frontiers in Applied Mathematics and Statistics

ORIGINAL RESEARCH
published: 26 May 2021
doi: 10.3389/fams.2021.601105

Sentiment-Guided Adversarial Learning for Stock Price Prediction

Yiwei Zhang[1], *Jinyang Li*[1], *Haoran Wang*[1] *and Sou-Cheng T. Choi*[2,3]*

[1]*Beijing Institute of Technology, Haidian District, Beijing, China,* [2]*Kamakura Corporation, Honolulu, HI, United States,*
[3]*Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, United States*

Prediction of stock prices or trends have attracted financial researchers' attention for many years. Recently, machine learning models such as neural networks have significantly contributed to this research problem. These methods often enable researchers to take stock-related factors such as sentiment information into consideration, improving prediction accuracies. At present, Long Short-Term Memory (LSTM) networks is one of the best techniques known to learn knowledge from time-series data and to predict future tendencies. The inception of generative adversarial networks (GANs) also provides researchers with diversified and powerful methods to explore the stock prediction problem. A GAN network consists of two sub-networks known as generator and discriminator, which work together to minimize maximum loss on both actual and simulated data. In this paper, we developed a sentiment-guided adversarial learning and predictive models of stock prices, adopting a popular variation of GAN called conditional GAN (CGAN). We adopted an LSTM network in the generator and a multilayer perceptron (MLP) network in the discriminator. After extensively pre-processing historical stock price datasets, we analyzed the sentiment information from daily tweets and computed sentiment scores as an additional model feature. Our experiments demonstrated that the average forecast accuracies of the CGAN models were improved with sentiment data. Moreover, our GAN and CGAN models outperformed LSTM and other traditional methods on 11 out of 36 processed stock price datasets, potentially playing a part in ensemble methods.

Keywords: stock prediction, data processing, labels, regression, long short-term memory, sentiment variable, conditional generative adversarial net, adversarial learning

OPEN ACCESS

Edited by:
Qingtang Jiang,
University of Missouri–St. Louis,
United States

Reviewed by:
Qi Ye,
South China Normal University, China
Qiang Wu,
Middle Tennessee State University,
United States

***Correspondence:**
Sou-Cheng T. Choi
schoi32@iit.edu

Specialty section:
This article was submitted to Mathematics of Computation and Data Science, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *31 August 2020*
Accepted: *01 February 2021*
Published: *26 May 2021*

Citation:
Zhang Y, Li J, Wang H and Choi S-CT (2021) Sentiment-Guided Adversarial Learning for Stock Price Prediction. Front. Appl. Math. Stat. 7:601105. doi: 10.3389/fams.2021.601105

1 INTRODUCTION

Stock prediction has been attached with great importance in the financial world. While stock fluctuation is very unpredictable, researchers have made every effort to simulate the stock variation because a relatively reasonable prediction can create massive profits and help reduce risks. Stock prediction researchers often consider two kinds of solution methods: classification and regression. Classification methods predict stock movement, while regression methods predict stock prices. Stock movement prediction can be seen as a simple classification task since it only predicts whether the stock will be up or down by a certain amount, or remain almost unchanged. However, many researchers focus their efforts on stock price prediction–a regression task that forecasts future prices with past values – since it could yield more profits than simple movement prediction. In a regression task, researchers have to tackle very complex situations to forecast the future price accurately. Like other time-series prediction problems, many factors

should also be taken into consideration besides historical prices. Investors' sentiments, economic conditions, and public opinions are all critical variables that should be added into a stock prediction system. Particularly, sentiment analysis has been identified as a useful tool in recent years. Not just price and price-related values, researchers have paid more attention to the extraction of sentiment information from newspapers, magazines, and even social media. For example, Ref. [1] examined the relation between social media sentiment and stock prices, in addition to economic indicators in their research on stock market. Ref. [2] Focused on combining stock-related events with the sentiment information from reviews of financial critics. They analyzed sentiments and predicted the fluctuation of the stock market with a matrix and tensor factorization framework. Besides mathematical methods, the development of natural language processing also provides additional techniques for analyzing text and sentiment information. Machine learning, data mining, and semantic comprehension have made extracting large amounts of stock sentiments possible. With the development of social media, people are increasingly inclined to exchange information through the Internet platform. Real-time stock reviews contain a wealth of financial information that reflects the emotional changes of investors. Works such as [3, 4]; and [5] analyzed sentiment information from large numbers of real-time tweets, which were related to stocks and corresponding companies, then investigated the correlation between stock movements and public emotions with supervised machine learning principles. In particular, Ref. [3] utilized two mood tracking tools, OpinionFinder and Google-Profile of Mood States (GPOMS), to analyze the text content of daily tweets, which succeeded in measuring varying degrees of mood. These research results proved that modern techniques are mature enough to handle mountains of sentiment data and that Twitter is a valuable text resource for sentiment analysis.

In our work, we used VADER (Valence Aware Dictionary and sEntiment Reasoner) [6] as a mood tracking tool to help us analyze the sentiment information from tweets. We extracted sentiment features from the past several days of tweets and input them into stock prediction models to predict future stock prices. Applying sentiment analysis to nine stocks, we trained the models with two-month-long training sets with tweets and tested the model performance with the final five days' data. We applied several linear and nonlinear models such as LSTM to this regression task. Moreover, referring to the theories of recurrent neural networks and generative adversarial networks, we designed a sentiment-guided model to improve the accuracy of stock prediction further.

In the remainder of this article, the organization is set in the following order: First, in **Section 2** we review existing stock prediction methods and the development of GANs. Then, in **Section 3**, we introduce our methods in data collection and processing. In **Sections 4** and **5**, we propose our sentiment-guided adversarial learning model as well as comparisons between our models and baselines. Finally, conclusions and future work are discussed at the end of this paper in **Section 6**.

2 RELATED WORKS

In this section, we first present an overview of the relatively recent generative adversarial networks, followed by a brief review of some traditional methods for stock price prediction.

2.1 Adversarial Learning of Neural Networks

Generative adversarial networks (GANs) [7], which try to fool a classification model in an adversarial minimax game, have shown high potential in obtaining more robust results compared to traditional neural networks [8]. In a GAN framework, a generator produces fake data based on noisy samples and attempts to minimize the difference between real and fake distribution, which is maximized by a discriminator oppositely. The GAN framework and GAN-based research have attracted huge attention in various fields recently. Existing GAN works mainly focus on computer vision tasks like image classification [9] and natural language processing like text analysis [10]. With the inception and extensive applications of knowledge representation of natural or complex data such as languages and multi-media, the GAN framework is also widely applied to classification missions for data in representation spaces (e.g., vector spaces of matrices) rather than just for the original data in feature spaces. Some researchers also extended applications of GAN to more challenging problems such as recommendation systems [11] or social network alignment problems [12].

