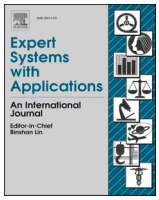
Expert Systems With Applications 257 (2024) 125043

Contents lists available at ScienceDirect

Expert Systems With Applications

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



A novel trading system for the stock market using Deep Q-Network action and instance selection

Myeongseok Park [a], Jaeyun Kim [b,*], David Enke [c]

[a] Department of Future Convergence Technology, Soonchunhyang University, 22 Soonchunhyang-ro, Asan-si, Chungcheongnam-do 31538, Republic of Korea
[b] Department of AI and Big Data, Soonchunhyang University, 22 Soonchunhyang-ro, Asan-si, Chungcheongnam-do 31538, Republic of Korea
[c] Laboratory for Investment and Financial Engineering, Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, 221 Engineering Management, 600 W. 14th Street, Rolla, MO 65409-0370, USA

ARTICLE INFO

Keywords:
Deep Q-Network
DQN Action Instance Selection trading system
Reinforcement learning
Supervised learning

ABSTRACT

Stock trading is a complex decision-making process that involves predicting market price movements. Many investors attempt to buy at low prices and sell at high prices, which can be difficult due to numerous factors that can influence stock prices. Consequently, researchers have developed trading systems designed to generate high stock market returns using machine learning or deep learning techniques. However, building a trading system using supervised learning can be challenging in real-time due to noisy stock price data. Reinforcement learning trading systems can play an active role in the stock market and their performance depends heavily on how the learning environment is constructed. Therefore, this study proposes a Deep Q-Network (DQN) Action Instance Selection Trading System (DAIS) to improve the limitations of both supervised learning and reinforcement learning trading systems. DAIS learns effective trading timing by labeling the behavior of DQN models and using instance selection to effectively handle noise in stock price data. Our proposed model is evaluated by testing 72 companies listed on Korean Composite Stock Price Indexes (KOSPI) and comparing the results with those obtained in previous studies using supervised and reinforcement learning trading systems. The results indicate that the DAIS trading system is more efficient than the compared systems.

1. Introduction

Stock trading refers to the exchange of stocks to earn investment profits. Making informed decisions about when to buy and sell stocks is one of the most important aspects of stock trading. However, predicting the stock market and making informed trading decisions can be challenging due to the diverse external factors that influence them. Fundamental and technical analyses are the two main methods used to predict stock markets and develop effective trading strategies. *Fundamental analysis* involves analyzing a company's intrinsic value and other elements to forecast future stock prices (Abarbanell & Bushee, 1997). By contrast, *technical analysis* utilizes past stock prices and trading volumes to identify price patterns or predict future stock prices (Murphy, 1999). Investors attempt to forecast future stock prices using these analytical methods; however, many stock market patterns are difficult to detect in their entirety. Consequently, machine learning and deep learning techniques have been employed in recent research to predict stock prices and fluctuations.

Chen et al. (2020) developed a financial trading system to predict stock prices using a Light Gradient Boosting Machine (LGBM). They compared the accuracy of their system with that of a Generalized Linear Model (GLM), Deep Neural Network (DNN), Random Forest (RF), or Support Vector Machine (SVM). The results showed that LGBM had the highest accuracy among the tested models. Dash and Dash (2016) proposed a trading system that combined a computationally efficient functional link Artificial Neural Network (CEFLANN) with a technical analysis for efficient trading decisions. They converted the output of CEFLANN into trading signals and conducted trading simulations. They compared the performance of their proposed trading system to that of a Support Vector Machine (SVM), K-nearest neighbor (KNN), and Decision Tree (DT) and found that their proposed system had a higher return rate than the compared models.

There are limitations to using machine-learning techniques to predict stock prices and trade in the stock market. Most of these methods are based on supervised learning, which makes it difficult to adapt to the real-time nature of the stock market. In these systems, training examples

* Corresponding author.
E-mail addresses: pmsk980122@sch.ac.kr (M. Park), kimym38@sch.ac.kr (J. Kim), enke@mst.edu (D. Enke).

https://doi.org/10.1016/j.eswa.2024.125043
Received 13 April 2023; Received in revised form 14 August 2023; Accepted 6 August 2024
Available online 8 August 2024

Table 1
Summary of related works.

Authors	Dataset	Prediction models	Dimension reduction	Labeling method
Lee (2009)	NASDAQ	SVM	Feature selection	Up–down
Zhong and Enke (2017)	S&P 500 ETF	ANN	Featureextraction	Up–down
Lv et al. (2019)	S&P 500 & CSI 300	ML models (LR, SVM, CART, RF, BN, and XGB) andDNN models (MLP, DBN, SAE, RNN, LSTM, and GRU)	N/A	Up–down
Pun and Wong (2019)	S&P 500 & Russell 2000	Glasso	Feature extraction	N/A
Xu et al. (2020)	Gree ElectricAppliances & ZTE corporation	CNN, Bi-LSTM	N/A	Up–down
Khan et al. (2020)	S&P 500	N/A	Feature selection &extraction	Up–down
Yang et al. (2020)	DJIA	PPO, A2C, DDPG	Feature selection	N/A
Carter et al. (2021)	S&P 500 &DAX	Multi-DQN	Feature extraction	N/A
Théate and Ernst (2021)	DIA & S&P 500	TDQN	N/A	N/A
Hirchoua et al. (2021)	BTCUSDT & S&P 500	DQN	N/A	N/A
Current Study	KOSPI 200	DT, GBM, KNN, LR, MLP, RF, XGB, DQN	Instance selection	DQN Action

are often based solely on price differences, which can be noisy and fail to consider the nonlinear and complex nature of stock prices (Han et al., 2023). Reinforcement learning has been used to develop trading systems to overcome these limitations.

Reinforcement learning is a form of machine learning where an agent learns by engaging in trial and error and receiving rewards or penalties for actions taken (Minh et al., 2013). It effectively solves sequential decision-making problems and has been applied in various fields, including gaming, autonomous driving, and robotics (Mnih et al., 2015). Stock trading can also be viewed as a sequential decision-making problem. By applying reinforcement learning to stock trading, flexible strategies can be developed to adapt to changing market conditions. However, the learning environment must be carefully designed (Li et al, 2022). Several studies use only stock price data to study reinforcement learning in stock trading (Bao et al., 2017; Karaoglu et al., 2017). The addition of technical indicators and stock price data may reduce noise and better reflect market changes (Gorgulho et al., 2011). However, technical indicators are not leading stock market indicators; thus, their generalizability is limited. Therefore, this study proposes a trading system that overcomes the limitations of supervised and reinforcement learning trading systems. Using technical indicators, the proposed trading system builds a reinforcement learning environment and trains a Deep Q-Network (DQN) algorithm. The DQN learns the flow of the stock market and extracts buy/sell signals by applying an instance selection technique to reconstruct the learning data. The reconstructed learning data are used to train a supervised machine learning algorithm and build a trading system. The results of the built trading system are compared with those of existing supervised learning and reinforcement learning trading systems.

This study makes the following key contributions:

(1) We propose a trading system that employs reinforcement learning to build a trading system, which is different from previous studies that used reinforcement learning to determine optimal trading times. Stock trading involves sequential decision-making, which depends on time flow; reinforcement learning can effectively solve this problem.
(2) We use an instance selection technique to overcome the limitations of supervised learning trading systems. A limitation of supervised learning trading systems is that they do not consider noise in stock data when learning. By using the instance selection technique, we can reduce the impact of noise and improve learning efficiency.
(3) By comparing the performance of our proposed DQN instance selection labeling method with that of existing supervised learning and reinforcement learning trading systems, our process proved to be suitable for trading system design. These three contributions demonstrate that providing optimal trading timing through labeling improves the trading system.

