso-Betanzos, A. (2017). Fast-mRMR: Fast minimum redundancy maximum relevance algorithm for high-dimensional big data: FAST-mRMR algorithm for big data. *International Journal of Intelligent Systems, 32*(2), 134–152.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135–1144).

Shah, D., Isah, H., & Zulkernine, F. (2019). Stock market analysis: A review and taxonomy of prediction techniques. *International Journal of Financial Studies, 7*(2), 26.

Shen, J., & Shafiq, M. O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. *Journal of big Data, 7*(1), 1–33.

Shi, L., Teng, Z., Wang, L., Zhang, Y., & Binder, A. (2018). DeepClue: Visual interpretation of text-based deep stock prediction. *IEEE Transactions on Knowledge and Data Engineering, 31*(6), 1094–1108.

Song, Y., Lee, J. W., & Lee, J. (2019). A study on novel filtering and relationship between input-features and target-vectors in a deep learning model for stock price prediction. *Applied Intelligence, 49*(3), 897–911.

Strader, T. J., Rozycki, J. J., & Root, T. H. (2017). Machine learning stock market prediction studies: Review and research directions. *Journal of International Technology and Information Management, 28*(4), 22.

TA-Lib (2021). Mrjbq7/ta-lib [Python]. <https://github.com/mrjbq7/ta-lib>.

Vanstone, B., & Hahn, T. (2010). *Designing stock market trading systems: with and without soft computing*. Harriman House Publishing.

Wang, Q., Xu, W., & Zheng, H. (2018). Combining the wisdom of crowds and technical analysis for financial market prediction using deep random subspace ensembles. *Neurocomputing, 299*, 51–61.

Yahoo Finance—Stock Market Live, Quotes, Business & Finance News. (n.d.). Retrieved February 22, 2021, from <https://finance.yahoo.com/>.

Yang, C., Zhai, J., & Tao, G. (2020). Deep Learning for price movement prediction using convolutional neural network and long short-term memory. *Mathematical Problems in Engineering, 2020*.

Yu, L., & Liu, H. (2003). *Feature Selection for High-Dimensional Data. A Fast Correlation-Based Filter Solution*.

Yu, P., & Yan, X. (2020). Stock price prediction based on deep neural networks. *Neural Computing and Applications, 32*(6), 1609–1628.

Zhang, Y., Apley, D. W., & Chen, W. (2020). Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific reports, 10*(1), 1–13.