Conditional GANs (CGANs) [13] add auxiliary information to input data, guiding the training process to acquire expected results. Additional information (like class labels or extra data) is fed into both generator and discriminator to perform the conditioning. This architecture is mostly used in image generation [14] and translation [15]. However, we noticed that the CGAN framework has rarely been applied to time-series prediction, specifically, stock prediction problem. In our work, we added sentiment labels to guide the training process of our model and achieved a stronger performance.

Previous work [16] has shown the capacity of GANs for generating sequential data and to fulfill the adversarial training with discrete tokens (e.g., words). Recurrent neural networks (RNNs) have also been widely used to solve problems based on sequential data, such as speech recognition and image generation. Ref. [17] first combined RNNs with a GAN framework and successfully produced many kinds of continuous sequential data, including polyphonic music. Inspired by this work, researchers paid more attention to the potential of RNNs in adversarial training. Refs. [18, 19] succeeded in producing realistic real-valued multi-dimensional time-series data and concentrated their work on medical applications. They built models capable of synthesizing realistic medical data and developed new approaches to create predictive systems in the medical domain. Ref. [20] utilized prerecorded human motion data to train their models and applied their neural network in random motion synthesis, online or offline motion control, and motion filtering. Our work drew on the idea of the RNN-GAN framework and applied time-series analysis to the financial problem of stock prediction. Although similar application in stock prediction has

been raised recently [21], the main difference was that our work also utilized text data from Twitter to create sentiment scores, which could gauge the market mood and guide our models to achieve more accurate prediction.

2.2 Stock Prediction Methods

The stock market prediction problem has a long history and is regarded as an essential issue in the field of financial mathematics. According to the development of research over the years, we can divide the prediction techniques into two categories: statistical and machine learning methods. Frequently used statistical techniques include the autoregressive method (AR), the moving average model (MA), the autoregressive moving average model (ARMA), and the autoregressive integrated moving average (ARIMA) [22]. These methods adopt the principles of random processes and essentially rely on past values of the sequences to analyze and predict the future data. Another technique is generalized autoregressive conditional heteroskedasticity (GARCH) [22], which takes the fluctuation of variance into account and nicely simulates the variation of financial variables. However, both kinds of techniques depend on strong prerequisites such as stationarity and independent and identically distributed (iid) random variables, which may not be satisfied by data in the real-world financial markets.

Recently, machine learning methods (including linear regression, random forest, and support vector machine) stimulate the interest of financial researchers and are applied to forecasting problems. Among them, support vector machine (SVM) and its application in the financial market revealed superior properties compared to other individual classification methods [23]. Since the emergence of deep learning networks, researchers have proved that neural networks can obtain better performance than linear models. Their capacity to extract features from enormous amount of raw data from diverse sources without prior knowledge of predictors makes deep learning a preferred technique for stock market prediction. Artificial neural networks and other advanced neural models like convolution neural networks (CNNs) are evidenced to be good at capturing the non-linear hidden relationships of stock prices without any statistical or econometric assumption [24]. Furthermore, neural networks have also been found to be more efficient in solving non-linear financial problems compared to traditional statistical methods like ARIMA [25]. Nowadays, RNNs are one of the most popular tools in time-series prediction problems. Notably, the LSTM network has been a great success due to its ability to retain recent samples and forget earlier ones [26]. Each LSTM unit has three different gates: forget gate, update gate, and output gate. LSTM units can change their states by controlling their inner operant, namely, their three gates. In our model, we utilized the LSTM network to extract features according to the timeline and to generate fake stock prices from real past price values. We also used sentiment measures to enhance the robustness of our prediction models and make the generative results approach the real distribution.

3 DATA AND METHODS

In this section, we document our data sources and preprocessing techniques. The latter part includes feature engineering, imputation of missing data, fast Fourier transform for denoising data, and last but not least, Isolation Forest for anomaly or outlier detection.

3.1 Data Collection and Sentiment Analysis

Before we started our work, we had the conviction that a collection of tweets related to the stocks could make comparatively accurate models of the investors' mood and thus reach better predictions of the stock market prices. Generally speaking, tweets have neutral, positive, or negative emotions; and we focused on those that contains one or several cashtags (unique identifiers for businesses), which could influence the stock's trend in the following day. If negative sentiment dominated a day, then the next day's stock prices would be expected to fall. The number of followers on one's Twitter account would also be a significant factor. The more followers of an account, the higher the influence of tweets from the account, and the more significant their impact would likely have on stock prices. Cashtags system is a particularly convenient feature of Twitter, allowing users to see what everyone is saying about public companies. The way this system works is similar to the well-known #hashtags of Twitter, except that a cashtag requires "$" followed by a stock symbol (e.g., $GOOG for Google, LLC; $FB for Facebook, Inc.; and $AAPL for Apple Inc.).

For our work, financial tweets and their retweets were downloaded from the website, https://data.world/kike/nasdaq-100-tweets. The time span of these tweets is 71 days from Apr 1–June 10, 2016. We could get all tweets mentioning any NASDAQ 100 companies from this source. The key daily stock price data includes Open prices (O), High prices (H), Low prices (L), and Close prices (C); they were subsequently crawled from Yahoo Finance with the Python package, pandas_datareader. We produced nine datasets, each one including tweets and price values for the nine companies as listed in **Table 1**. These companies are from different sectors/industry groups by the Global Industry Classification Standard (GICS). They could be classified into three price-trend categories: descending prices, mildly fluctuating prices, and ascending prices. **Figure 1** shows the Open price curves of nine stocks with the three distinct trends.

To analyze the sentiment of each tweet, we used VADER [6], which is available from vader-sentiment, a ready-made Python machine learning package for natural language processing. VADER is able to assign sentiment scores to various words and symbols (punctuations), ranging from extremely negative (−1) to extremely positive (+1), with neutral as 0. In this way, VADER could get an overall sentiment score for a whole sentence by combining different tokens' scores and analyzing the grammar frames. We assumed that the neutral emotion plays a much weaker role in the overall market's mood since neural sentiment tends to regard the stock market as unchanged in a specific period. Thus, we excluded the neutral sentiment scores in our analysis and only took the negative and positive moods into consideration. VADER places emphasis on the recognition of

TABLE 1 | Stock prices of nine companies selected for experiments in this study.