The remainder of this paper is organized as follows. Section 2 describes related work on supervised and reinforcement learning trading systems. Section 3 delineates the research process. Section 4 presents the experimental results and discussion. Finally, Section 5 concludes the paper and outlines the scope of future research.

2. Related works

Recent studies have shown that machine- or deep-learning techniques in the stock market can be highly effective. Algorithmic trading systems that use these techniques can identify patterns in the stock market and find profitable trading methods, leading to improved results. This section introduces recent research on machine- and reinforcement-learning-based trading systems. These studies demonstrate the potential of advanced technologies for stock trading. Table 1 is a summary table that presents the applied techniques and labeling availability of the related studies.

2.1. Machine learning trading system

Most machine-learning trading system studies use supervised learning to predict fluctuations in stock prices or the prices of stocks themselves. Lee (2009) proposed an SVM-based prediction model using a hybrid feature selection method to predict trends in the stock market. It combined the advantages of wrapper and filter methods to extract optimal features. To verify the performance of the proposed model, it was compared with that of a back-propagation neural network. The results revealed that the SVM-based prediction model using the proposed hybrid feature selection method exhibited high accuracy and generalization performance.

Lv et al. (2019) built a machine learning trading system using technical indicators. Technical indicators were used to construct training data and predict stock trends as a trading signal. The predicted trading signal was used to formulate a trading strategy and conduct a backtest. In addition, the algorithm's performance was analyzed by considering cases with and without trading costs. The results showed that the machine learning algorithms outperformed the DNN algorithm. However, in the case of trading costs, the DNN algorithm was superior to machine-learning algorithms, thus indicating that the DNN algorithm had strong resistance to trading costs.

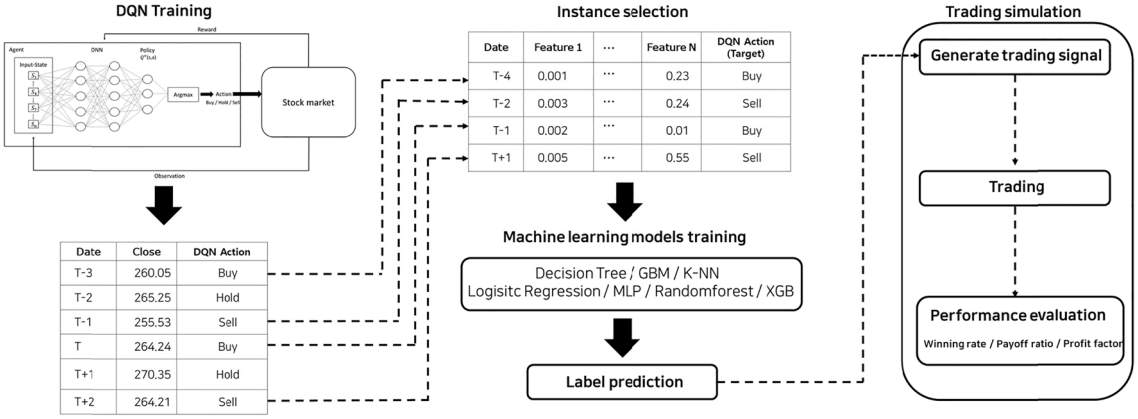


Fig. 1. Proposed trading system framework.

Xu et al. (2020) investigated the effect of online financial news on stock price movements by constructing an LSTM model that incorporates stock price data, news event characteristics, and emotional orientations. To evaluate the performance of their model, experiments were conducted on two individual stocks from different industries. According to the results, using financial online news data enhanced prediction accuracy compared to only using historical stock price data.

Hossain et al. (2022) suggested a BRBES (Belief Rule-Based Expert System). This model uses a machine-learning technique to predict stock prices over the next five days using Bollinger bands, a technical indicator. The performance evaluation metrics used were 1) accuracy, 2) Area Under ROC Curve, 3) Root Mean Squared Error, 4) Type I Error, 5) Type II Error, 6) R2 Value, and 7) fit/loss ratio. To compare the performance of the proposed model, an SVM, an Adaptive Neuro-Fuzzy Inference System (ANFIS), and a heuristic approach were used. The results indicated that the BRBES exhibited superior performance in all performance evaluation metrics. Most supervised learning trading system studies use up–down labeling to predict stock trends. The up-down labeling method is described in detail in section 2.1.1.

2.1.1. *Up/down labeling*

Up–down labeling is frequently used in the development of supervised learning trading systems. It involves labeling the difference between the stock price at time T and time T + 1 as "up" if the value is positive and "down" if the value is negative (Han et al., 2023). In this study, up-down labeling was performed using the closing price for comparison with the proposed method. Up–down labeling can be defined as

$$\text{Up} = \text{Close price}_{t+1} > \text{Close price}_{t} \quad (1)$$

$$\text{Down} = \text{Close price}_{t+1} < \text{Close price}_{t} \quad (2)$$

2.2. *Reinforcement learning trading system*

A limitation of supervised learning trading systems is that they require a large amount of labeled data for effective training. Obtaining this data can be difficult, especially for less liquid or widely traded assets. In addition, supervised learning systems may be prone to overfitting, particularly if they do not include sufficient training data representative of market conditions. Finally, supervised learning systems may struggle to adapt to dynamic market conditions as they rely on past data and cannot incorporate new information in real-time. To overcome these limitations, there has been an increase in research on utilizing reinforcement learning to build trading systems.

Carter et al. (2021) suggested using an ensemble reinforcement learning approach that learns to maximize the return function in the training phase without considering the rise or fall of the stock market. The proposed method used multiple Q-learning agents trained with the same data and executed an ensemble operation in the real stock market. A buy-and-hold strategy was compared to this method to evaluate its performance. Consequently, it was confirmed that a multi-DQN method was superior to the buy-and-hold strategy.

Théate and Ernst (2021) present a Trading Deep Q-Network (TDQN) to determine the optimal trading position in the stock market at any given time. This model identified strategies for trading policies to maximize the Sharpe ratio performance indicator. To assess the performance of their model, they compared it to benchmark strategies such as Buy and Hold, Sell and Hold, Trend following with moving averages (TF), and mean reversion with moving averages (MR). The results revealed that the TDQN exhibited superior performance compared to the benchmark strategies. Additionally, the TDQN proved to be a robust model, even when trading costs were considered.

Hirchoua et al. (2021) proposed an innovative rule-based policy approach to training deep reinforcement learning agents for automated financial trading. This approach modulated reinforcement learning action selection by using the concept of proximal policy optimization, thereby improving agent decision-making. Additionally, they developed algorithms with reinforcement learning that maximize risk aversion. Experiments on eight real stocks validated this approach's effectiveness, thus yielding lower transaction costs and excess market returns.

Yang et al. (2020) devised an ensemble strategy that used deep reinforcement learning to maximize investment returns and learn trading strategies. They used three actor-critic-based algorithms to train their deep reinforcement learning agent: Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), and Deep Deterministic Policy Gradient (DDPG). This trading strategy was then tested on the 30 individual Dow Jones Industrial Average stocks. Performance was evaluated by comparing with the Dow Jones Industrial Average (DJIA index) and the traditional minimum variance portfolio allocation strategy. The results indicated that the strategy outperformed the benchmarks in terms of Sharpe ratio.

3. Proposed system

The proposed trading system consists of three steps. First, we employ the DQN algorithm to gather information about the stock market and extract actions. Second, the learning data and machine learning algorithms were constructed using instance selection. Third, we use a machine learning algorithm to predict and generate a trading signal. Finally, we conduct a simulation using the generated trading signal and evaluate its performance. Fig. 1 illustrates the trading system framework.

Table 2
Technical indicators used as input-state.