Company	Symbol	Sector	Industry group
Apple Inc	AAPL	Information Technology	Technology Hardware & Equipment
Cisco Systems Inc	CSCO	Information Technology	Communications Equipment
Electronic Arts Inc	EA	Communication Service	Media & Entertainment
Eastbay Inc	EBAY	Consumer Discretionary	Internet & Direct Marketing Retail
Endo International PLC	ENDP	Health Care	Pharmaceuticals
Kandy Hotels Co	KHC	Consumer Discretionary	Hotels Resorts & Cruise Lines
NVIDIA Corporation	NVDA	Information Technology	Semiconductors
Starbucks	SBUX	Consumer Discretionary	Consumer Services
Skyworks Solutions Inc.	SWKS	Information Technology	Semiconductors

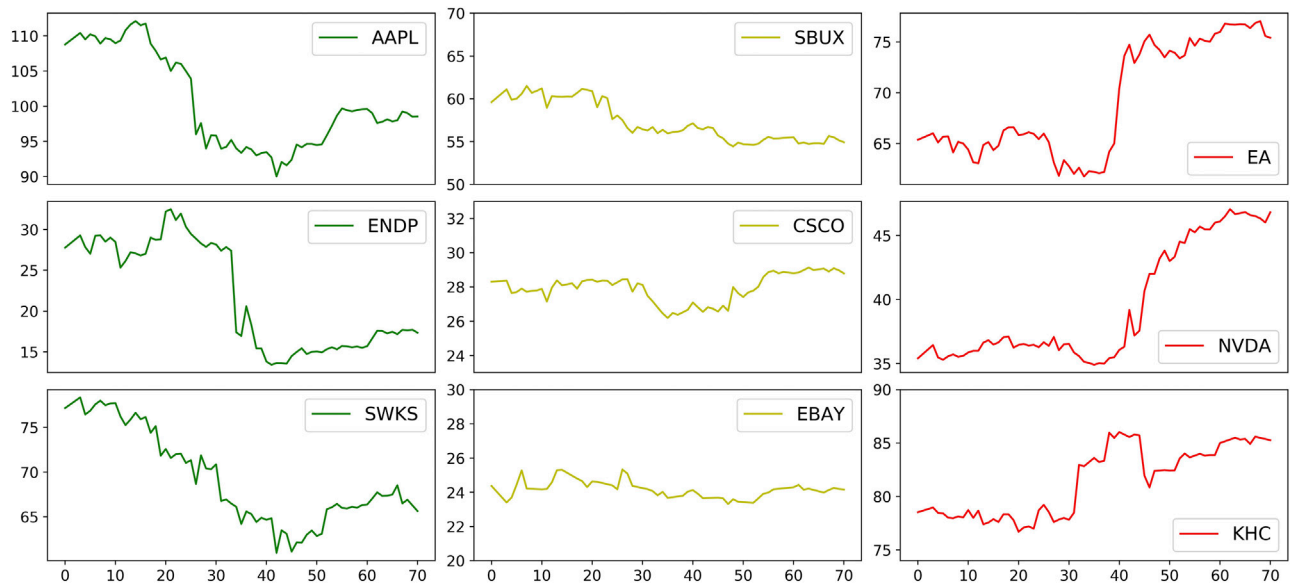

FIGURE 1 | The nine stocks classified into three classes with distinct price trends. The variations of Open price from April 1 to June 10, 2016 are displayed. The first column presents the stocks with descending prices; the second represents the stocks with mildly fluctuating prices; and the third represents the stocks with ascending prices.

uppercase letters, slang, exclamation marks, and the most common emojis. Tweet contents are not written academically or formally, so VADER is suitable for social media analysis. However, we needed to add some new words to the original dictionary of VADER, because VADER missed or misestimated some important words in financial world and therefore caused inaccuracy. For example, "bully" and "bullish" are negative words in the VADER lexicon, but they are positive words in the financial market. We updated the VADER lexicon with the financial dictionary Loughran-McDonald Financial Sentiment Word Lists [27], which include many words used in the stock market. This dictionary has seven categories, and we adopted the "negative" and "positive" lists. We further deleted about 400 existing words in VADER that overlap the Loughran-McDonald Financial Sentiment Word Lists. Finally, we added new negative and positive words to the VADER dictionary and attached sentiment scores to them, with +1.5 to each positive word and −1.5 to each negative word from the Loughran-McDonald lists.

To get one day's overall sentiment score for one stock, we first analyzed all related tweets with VADER and gained the scores of each tweet. Considering that the more followers the bigger influence, we further regarded the number of followers as weights and calculated the weighted average of sentiment scores. The daily percentage change of this average was taken as a comprehensive factor, called compound_multiplied, for 1 day. One problem of our data was that we have only tweets' data but no price data on the non-trading days. To make full use of the tweets' data, we filled the gaps up with the price values from past trading days. We utilized moving averages to fill in the missing values.

We also normalized the compound_multiplied variable as neural networks generally perform better and more efficiently on scaled data. **Figure 2** shows the box plots of compound_multiplied and per_change (see Section 5.1.2 for details) for the nine stocks before the data was scaled.

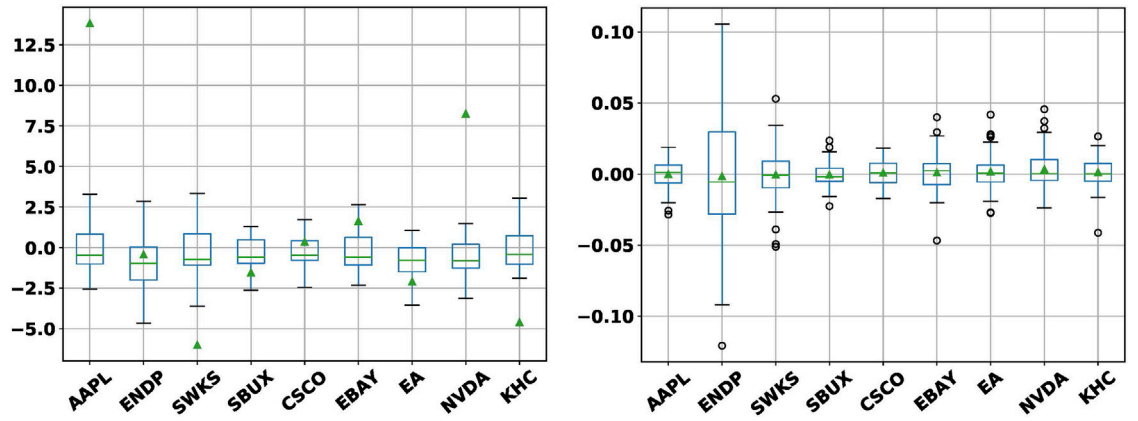

FIGURE 2 | Left: Box plots of changes in daily sentiment scores aggregated from tweets that contained cashtags of selected stocks over the time period Apr 2–Jun 10, 2016. Outliers are omitted to declutter the graph. The triangles mark the mean values. **Right**: Box plots of per_change with outliers shown as circles.