Indicator	Description	Parameters
ADX	Average Directional Movement Index	timeperiod = 14
ADXR	Average Directional Movement Index Rating	timeperiod = 14
APO	Absolute Price Oscillator	fastperiod = 12 slowperiod = 26 matype = 0
AROON	Aroon	timeperiod = 14
AROONOSC	Aroon Oscillator	timeperiod = 14
BOP	Balance Of Power	N/A
CCI	Commodity Channel Index	timeperiod = 14
CMO	Chande Momentum Oscillator	timeperiod = 14
DX	Directional Movement Index	timeperiod = 14
MACD	Moving Average Convergence/Divergence	fastperiod = 12 slowperiod = 26 signalperiod = 9
MFI	Money Flow Index	timeperiod = 14
MINUS_DI	Minus Directional Indicator	timeperiod = 14
MINUS_DM	Minus Directional Movement	timeperiod = 14
MOM	Momentum	timeperiod = 14
PLUS_DI	Plus Directional Indicator	timeperiod = 14
PPO	Percentage Price Oscillator	timeperiod = 14
ROC	Rate of Change	timeperiod = 10
RSI	Relative Strength Index	timeperiod = 14
STOCHF	Stochastic Fast	fastkperiod = 5 fastdperiod = 3 fastd_matype = 0
TRIX	1-day Rate-Of-Change (ROC) of a Triple Smooth EMA	timeperiod = 30
ULTOSC	Ultimate Oscillator	timeperiod1 = 7 timeperiod2 = 14 timeperiod3 = 28
WILLR	Williams' %R	timeperiod = 14

3.1. Step 1: DQN

3.1.1. Q-learning

Q-learning-based reinforcement learning is an algorithm based on Markov Decision Process (MDP) theory (Mnih et al., 2013). It learns by maximizing rewards through state-value and action-value functions. The state value function in state s is defined as the expected value of the reward obtained in the future.

$$V_\pi(s) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \mid S_t = s] \quad (3)$$

Eq. (3) refers to a discount factor with a value between 0 and 1, which determines how the future situation affects current learning. The action-value function adds an action to the state-value function. It is defined as the expected value of the reward obtained when an action is performed in a given state.

$$Q_\pi(s,a) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \mid S_t = s,\ A_t = a] \quad (4)$$

Eq. (4) is defined in Eq. (5) according to the Bellman equation. In a given state, this represents the action that maximizes its value. Thus, Q-learning is performed by approximating the action-value function in Eq. (5).

$$Q(s,a) = r + \gamma \max_{a'} Q(s', a') \quad (5)$$

3.1.2. DQN

In a DQN, the loss function is the difference between the Q-value predicted by the neural network and the target Q-value. This loss function is used to update the weights of the neural network and improve the accuracy of Q-value prediction. The target Q-value is the reward for the current action plus the maximum predicted Q-value for

Table 3
Example of trading strategy.

Date	Prediction	Signal
T	Buy	Buy
T + 1	Sell	Sell
T + 2	Sell	No action
T + 3	Buy	Buy
T + 4	Buy	Hold
T + 5	Sell	Sell
T + 6	Sell	No action

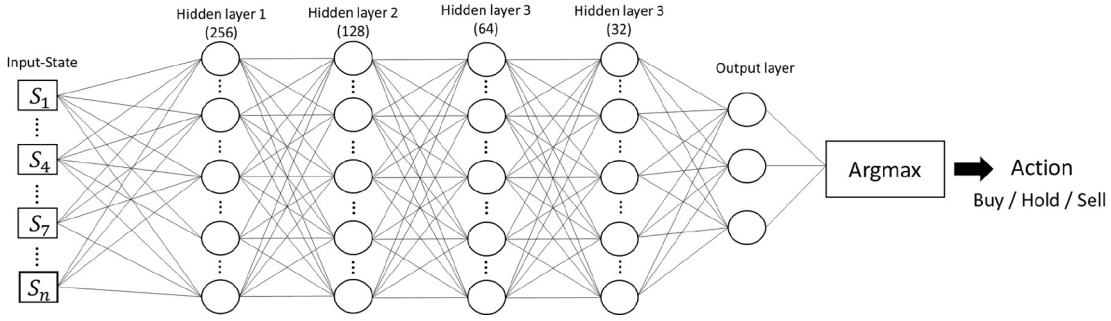


Fig. 2. DNN architecture.

Date	Close	DQN Action
T-3	260.05	Buy
T-2	265.25	Hold
T-1	255.53	Sell
T	264.24	Buy
T+1	270.35	Hold
T+2	264.21	Sell

Instance selection

Date	Feature 1	…	Feature N	DQN Action (Target)
T-4	0.001	…	0.23	Buy
T-2	0.003	…	0.24	Sell
T-1	0.002	…	0.01	Buy
T+1	0.005	…	0.55	Sell

Fig. 3. The process of instance selection.

Table 4
List of KOSPI stocks used in the experiment.

Stock	Ticker	Stock	Ticker
Yuhan Corporation	000100	S-1	012750
CJ Logistics	000120	Korea Electric Power	015760
KIA	000270	Samsung Securities	016360
SK hynix	000660	SK Telecom	017670
Hyundai E&C	000720	Hanon Systems	018880
Samsung Fire & Marine Insurance	000810	Shin Poong Pharmaceutical	019170
CJ	001040	Coway	021240
Amore Group	002790	Lotte shopping	023530
Ssangyong Cement Industrial Co., Ltd	003410	Industrial Bank	024110
Korean Air	003490	Samsung Engineering	028050
LG	003550	Samsung Card	029780
Posco Chemical Co Ltd	003670	CheilWorldwide	030000
Hyundai Steel	004020	KT	030200
LOTTE Corporation	004990	LG U+	032640
Hyundai Motor	005380	KT&G	033780
POSCO Holdings	005490	Doosan Enerbility	034020
DB Insurance	005830	LG Display	034220
Samsung Electronics	005930	Kangwon Land	035250
NH INVESTMENT & SECURITIES	005940	NAVER	035420
GS Engineering & Construction	006360	Kakao	035720
Samsung SDI	006400	Korea Gas Corporation	036460
Mirae Asset Securities	006800	NC soft	036570
Hotel Shilla	008770	Daewoo Shipbuilding & Marine Engineering	042660
Hanmi Science	008930	LG Household & Healthcare	051900
Samsung Electro-Mechanics	009150	LG Chem Ltd.	051910
Korea Shipbuilding & Offshore Engineering	009540	Shinhan Financial Group	055550
Hanwha Solutions	009830	LG electronics	066570
Korea Zinc Company	010130	Celltrion	068270
Samsung Heavy Industries	010140	Korea Investment Holdings	071050
S-Oil	010950	GS	078930
LG Innotek	011070	HYUNDAI GLOVIS	086280
Lotte Chemical	011170	Hana Financial Group	086790
HMM	011200	AmorePacific	090430
KUMHO PETROCHEMICAL	011780	SK INNOVATION	096770
SKC	011790	CJ CheilJedang	097950
Hyundai Mobis	012330	KB Financial Group	105560

the next state discounted by the discount factor (gamma). By minimizing the loss function, DQN learns to choose actions that maximize long-term reward (Mnih et al., 2013).

3.1.3. *DQN input-state*

This study uses technical indicators to build a stock market situation as a state for the agent to learn from reinforcement learning. Using only stock prices to establish complex stock market conditions can be challenging. Therefore, technical indicators have been utilized to provide a more comprehensive understanding of the market. The agent can make informed decisions during the learning process by considering these indicators as part of the state. All of the technical indicators used in this study are momentum indicators. Momentum indicators gauge the potency of a trend and are capable of identifying overbought and oversold conditions in the market, aiding in the identification of favorable buy and sell signals. Unlike trend-following indicators, which fluctuate in response to stock price trends, momentum indicators remain

Table 5
Trading performance by DQN and DAIS trading system.