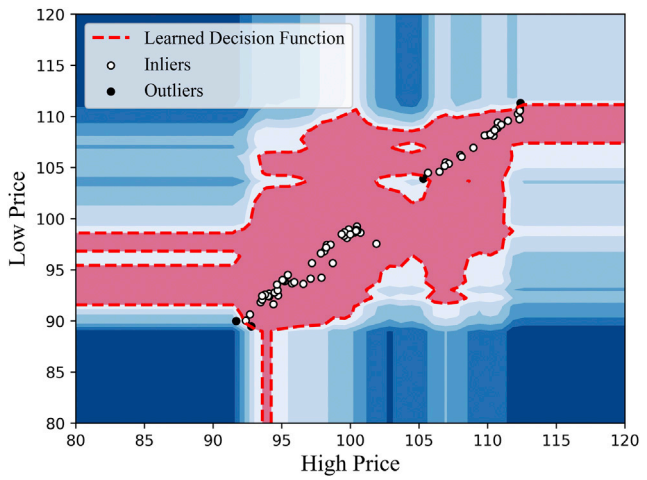

FIGURE 3 | Detecting outliers in the dimensions of High price and Low price with Isolation Forest on AAPL-Open in the training dataset. Only four points of outliers were flagged out by the algorithm.

3.2 Data Processing for Stock Prices

As discussed before, we collected stock prices from nine stock symbols, each of which contains four columns of time-series price data for each trading day: Open, High, Low, and Close. To change the sequential data into suitable input for our models, we did data cleaning and transformation for the training datasets, and approximate normalization for our both training and testing data. We present further details of these transformations below.

3.2.1 Data Cleaning

In consideration of the volatility of the stock market, we had to tackle the price data on days that showed abnormal trends or fluctuation. To accomplish the anomaly detection, we utilized the Isolation Forest (iForest) algorithm [28] to find such data points before treating them. Isolation Forest can directly describe the degree of isolation of data points based on binary trees without using other quantitative indicators. The main steps of distinguishing outliers were as follows: For a group of data with multiple dimensions, we attempted to build a binary search tree (BST). We first picked several dimensions, and then randomly selected one dimension and a value between the maximum and the minimum in that dimension as the root of the BST. Next, we divided the data into two groups (i.e., subtrees of the BST) according to the selected value. Likewise, we continued to subdivide the data according to another randomly selected value in another feature dimension—this step was repeated until the BST could not be further subdivided. Finally, we detected the anomaly data according to the path lengths of the nodes in the BST.

More specifically, in our work, we adopted only two dimensions in the iForest algorithm and detected the outliers within two steps. **(I)** We selected High price and Low price as the two dimensions to detect anomaly situations that stock prices fluctuated intensely in one day. **Figure 3** shows the anomaly detection results when we applied the iForest to the AAPL dataset. **(II)** For each one of the four price series (O, H, L, C), we respectively selected the differences $y_t - y_{t-1}$ and $y_{t+1} - y_t$, in which y_t was the price data on a particular date t, as the two input dimensions to iForest. In this way, we were able to detect anomaly local trends on the timeline. **Figure 4A** shows the detection result of 'Open Price' from the AAPL dataset, and **Figure 4B** displays the anomaly in the form of time series. After we identified each outlier y_t by considering its distances from $y_{t\pm1}$ using iForest, we replaced y_t with the rolling average $\frac{1}{3}(y_{t-2} + y_{t-1} + y_t)$. As a result, we prevented our model from being excessively influenced by outliers during the training process.

3.2.2 Data Transformation

We noticed the successful applications of RNNs in wave forms like sinusoids, so we further transformed our time-series datasets into a wave-like form in order to improve the training effect. In our prediction problem, we assumed that the changes in stock data were periodic, which gave us the opportunity to introduce Fourier Transform into our data processing work. Fourier transform can decompose a periodic function into a linear combination of

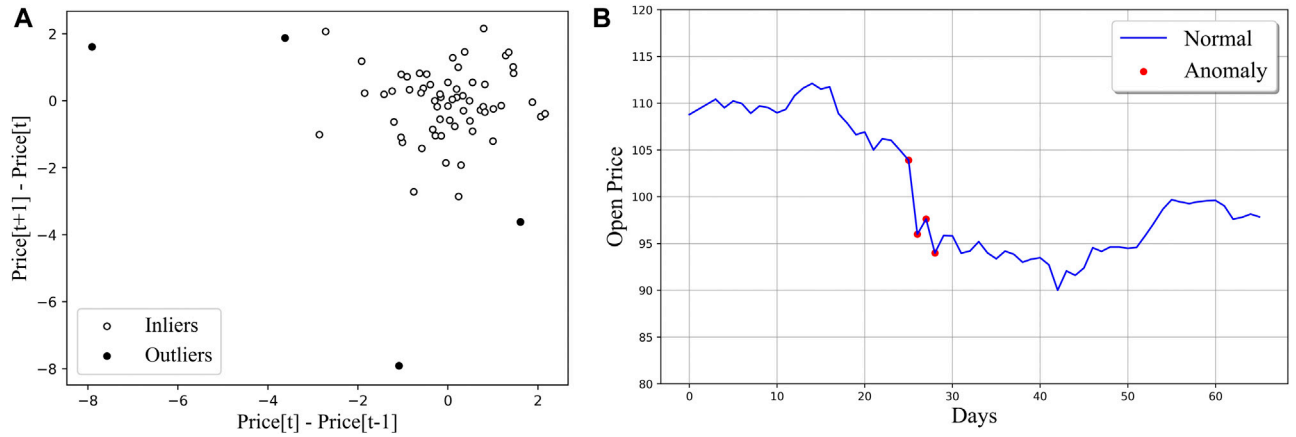

FIGURE 4 | (A) Detecting outliers according to $(y_t - y_{t-1})$ and $(y_{t+1} - y_t)$ with Isolation Forest using the AAPL-Open training set as an illustration. **(B)** Anomaly detection results in the form of time series on the same price values in **(A)**.