Model	No. trades	Winning ratio	Payoff ratio	Profit factor	Sharpe ratio
Decision tree	90.30	0.48	1.10 (0.30)	1.11 (0.53)	1.03
GBM	88.27	0.47	1.10 (0.28)	1.10 (0.57)	0.66
knn	78.62	0.49	1.10 (0.40)	1.13 (0.67)	0.92
Logisitic Regression	52.70	0.49	1.12 (0.59)	1.21 (1.93)	0.58
MLP	61.35	0.49	1.12 (0.53)	1.21 (0.89)	0.67
Random forest	76.98	0.49	1.15 (0.34)	1.20 (0.58)	1.09
XGB	89.04	0.49	1.04 (0.28)	1.09 (0.52)	0.93
DQN	84.01	0.46	0.97 (0.45)	0.94 (0.59)	0.38

Note: Values for trading performance are given as average (standard deviation).

Table 6
Trading performance by Up–down labeling.

Up–down labeling					
Model	No. trades	Winning ratio	Payoff ratio	Profit factor	Sharpe ratio
Decision tree	93.81	0.45	0.99 (0.28)	0.84 (0.33)	−0.42
GBM	75.58	0.36	0.83 (0.35)	0.52 (0.35)	−3.36
knn	83.32	0.47	1.02 (0.23)	0.94 (0.35)	0.37
Logisitic Regression	46.15	0.35	0.98 (0.59)	0.67 (0.64)	−3.01
MLP	66.28	0.46	1.09 (0.44)	1.01 (0.62)	0.06
Random forest	72.54	0.37	0.79 (0.24)	0.50 (0.25)	−3.07
XGB	86.85	0.45	0.94 (0.36)	0.82 (0.37)	−2.87

Note: Values for trading performance are given as average (standard deviation).

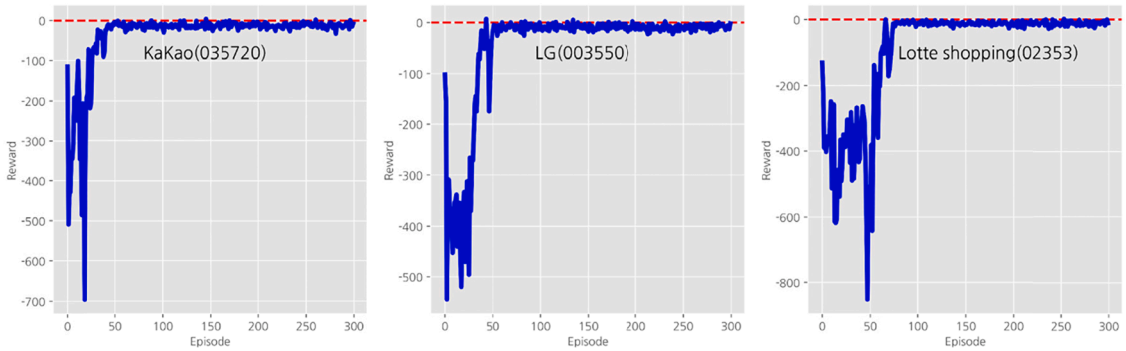


Fig. 4. DQN Cumulative reward.

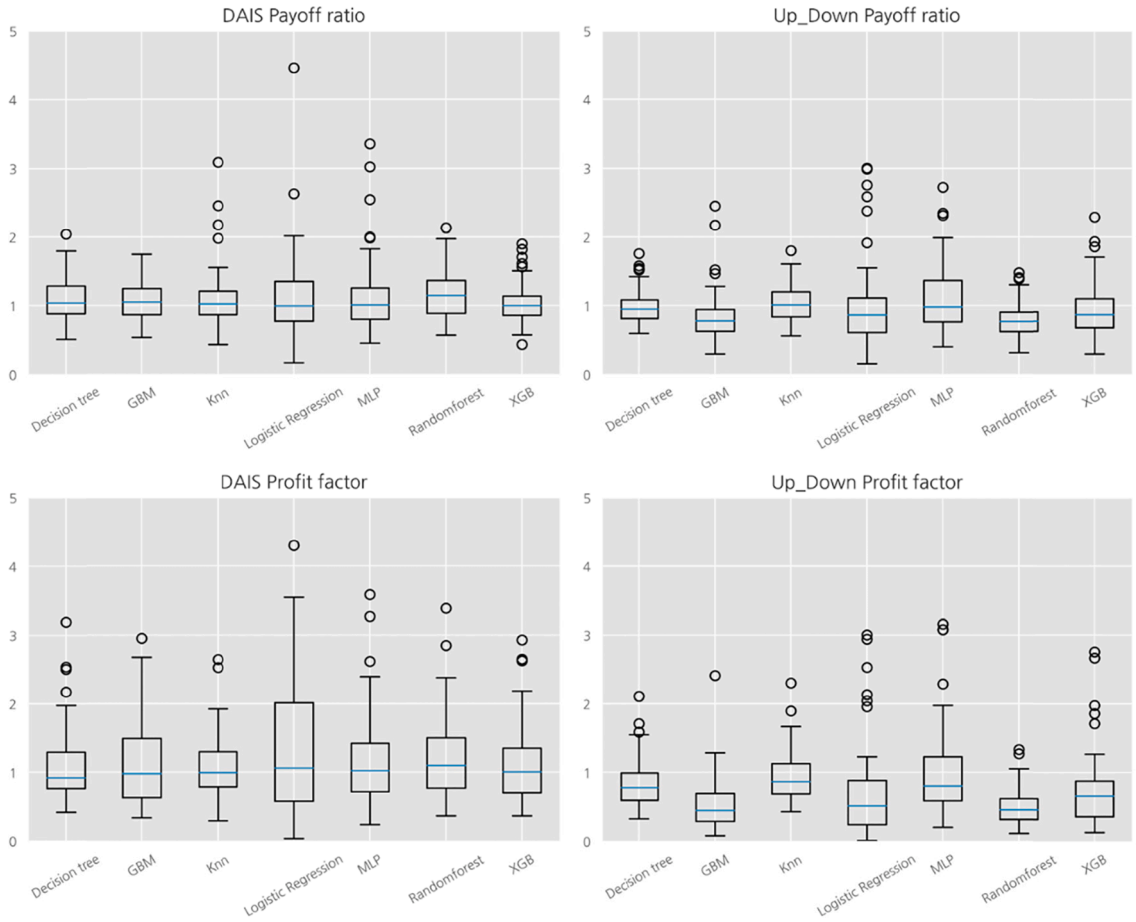


Fig. 5. Boxplot of payoff ratio and profit factor by the different trading systems.

unaffected by the price direction bias, enabling a more comprehensive analysis. Consequently, these price range-bound momentum indicators were evaluated to derive decision-making rules. Table 2 lists the technical indicators used to construct the reinforcement learning environment.

3.1.4. DQN agent

This study uses technical indicators to construct input states for the agent to determine its actions. Fig. 2 illustrates the four hidden neural network layers serving as an agent. The activation function used for the agent was a Rectified Linear Unit (ReLU), and the optimization algorithm used Stochastic Gradient Descent (SGD). The discount rate was set to 0.9, and the number of learning iterations was set to 300 to train the agent. The agent can decide between buying, selling, and holding. Through its neural network, the agent calculates the values of the buy, sell, and hold actions and then chooses the action with the highest value. Reward functions determine the rate of return at the current time of action, which is crucial for reinforcement learning.