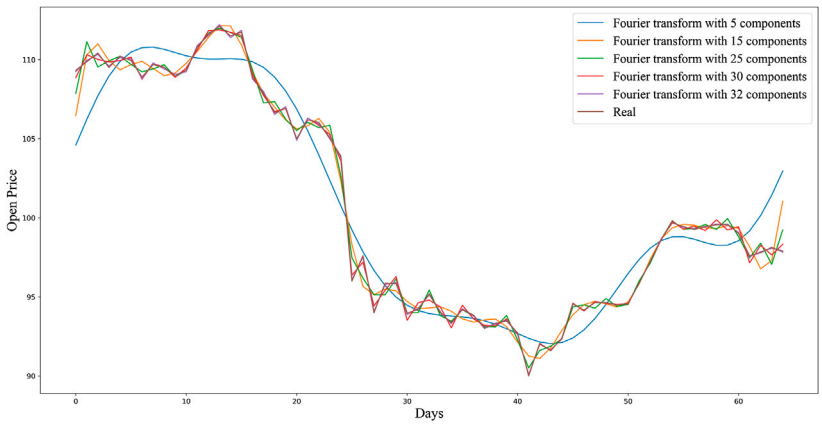

FIGURE 5 | Fourier transform curves with 5, 15, 25, 30, and 32 components on AAPL-Open training set.

orthogonal functions (such as sinusoidal and cosine functions) [29]. With Fast Fourier Transform (FFT), we could extract both global and local trends of our stock datasets and thus reduce their noise. For each of the four prices (O, H, L, C of a stock dataset), we respectively used FFT to create a series of sinusoidal waves (with different amplitudes and frames) and then combined these sinusoidal waves to approximate the original curve. In **Figure 5**, we can see that the pink curve, which was made by a combination of 32 components, closely approximates the original price graph of AAPL-Open training set. By using the derived curve, namely, the denoised sequential data, we could enhance the smoothness of our training set, and reduce noisiness in our data. Consequently, we smoothened the time-series data to make our training data more predictable.

3.2.3 Approximate Normalization

To make sure that our models could learn the features of time-series variables, we also modified the sequential price data to make it satisfy (roughly) normal distribution. In financial research, there is a traditional assumption that the simple daily returns of time-series data are iid normal samples, as long as the sequential data has constant mean and same variance over time; see [30]; for example. We calculated R_t, defined as one-period simple returns of the price values before inputting them into our supervised learning models as targets, where y_t represents one of the four prices for a given stock on day t that we are interested to predict:

$$R_t = \frac{y_t - y_{t-1}}{y_{t-1}}. \tag{1}$$

Let $\widehat{R}_t$ be an estimate of R_t returned by a machine learning model trained with predictors including historical returns, $R_{t-1}, R_{t-2}, \cdots, R_s$ for some $s < t$. Subsequently, we can obtain an estimate of the stock price, $\widehat{y}_t = y_{t-1}(1 + \widehat{R}_t)$.

4 MODEL THEORY

In this section, we present the architecture of our CGAN framework in relation to the sub-networks of our choice.

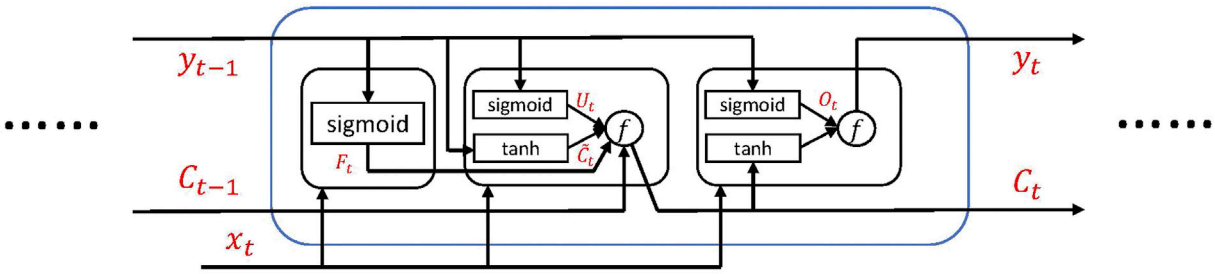

FIGURE 6 | Information flow through an LSTM unit.

4.1 Long Short-Term Memory Networks

Introduced by [31], LSTM is a special kind of RNN, which can learn both short-term and long-term correlations effectively from sequential data. An LSTM network usually consists of many repeating modules, the LSTM units, and we can tune the number of the units to improve the performance of the network. These LSTM units concatenate with each other in line with the information transmitting from one to another. Each unit contains three kinds of gates: Forget gate (decides what kind of old information to be discarded), Update gate (decides what kind of new information to be added), and Output gate (decides what to be output). **Figure 6** shows how the past information flows through an LSTM unit between two time steps, and how the LSTM unit transforms the data in both its long-term and short-term memory. The information transmission process is from C_{t-1} to C_t, which will be changed by inner operants. C_t is a latent variable; χ_t is the input; and y_t is the output. The mathematical operations in the three gates are defined as follows, where $W_{\cdot}$ and $b_{\cdot}$ are parameters to be estimated for minimizing some loss functions:

I. Forget gate:

$$F_t = \text{sigmoid}\left(W_f\left[y_{t-1}, \chi_t\right] + b_f\right). \tag{2}$$

II. Update gate:

$$\begin{aligned} U_t &= \text{sigmoid}\left(W_u\left[y_{t-1}, \chi_t\right] + b_u\right), \\ \tilde{C}_t &= \tanh\left(W_C\left[y_{t-1}, \chi_t\right] + b_C\right), \\ C_t &= F_t{}^{*}\mathrm{C}_{t-1} + \mathrm{U}_t{}^{*}\tilde{C}_t. \end{aligned} \tag{3}$$

III. Output gate:

$$\begin{aligned} O_t &= \text{sigmoid}\left(W_o\left[y_{t-1}, \chi_t\right] + b_o\right), \\ y_t &= O_t{}^{*}\tanh\left(\mathrm{C}_t\right). \end{aligned} \tag{4}$$

Note that sigmoid and tanh are activation functions applied to an input vector elemenwise, whereas * represents elementwise multiplication of vectors. These three gates cooperate with each other and together determine the final information that is output from an individual unit. In this work, we took advantage of the memory property of LSTM and improved its accuracy with adversarial learning framework.

Nowadays, LSTM has been widely applied in many research fields such as machine translation, language modeling, and image generation.

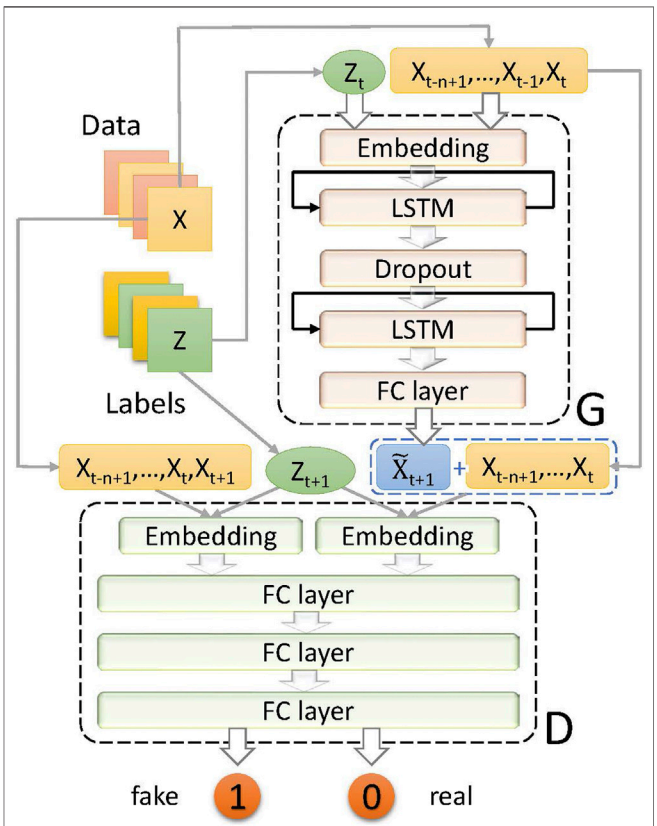

FIGURE 7 | The architecture of our conditional adversarial learning model.

4.2 Adversarial Learning Model

As the conditional GAN framework [13] has proved to be a great success, we adopted this idea in our GAN-based model to improve the training effect. **Figure 7** illustrates the architecture of our GAN-based model. Let $\mathbf{X}_t = \{X_{t-n+1}, \ldots, X_{t-1}, X_t\}$ represent the input data in the form of time series, in which each term is the historical data from the corresponding day from $n-1$ days ago to end of current day t. Note that X_t is a vector that includes all the daily factors on day t. In addition, let Z_t be the sentiment label on day t. Our generator is a multiple-input single-output system, which inputs n historical days' data with m factors from the stock market each day (plus,

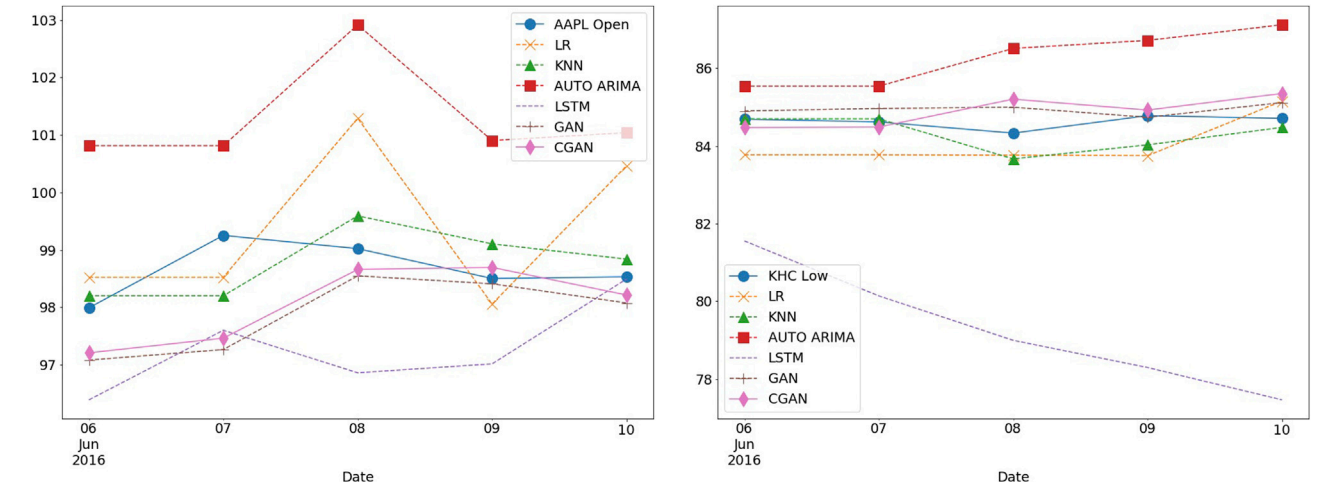

FIGURE 8 | Comparison of the prediction curves of different models on the AAPL-Open and KHC Low test sets. For AAPL Open, among all models, KNN achieved the minimum MAPE over the five-day test period; see **Table 3**. However, on each of the last three days, CGAN actually performed better than KNN. For KHC Low, GAN predictions were closest to the market values and CGAN came next.

the label) and outputs a specific kind of future price data (Open, High, Low, or Close) as the prediction. Since the stock data is typical time-series data, we adopted an LSTM network in our generative model denoted by G. LSTM was trained to extract the features of price data from the past n days, and then used to predict stock prices on day $t + 1$. The generator model G generally consists of three kinds of layers: embedding layer, LSTM layer, and fully connected (FC) layer. An embedding layer first embeds a label in the latent space and turns it into a dense vector, then combines the latent vector with the input matrix, which is the price data from the past in our application. Afterward, an LSTM layer generates some prediction result. A dropout layer could be added between two LSTM layers to make the network have a better generalization and less likely to overfit. Lastly, an FC layer with the Leaky Rectified Linear Unit (ReLU) activation function outputs simulated or fake data $\tilde{X}_{t+1}$, which approximates the real target data X_{t+1}. The output of our generator is defined as follows:

$$\tilde{X}_{t+1} = G(\mathbf{X}_t \mid Z_t). \tag{5}$$

In our work, $m = 4$ and $n = 5$. The four factors were historical Open price, High price, Low price and Close price, which could be seen as four features of the stock data for the price prediction problem. Specifically, we utilized all four kinds of stock prices in the past five business days to forecast each kind of price (O, H, L, or C) on the next day.

The purpose of our discriminator D is to classify input data as real or fake. This classification model is expected to output 0 when it receives real input data or 1 when it receives fake input data. For the input, we concatenated $\mathbf{X}_t$ with $\tilde{X}_{t+1}$ and X_{t+1} respectively to get the fake data $\tilde{\mathbf{X}}_{t+1} = \{X_{t-n+1}, \ldots, X_t, \tilde{X}_{t+1}\}$ and the real data $\mathbf{X}_{t+1} = \{X_{t-n+1}, \ldots, X_t, X_{t+1}\}$. In this way, we could make the discriminator learn the features of sequential data better. Our discriminator D consists of two parts: two embedding layers and a multilayer perceptron (MLP) network. The embedding layers worked in the way same as those in G, i.e., the embedding layers transformed the data and labels into suitable input for MLP. The MLP model then projected the data into higher dimensional space and effectively accomplished the classification task. The Leaky ReLU activation function was used in the hidden layers of the MLP and the sigmoid function was used in the output layer. Returning either 0 and class 1, the output of a discriminator while making a correct decision is as follows:

$$0 = D(\mathbf{X}_{t+1} | Z_t), \tag{6a}$$

$$1 = D\left(\tilde{\mathbf{X}}_{t+1} \middle| Z_t\right). \tag{6b}$$

Specifically, if we select Open price to be predicted by G, we would concatenate the prediction result to Open price values from the past n days. Then, we distinguished this sequence from the corresponding one sampled from real data with D.