3.2. Step 2: Instance selection

Instance selection is a data preprocessing method used to improve supervised learning algorithms. This involves selecting the subset of available data most relevant to the task. Instance selection can improve model accuracy by removing noise or outliers. In addition, reducing the dataset size can make the learning process more efficient because it requires less computation (Malekipirbazari et al., 2021; Sun & Li, 2011). This study uses instance selection to filter the data and enhance the performance of machine learning models.

3.2.1. Building learning data with instance selection

This study uses instance selection to address stock data noise, non-stationarity, and complexity. The learning data are constructed by selecting the buy/sell action of the reinforcement learning as an instance and including the technical indicators from the day before ($t-1$). The instance selection process is illustrated in Fig. 3.

3.2.2. Machine learning training and predictions

In this study, machine learning algorithms are trained using learning data constructed through instance selection. Our experiments employ Decision Trees, Gradient Boosting Machine (GBM), K-nearest neighbor (K-NN), logistic regression, Multilayer Perceptron (MLP), Random Forest (RF), and Extreme Gradient Boosting (XGB) to construct the algorithm.

A *Decision Tree* is a tree-like model used for classification and prediction. It divides the feature space into regions, each represented by a leaf node in a tree. Each internal node represents a decision based on the value of a feature, and the branches of the tree correspond to the possible values of that feature. Recursively partitioning the feature space based on the feature that yields the highest reduction in entropy produces a tree. The test data is passed down the tree when a prediction is made. Decision trees are simple to understand and interpret and are widely used in many applications, including machine learning and data mining (Quinlan, 1986).

Gradient Boosting Machines (GBMs) are a type of supervised machine learning algorithm that can be used for regression and classification tasks. They work by sequentially training weak models, such as decision trees, and combining their predictions to form a stronger model. The key idea behind gradient boosting is to train weak models to maximize the gradient of the loss function. It allows the model to correct mistakes

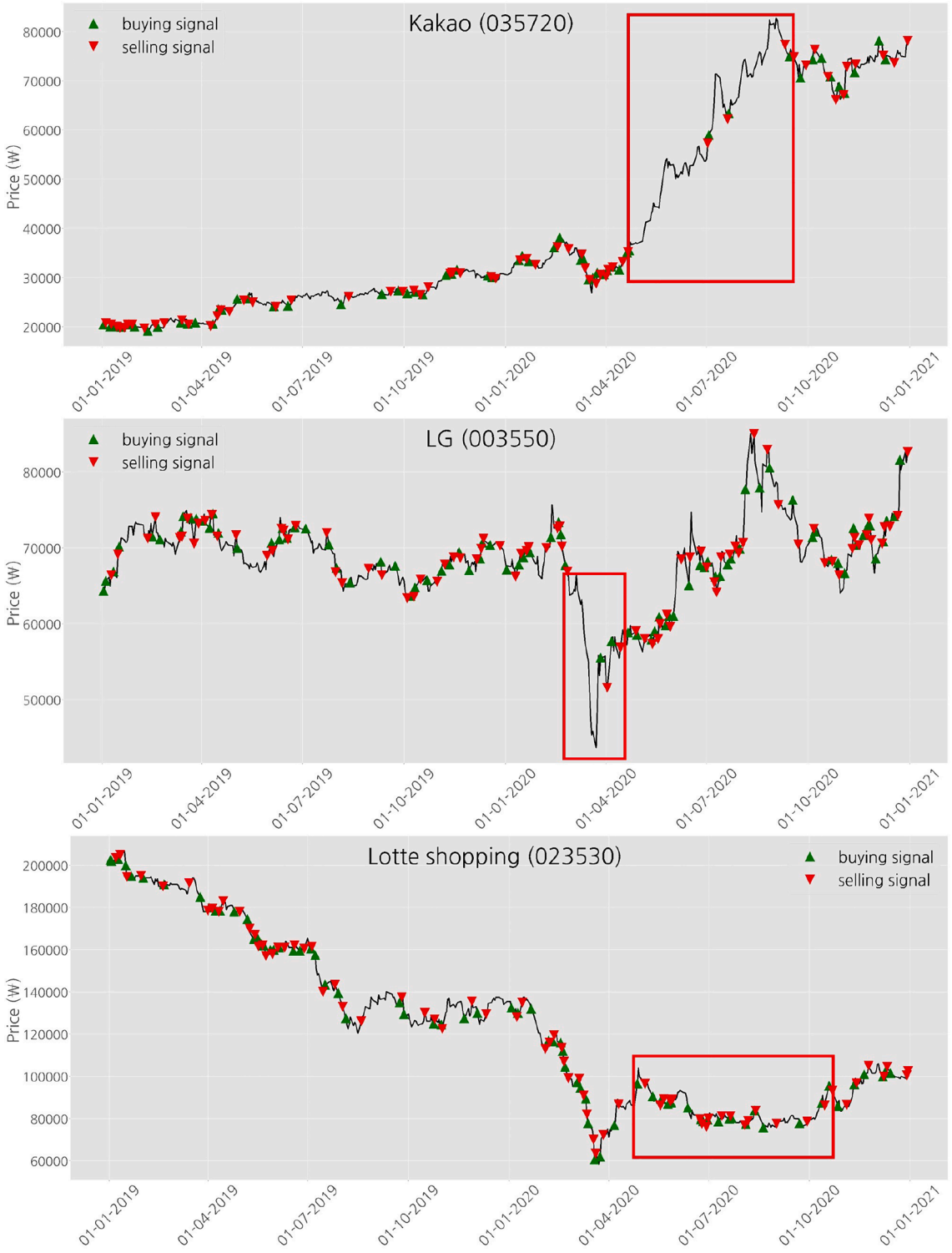


Fig. 6. Trading signal of DAIS trading system (From top to bottom: Uptrend, Sideways, and Downtrend Trend).

made by previous weak models and improve prediction accuracy (Friedman, 2001).

The *k-nearest neighbors* (K-NN) algorithm is a classification algorithm based on the principle of similarity. In the K-NN algorithm, a data point is classified based on the classification of its k-nearest neighbors in the feature space. The k-nearest neighbors are determined by the Euclidean distance between the data point and other points in the feature space. The user typically sets the value of k, and a larger value results in a smoother decision boundary. K-NN is a simple and effective algorithm for classification tasks; however, it can be computationally expensive because it requires storing all the training data and calculating the distances for each test point (Peterson, 2009).

Logistic regression is a statistical method for predicting binary outcomes, such as the probability of an event occurring. It is a supervised learning algorithm that uses input features to predict the likelihood of an event. The algorithm is based on a logistic function that maps the input features to a range between 0 and 1, thus representing the probability of an event occurring. Logistic regression can be trained using various

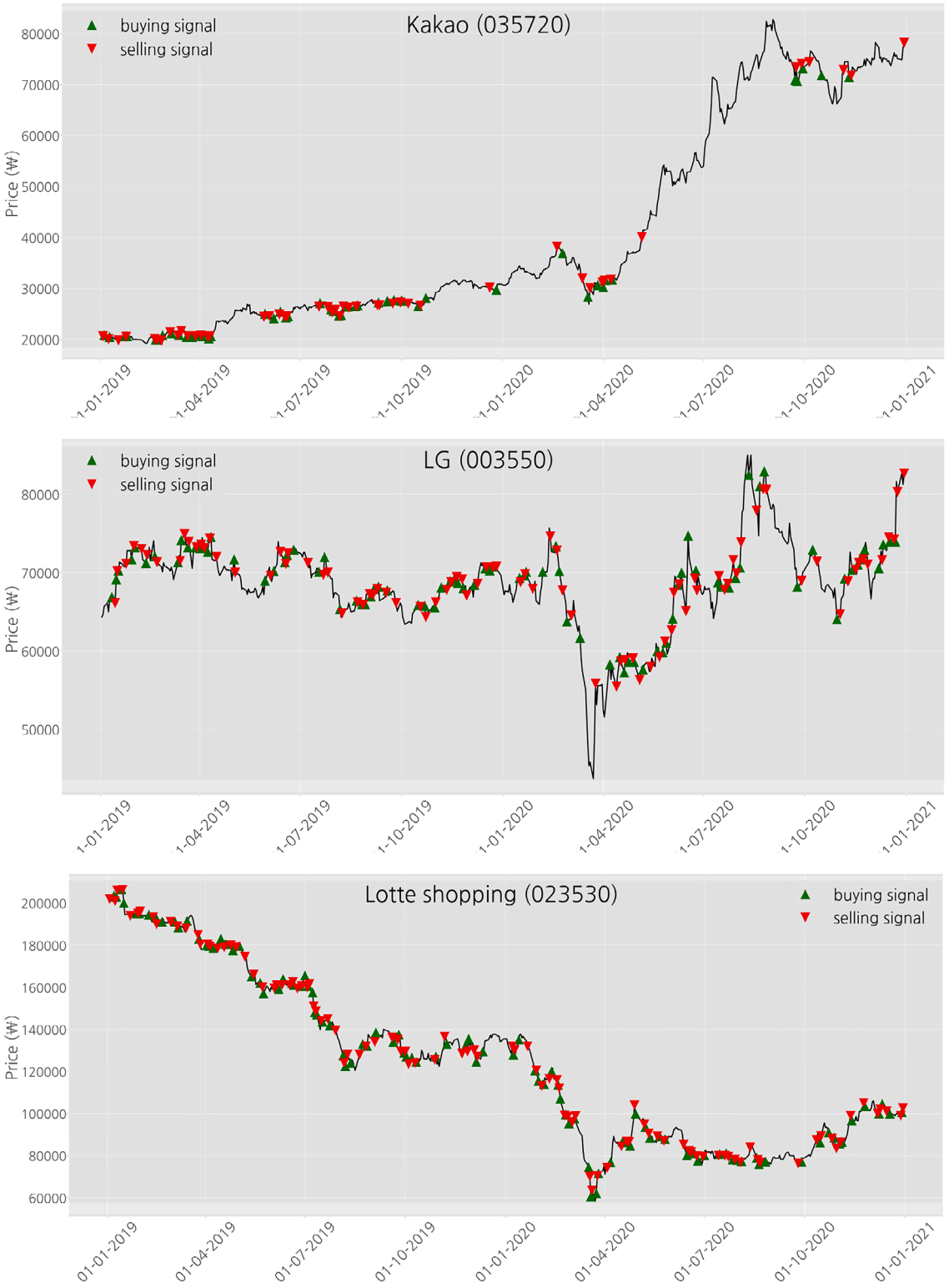


Fig. 7. Trading signal of DQN trading system (From top to bottom: Uptrend, Sideways, and Downtrend Trend).

optimization algorithms, such as gradient descent, and is regularized to prevent overfitting. It has been employed in various applications, including credit risk assessment, medical diagnosis, and marketing research (James et al., 2013).

Multilayer Perceptron (MLP) is a supervised learning algorithm that utilizes a feedforward artificial neural network to classify data. It comprises multiple layers of artificial neurons. Each layer receives input from the previous layer and passes it through a nonlinear activation function before passing it onto the next layer. The final layer of the MLP produces model output that can be used for classification or regression tasks. MLPs are widely used in various fields, including image recognition, natural language processing, and financial forecasting. A main advantage of MLPs is their ability to learn complex relationships between inputs and outputs through training. However, they can also be prone to overfitting if the number of layers or neurons is not carefully selected (Bishop, 1995).

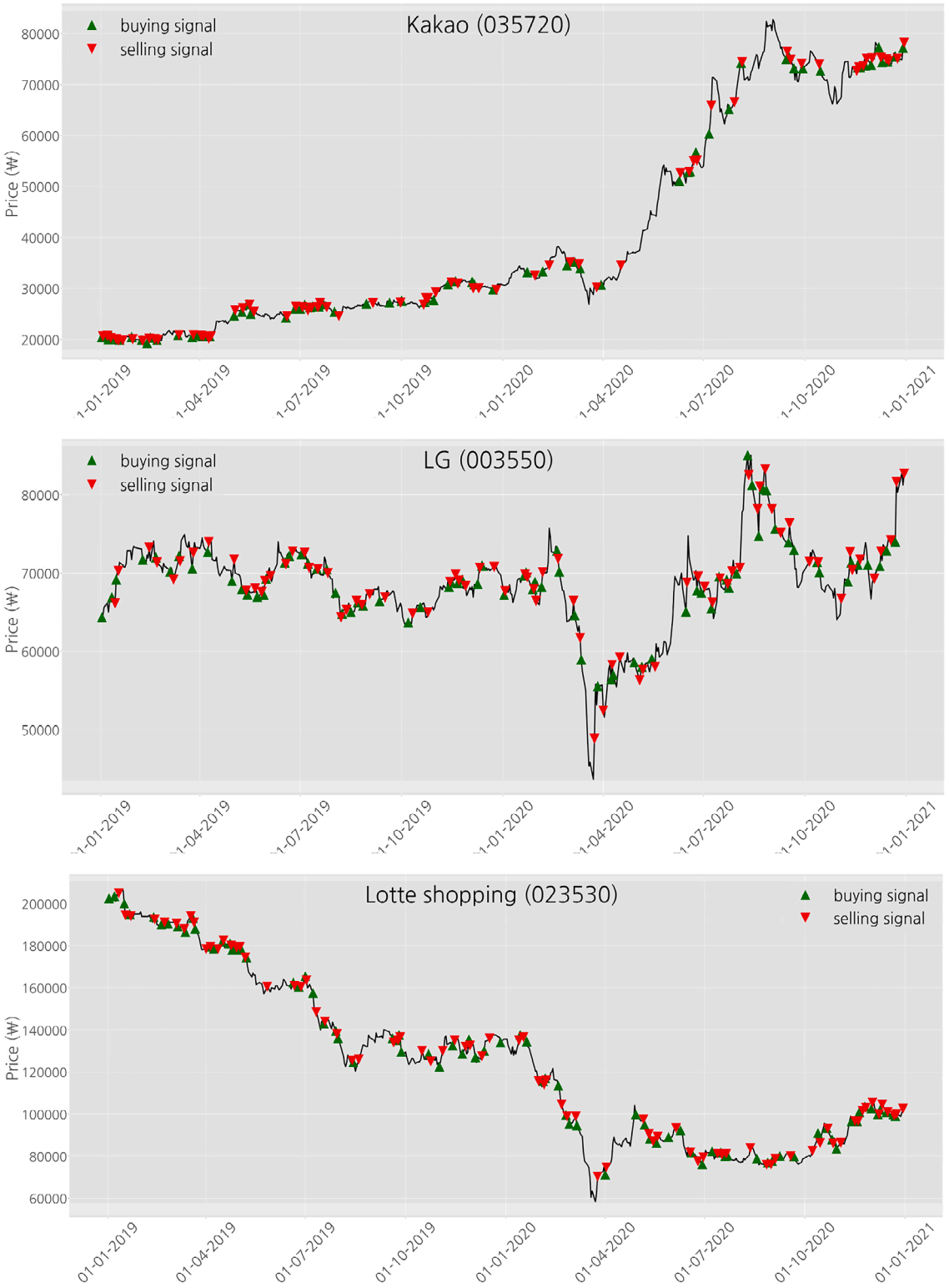


Fig. 8. Trading signal of Up–down trading system (From top to bottom: Uptrend, Sideways, and Downtrend Trend).