We alternatively trained G and D with binary cross-entropy loss (i.e., log loss), $L(\tilde{y}, y) = -y \log(\tilde{y}) - (1 - y)\log(1 - \tilde{y})$, where y is the target value and $\tilde{y}$ is a predicted value. On the one hand, our generator G simulated the fluctuations of the stock market and to minimize the difference between its prediction and the real data. On the other hand, our discriminator D tried to distinguish and maximize that difference. The training of G focused on making the discriminator D confused; it attempted to minimize the adversarial loss so that D could not discriminate the prediction easily. The adversarial loss at each data point on day $t + 1$ for the generator was

$$G_{loss}\left(\tilde{\mathbf{X}}_{t+1}\right) = L\left(D\left(\tilde{\mathbf{X}}_{t+1} \middle| Z_t\right), 0\right). \tag{7}$$

The training of D focused on improving its ability to distinguish the difference between $\mathbf{X}_{t+1}$ and $\tilde{\mathbf{X}}_{t+1}$, so it attempted to maximize the adversarial loss at each data point on day $t + 1$:

$$D_{\text{loss}}\left(\mathbf{X}_{t+1}, \tilde{\mathbf{X}}_{t+1}\right) = L(D(\mathbf{X}_{t+1} | Z_t), 0) + L\left(D\left(\tilde{\mathbf{X}}_{t+1} \middle| Z_t\right), 1\right). \tag{8}$$

In general, the generator tried to make $D(\tilde{\mathbf{X}}_{t+1}|Z_t) = 0$ while the discriminator tried to achieve $D(\mathbf{X}_{t+1}|Z_t) = 0$ and $D(\tilde{\mathbf{X}}_{t+1}|Z_t) = 1$. [7] defined the loss function of this specific binary classification task as cross-entropy loss. To achieve the training process described by **Eq. (7)** and **Eq. (8)**, we updated D by maximizing $E_{x \sim p_{\text{real}}(x)}[\log D(x|z)] + E_{\tilde{x} \sim p_{\text{fake}}(\tilde{x})}[\log(1 - D(G(\tilde{x}|z)))]$, and then updated G by minimizing $E_{\tilde{x} \sim p_{\text{fake}}(\tilde{x})}[\log(1 - D(G(\tilde{x}|z)))]$. In this way, the total loss function of the minimax game over all training samples was as follows:

$$\min_G \max_D L_{\text{RMSprop}}(D, G) = E_{x \sim p_{\text{real}}(x)}[\log D(x|z)] + E_{\tilde{x} \sim p_{\text{fake}}(\tilde{x})}[\log(1 - D(G(\tilde{x}|z)))], \tag{9}$$

where $p_{\text{real}}(x)$ was the distribution of real data, $p_{\text{fake}}(\tilde{x})$ was the distribution of fake data, and lastly, z represented the label vector.

The two models G and D were trained iteratively with the RMSprop optimizer, a stochastic gradient descent algorithm with mini batches [32]. In each iteration, we first trained the discriminator r times and then trained the generator one time, in which r was seen as a hyper-parameter to be tuned in our training process. The reason for training D first was that a trained discriminator would help improve the training effect of the generator. Besides, we also adopted gradient clipping (ensuring the norm of a gradient not too large in magnitude) for both G and D at the end of each iteration to avoid the potential problem of gradient explosion.

If Z_t is omitted from the discussion in this subsection, then we have simply a GAN model. In practice, we used Keras [33] with TensorFlow [34] version 2 as the backend to implement our LSTM, GAN, and CGAN models.

5 EXPERIMENT

In this section, we summarize the numerical results of our learning and prediction models in multiple error measures in response to hyperparameter search when applicable.

5.1 Experimental Settings

We collected price data and related tweets of nine companies from April 1st to June 10th, 2016. The data before June 5th was taken as the training set and the last five days' data as the test set. For each dataset, we respectively experimented on Open price, High price, Low price, and Close price, so we totally built and tested $9 \times 4 = 36$ models and datasets (36 groups of experiments).

We have already illustrated the main data processing work in **Section 3**. Here, we elaborate on specific configuration of our models.

5.1.1 Baseline Models

Linear Multiple Regression (***LMR***) is a classical statistical approach for analyzing the relationship between one dependent variable and one or more independent variables. Its simple formula in matrix notation is $Y = X\beta + \varepsilon$, where ε is iid normal with mean $\mathbf{0}$ and variance matrix, $\sigma^2\mathbf{I}$, and $\mathbf{I}$ is the identity matrix of order the same as number of independent observations in Y.

K-Nearest Neighbors (***KNN***) [35] can be used in both classification and regression problems. In our time-series regression problem, the input is the k closest training sequences from the feature space according to the timeline, and the output is the average of the values of the k nearest neighbors.

Autoregressive Integrated Moving Average (***ARIMA***) is composed of autoregression (AR), an integrated (I) model that calculates differences, and moving average (MA). Auto ARIMA [36, 37] can automatically perform grid search with parallel processing to find an optimal combination of p, q, and d, which are the parameters associated with order of AR, degree of differencing with I, and order of MA, respectively.

Long Short-Term Memory is a special kind of recurrent neural network. We tuned three hyper-parameters to improve its training effect: the number of LSTM units, step size, and training epochs.

5.1.2 Data Pre-Processing Techniques

Besides Open price, High price, Low price, and Close price, we also added two more engineered factors to the dataset as input to predict the price on day $t + 1$. One was the sentiment variable, compound_multiplied, which was obtained with the method illustrated in **Section 3.1**. The other was per_change, which was obtained by the equation: per_change := $100\% \times [\text{Close}(t) - \text{Open}(t)]/\text{Open}(t)$. **Table 2** shows the AAPL dataset sample with six columns, where the compound_multiplied was the sentiment variable.

Transformations for baseline models (***LMR, KNN, and auto ARIMA***) When predicting one type of stock price mentioned above, we utilized the two derived factors, per_change and compound_multiplied, as predictors.

Techniques for neural-network models (***LSTM, GAN, and CGAN***) **(I)** As for LSTM, we chose only three factors as input. For instance, we used 'Open price', per_change and compound_multiplied from day t as input when we predicted the Open price on day $t + 1$. However, for our adversarial learning model, we chose four kinds of prices as input and utilized the other two factors to create labels. The details about the labels were discussed below. **(II)** We converted time-series data to data that was fitting in supervised learning problems. Specifically, if we predicted the price on day $t + 1$ with the prices from the past 5 days, we would create a six-term sequence $X_{t+1} = \{x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1}, x_t, x_{t+1}\}$. In other words, we used the first five time-lagged columns as input and the last column as the real target price on day $t + 1$.