Random forest is a machine learning algorithm that creates an ensemble of decision trees from a randomly selected subset of training data. Each decision tree's predictions were aggregated to make predictions. Random forests have been successfully used in various applications, including image classification, spam filtering, and stock price prediction. An advantage of Random forests is that they can handle high-dimensional data and are resistant to overfitting (Breiman, 2001).

Extreme Gradient Boosting (XGB; XGBoost) is a robust and widely used tree-boosting algorithm that has shown impressive results in a variety of machine learning tasks. It is designed to be efficient, flexible, and portable and has been widely adopted in industry and academia owing to its strong performance in various tasks. XGBoost is known for its ability to handle large-scale data and robustness to noise and missing values (Chen & Guestrin, 2016).

After learning each machine learning algorithm, the predictions necessary for the trading strategy are generated using the test data.

3.3. Step 3: Trading simulation

3.3.1. Trading strategies

The trading strategy is designed based on a trained machine-learning model's predicted buy and sell signals. When a buy position is triggered, the strategy holds stocks until the next sell position is signaled. No trade is made when a sell position is detected until the next buy position is signaled. When a buy position is triggered, the stock is bought at the opening price the next day. When a sell position is signaled, the stock is sold at the opening price the following day. To simulate actual trading, a trading fee of 0.015 % is applied. Table 3 presents an example of a trading strategy.

3.3.2. Performance evaluation

We evaluate the performance of the proposed trading system using trading evaluation metrics, which are the number of trades, win rate, payoff ratio, profit factor and Sharpe ratio. The payoff ratio is the average profit ($\overline{W}$) divided by the average loss ($\overline{L}$), while the profit factor is the total profit ($\sum W$) divided by the total loss ($\sum L$). Consequently, the payoff is evaluated in conjunction with the profit factor as it is influenced by the win rate. A profit factor greater than 1 indicates a profit, whereas a value less than 1 indicates a loss. The payoff ratio (P_r) and profit factor (P_f) are calculated using Eq. (6). The Sharpe ratio (S_r) is a widely used performance indicator in the fields of finance and algorithmic trading. It is particularly suitable for evaluating the performance as it considers both the generated profit and the risk associated with the trading activity. In this study, the annualized Sharpe ratio is used to assess the trading performance, and it is computed as shown in Eq. (7); R_t represents the return per trade of the trading strategy over a specific time period, R_f denotes the risk-free rate (assumed to be 0 in this study), and σ_r represents the standard deviation of $R_t - R_f$.

$$P_r = \frac{\overline{W}}{\overline{L}}, P_f = \frac{\sum W}{\sum L} \tag{6}$$

$$S_r = \frac{E[R_t - R_f]}{\sigma_r} \times \sqrt{252} \cong \frac{E[R_t]}{\sigma_r} \times \sqrt{252} \tag{7}$$

4. Experimental results and discussion

The code and dataset employed in this research are made available at https://github.com/pmsk98/DAIS-System/.

4.1. Dataset

The stock data analyzed in this study comprise the top 100 KOSPI companies by market capitalization. The stock data consist of daily open, high, low, and closing prices and volume from 2013 to 2020. Stocks without stock price information (such as those recently listed) from 2013 to 2020 are excluded from the experiment. The training period is set from 2013 to 2018 and the test period is set from 2019 to 2020. This study uses 1,473 stock price data points over a specified period as learning data. When instance selection is conducted, the average learning data is reduced to 438.94. A total of 72 stocks are used in the experiments, as listed in Table 4. The FinanceDataReader package in Python is used to extract stock data.

4.2. Comparison of DQN and DAIS systems

Fig. 4 shows the cumulative reward graph of DQN. Initially, there was unstable reward fluctuation, but as the training progressed, the graph exhibited a gradually increasing and stable trend. This phenomenon provides strong evidence that the agent converges to a better policy and optimizes the cumulative reward. The cumulative reward function graph converged to a constant value, showing no further significant fluctuations, and maintaining a stable value.

Table 5 compares the trading performances of the DQN and the proposed models for evaluating the efficacy of the proposed approach. The DQN and proposed models are identical. In the experiment, 72 stocks are used. Their average performance (standard deviation) is presented.

Among the machine learning models that applied the proposed approach, the Random forest algorithm demonstrates the highest investment performance. While the logistic regression and MLP models show the highest payoff ratio and profit factor, their standard deviations are relatively large compared to the other algorithms. This finding suggests that establishing a stable trading system is difficult. However, the Random forest algorithm has a payoff ratio and profit factor of 1.15 and 1.20, respectively, and achieves high profits with low volatility.

Furthermore, the average winning rate of the machine learning models applying the proposed approach is approximately 0.48, which does not exceed 0.5. However, both the payoff ratio and profit factor exceed 1. For the DQN model, the average winning rate was 0.46, which was not statistically different from the proposed approach. However, the payoff ratio and profit factor for the DQN model are lower at 0.97 and 0.94, respectively. In addition, the Random forest model of the proposed approach exhibits lower volatility and fewer transactions than the DQN model. It also enables the creation of a stable trading system while reducing transaction costs. The Random forest model demonstrates the best experimental result with an average Sharpe ratio of 1.09, which measures the risk-adjusted return of an investment, indicating that with a higher value it provides a stable and efficient investment strategy relative to the risk taken. On the other hand, the DQN model lags behind with a Sharpe ratio of 0.38 when compared to other machine learning models. This suggests inferior performance of the DQN model in comparison to the others. Therefore, rather than applying the DQN model directly to the stock market, it is more efficient to utilize the DQN to identify and learn the optimal market timing to establish a stable trading system.

4.3. Comparison of up-down labeling and DAIS systems

The up-down labeling and investment performance indicators commonly used in most studies are compared to evaluate the proposed approach, which is based on DQN action. Table 6 compares the trading performances of the up-down and DQN action labeling methods. The performance of the 72 stocks used in the experiment is presented in mean (standard deviation) format.

Consequently, the payoff ratio and profit factor performance of the up-down labeling method are lower than those of the proposed approach, despite having more learning data. The standard deviation of the up-down labeling method is lower than that of the proposed approach, thus reducing volatility. However, the inability to achieve profits makes the purpose of the trading system difficult to achieve. There is no significant difference in the number of trades between the up-down labeling method and the proposed approach. However, the proposed approach generates more profits from similar trades. Moreover, the Sharpe ratio derived from the up-down labeling method demonstrates a notable discrepancy, underperforming relative to the results from the proposed DAIS system. While the Random forest model, which performed the best in the proposed approach, achieved an average Sharpe ratio of 1.09, the up-down labeling resulted in a Sharpe ratio of −3.07, indicating a substantial difference. This highlights that the labeling method can have a considerable impact on developing a machine learning stock trading system. Therefore, the choice of labeling method is crucial when establishing a trading system, and the proposed DAIS system offers a more promising approach for achieving stable and efficient trading results. It suggests that this approach effectively addresses noise, non-stationarity, and complexity – which are characteristic of stock market time-series data, and that the learning data produced by the proposed approach are more suitable for trading systems than those generated by the up-down labeling method.

Fig. 5 illustrates a box plot comparison of the proposed DQN Action Instance Selection (DAIS) system and up-down labeling regarding the payoff ratio and profit factor. First, when comparing the payoff ratio, the median value of DAIS exceeds 1 and is higher than that of up-down labeling. This result indicates that DAIS has less deviation in the payoff ratio and can yield more profits on average than up-down labeling. Additionally, when looking at the profit factor, the median value of the proposed DAIS exceeds 1, whereas up-down labeling does not. Therefore, from a long-term trading perspective, DAIS has a higher potential for profit than up-down labeling. Accordingly, we infer that the proposed DAIS system addresses the issues of up-down labeling and learns optimal market timing.

4.4. Trading signal analysis

Figs. 6, 7, and 8 illustrate trading signals for the Random forest of DAIS, DQN, and Random forest of up-down labeling, respectively. The analysis uses stocks with rising, sideways, and falling patterns, which are prevalent in the stock market. The first analysis reveals an increasing trend (035720, Kakao). In the case of DAIS, a buying position is made when the rise began. After the stock price rises again, the position is taken with no short lateral trading. It implies that profits are realized by terminating the transaction appropriately. In the case of the DQN, the buying position is established at the start of the rise; the selling position is initiated when the rise ends, and the stock price falls. This process can generate large returns; however, its disadvantage is that it requires holding stocks for an extended period and does not consider volatility. More transactions were conducted with up-down labeling than in the other trading systems. Nevertheless, it may incur high transaction fees and might not result in large profits.

The second analysis involves the sideways pattern (003550, LG). In the case of DAIS and DQN, losses are prevented by conducting transactions where stock prices are flat. However, in the case of up-down labeling, losses occur due to transaction fees. This result highlights the strength of reinforcement learning in learning the characteristics of stock prices that change in real-time.

The third analysis is of a declining pattern (023530, Lotte shopping). In the case of DAIS, the number of transactions is less than that of the DQN and up-down labeling, which implies a reduction in transactions and losses where possible. It also confirms that profits are realized by increasing the number of transactions in the sideways section. Therefore, the proposed DAIS system learns the optimal market timing and displays generalized performance regardless of rising, sideways, or falling market movements.

5. Conclusion

Our study discusses the construction of a trading system that addresses the limitations of supervised learning and reinforcement learning. We introduce a trading system that learns the optimal trade timing by using use the DQN reinforcement learning algorithm. After determining the timing of trades, we reorganize the learning data using the instance selection technique to solve the noise problem in the stock data. To verify the proposed trading system, we apply it to 72 listed KOSPI stocks and compare it with both supervised learning trading systems and reinforcement learning trading systems. The experiment demonstrates that the proposed trading system has excellent trading performance despite limited learning data. The proposed DAIS trading system can be utilized for high-frequency, or even higher-frequency trading, allowing for multiple trades to be executed throughout the day. By performing predictions for each individual stock within a time frame of less than 1 min, the system can capitalize on short-term price fluctuations and capture frequent trading opportunities. However, there is a concern regarding the training time of the DAIS system. The training time for each individual stock takes approximately 10 min, posing a limitation as it might overlap with real-time trading activities. Therefore, to avoid interference with live trading, it is necessary to conduct the training process during non-trading hours. Additionally, the use of a short sampling window is preferred for high-frequency trading. Nonetheless, this study employed the hold-out method, which considers the entire period, allowing for a comprehensive analysis. Consequently, if one intends to apply the DAIS system for high-frequency trading, alternative sampling window methods need to be adopted.

This study had some limitations. First, we arbitrarily selected the technical indicators to construct the reinforcement learning environment. The results may have been different if we had used more technical indicators or significant technical indicators. Second, we used the DQN algorithm in this study. However, we may find improved trade timings if we employ other reinforcement learning algorithms such as A2C, A3C, and DDPG. In addition, we determined the DQN hyperparameters; however, we can optimize the model through methods such as GridSearch. Third, our results may be biased because we analyzed only large Korean stocks listed on the KOSPI market. Therefore, it is necessary to apply the proposed trading system to various markets such as the NASDAQ, bonds, and futures markets.

CRediT authorship contribution statement

Myeongseok Park: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Jaeyun Kim:** Conceptualization, Methodology, Validation, Resources, Writing – review & editing, Supervision, Project administration. **David Enke:** Validation, Supervision, Writing – review & editing.

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.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1A2C1092808), and this work was supported by the Soonchunhyang University Research Fund.

References

Abarbanell, J. S., & Bushee, B. J. (1997). Fundamental analysis, future earnings, and stock prices. *Journal of Accounting Research, 35*(1), 1–24.
Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS One, 12*(7), Article e0180944.
Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.
Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.
Carter, S., Ferreira, A., Podda, A. S., Recupero, D. R., & Sanna, A. (2021). Multi-DQN: An ensemble of Deep Q-learning agents for stock market forecasting. *Expert Systems with Applications, 164*, Article 113820.
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., Liu, K., Xie, Y., & Hu, M. (2020). Financial trading strategy system based on machine learning. *Mathematical Problems in Engineering, 2020*(1), 1–13.
Dash, R., & Dash, P. K. (2016). A hybrid stock trading framework integrating technical analysis with machine learning techniques. *The Journal of Finance and Data Science, 2*(1), 42–57.
Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1189–1232.
Gorgulho, A., Neves, R., & Horta, N. (2011). Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition. *Expert systems with Applications, 38*(11), 14072–14085.

Han, Y., Kim, J., & Enke, D. (2023). A machine learning trading system for the stock market based on N-period Min-Max labeling using XGBoost. *Expert Systems with Applications, 211*, Article 118581.

Hirchoua, B., Ouhbi, B., & Frikh, B. (2021). Deep reinforcement learning based trading agents: Risk curiosity driven learning for financial rules-based policy. *Expert Systems with Applications, 170*, Article 114553.

Hossain, E., Hossain, M. S., Zander, P. O., & Andersson, K. (2022). Machine learning with Belief Rule-Based Expert Systems to predict stock price movements. *Expert Systems with Applications, 117706*.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112,, 18.

Karaoglu, S., Arpaci, U., & Ayvaz, S. (2017). A deep learning approach for optimization of systematic signal detection in financial trading systems with big data. International Journal of Intelligent Systems and Applications in Engineering, 2017 (Special Issue), 31-36.

Khan, W., Ghazanfar, M. A., Azam, M. A., Karami, A., Alyoubi, K. H., & Alfakeeh, A. S. (2020). Stock market prediction using machine learning classifiers and social media, news. *Journal of Ambient Intelligence and Humanized Computing*, 1–24.

Lee, M. C. (2009). Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications, 36*(8), 10896–10904.

Li, Y., Liu, P., & Wang, Z. (2022). Stock trading strategies based on deep reinforcement learning. *Scientific Programming*.

Lv, D., Yuan, S., Li, M., & Xiang, Y. (2019). An empirical study of machine learning algorithms for stock daily trading strategy. *Mathematical Problems in Engineering*.

Malekipirbazari, M., Aksakalli, V., Shafqat, W., & Eberhard, A. (2021). Performance comparison of feature selection and extraction methods with random instance selection. *Expert Systems with Applications, 179*, Article 115072.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.

Murphy, J. J. (1999). Technical analysis of the financial markets: A comprehensive guide to trading methods and applications. *Penguin.*

Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia, 4*(2), 1883.

Pun, C. S., & Wong, H. Y. (2019). A linear programming model for selection of sparse high-dimensional multiperiod portfolios. *European Journal of Operational Research, 273*(2), 754–771.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106.

Sun, J., & Li, H. (2011). Dynamic financial distress prediction using instance selection for the disposal of concept drift. *Expert Systems with Applications, 38*(3), 2566–2576.

Théate, T., & Ernst, D. (2021). An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications, 173*, Article 114632.

Xu, Y., Lin, W., & Hu, Y. (2020, December). Stock Trend Prediction using Historical Data and Financial Online News. In 2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom) (pp. 1507-1512). IEEE.

Yang, H., Liu, X. Y., Zhong, S., & Walid, A. (2020). Deep reinforcement learning for automated stock trading: An ensemble strategy. In *In Proceedings of the First ACM International Conference on AI in Finance* (pp. 1–8).

Zhong, X., & Enke, D. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications, 67*, 126–139.