5.1.3 Settings for Adversarial Learning Models

Labels We utilized the two variables, compound_multiplied and per_change, to create our final sentiment label in the CGAN models. The reason for adding the per_change was that we would like to take the local trend of past stock variation into account. Finally, the label was set to three classes: 0, 1, and 2. The three classes respectively represent three different tendencies of forecasting prices: up, down, and almost unchanged. Our

TABLE 2 | Stock price data and two engineered features on the first five days of the AAPL dataset, where NaN represents Not a Number.

Date	Open	High	Low	Close	per_change	compound_multiplied
4/1/16	108.78	110.00	108.20	109.99	1.1123	NaN
4/2/16	109.33	110.73	108.89	110.37	0.9529	1.1750
4/3/16	109.87	111.46	109.58	110.74	0.7934	-0.1666
4/4/16	110.42	112.19	110.27	111.12	0.6339	-0.1836
4/5/16	109.51	110.73	109.42	109.81	0.2739	0.2064

model was trained to learn the categories from labels with the training set and enhance its prediction accuracy. Let Z_t represent the label on day t. The process of creating the labels could be captured by the following min-max scaling of each $v_t \in [0, 3]$ before mapping it to $Z_t \in \{0, 1, 2\}$, meaning $[0, 1) \mapsto 0$, $[1, 2) \mapsto 1$, and $[2, 3] \mapsto 2$:

$$v_t = \lambda_1 \times \text{compound\_multiplied}(t) + \lambda_2 \times \text{per\_change}(t), \quad (10)$$
$$Z_t = \min\left(2, \left\lfloor 3\frac{v_t - \min(v_t)}{\max(v_t) - \min(v_t)}\right\rfloor\right),$$

where $\lambda_1 + \lambda_2 = 1$ and $0 \le \lambda_2, \lambda_2 \le 1$. Empirically, tuning λ_1 and λ_2 could change the training accuracies.

We also did experiments to evaluate the impact of the sentiment analysis had made in our work. Specially, we modified our CGAN models by taking out the embedding layers and not inputting the sentiment labels Z_t. Without the labels Z_t, we have GAN models. In this way, we could compare the training effect of our models with or without sentiment labels.

Hyper-parameters As discussed in Section 4, we alternately trained the D and the G. The epoch ratio, r, defined as epochs of D to epochs of G, was in the range of $[1, 4]$.

The learning rates for D and G, α_D and α_G, were searched in the range, $[5 \times 10^{-5}, 8 \times 10^{-4}]$, respectively, with the same decay, 10^{-8}. We also tuned α_D and α_G in our experiments to improve the training process. Besides, the gradient clipping thresholds were set to be the same at 0.8 for both the generator and the discriminator.

5.2 Results and Discussions

We used mean absolute percentage errors (MAPE) as the main metric to evaluate the performance of our model. The metric equation is

$$MAPE = \frac{100\%}{N}\sum_{t=1}^{N}\left|\frac{y_t - \tilde{y}_t}{y_t}\right|,$$

where y_t is the real price on day t, $\tilde{y}_t$ is the prediction price by a model, and N is number of data points. In our experiments, we first trained the CGAN model until the loss curves of both the generator and the discriminator converged. The training process was described in **Section 4**. After that, we tested the generator "LSTM model" to get the prediction results of testing data. **Figure 8** displays the comparison of prediction curves on the AAPL-Open and KHC-Low test sets. As our test sets contained five-day-long data, we calculated the average MAPEs for the five days' OHLC prices as the performance metric of the model.

Empirically, we tuned three kinds of hyper-parameters to improve the performance of our adversarial learning model: the learning rates of the sub-networks, α_D and α_G; the epoch ratio r; weightage λ_1 and λ_2 in **Eq. (10)**. **Table 3** show the MAPE results of the nine test sets. We did four groups of experiments, respectively on O, H, L, C, for each test set and then calculated the average MAPE to compare the performances of different models effectively. KNN achieved minimum mean MAPE for AAPL, SBUX, CSCO, SWKS, and KHC; linear models for EA and ENDP; GAN and CGAN for NVDA and EBAY, respectively. While KNN achieved minimum errors most of the time, GAN and CGAN were the best for 11 of the 36 stock prices and the second best for 18. In terms of overall average of all nine MAPEs (last column of **Table 4**), KNN was the best and CGAN came as a close second. Without sentiment labels, GAN models on average had a higher average of all MAPEs than CGAN, showing that our sentiment labels help generally in improving price prediction accuracy.

Root mean square errors (RMSE), mean square errors (MSE), mean absolute errors (MAE), and symmetric mean absolute percentage errors (SMAPE) were also used to verify the training results. Their defining equations are recapped here:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - \tilde{y}_t)^2}, \qquad MSE = \frac{1}{N}\sum_{t=1}^{N}(y_t - \tilde{y}_t)^2,$$
$$MAE = \frac{1}{N}\sum_{t=1}^{N}|y_t - \tilde{y}_t|, \qquad SMAPE = \frac{100\%}{N}\sum_{t=1}^{N}\frac{|y_t - \tilde{y}_t|}{(|y_t| + |\tilde{y}_t|)/2}.$$

To evaluate our model in a more comprehensive view, we selected AAPL and EBAY to illustrate these four metrics. We respectively obtained the results of O, H, L, C from all models and then calculate the average errors to compare the performance more carefully. **Table 5** for AAPL shows that KNN performed best on average while CGAN came second. **Table 5** for EBAY demonstrates that the CGAN models outperformed all the baselines not only in MAPE but also in RMSE, MSE, MAE, and SMAPE.

As discussed above, we treated λ_1 and λ_2 as hyper-parameters. Considering the importance of condition labels, adjusting the proportion of sentiment information would be necessary every time we trained the CGAN model. For short-term stock data, we need to train the model only once, since people's attitude toward a stock often remains more or less the same in a short period. However, readjusting λ_1 and λ_2 every month maybe useful. We collected another tweet dataset about Apple Inc. with a time span from Jan. 1, 2014 to Dec. 31, 2015 (available from stocknet-dataset by [5]. We selected three periods in these two-year-long datasets and calculated the MAPEs of prediction

TABLE 3 | Tables 3a–i present the MAPEs of forecasted O, H, L, and C prices from various models for nine stock symbols. The smallest MAPE is highlighted in bold across the models in each column for every stock price; the second smallest is underlined. The last column contains the average values of the MAPEs of the four price targets.

	Open	High	Low	Close	Mean/%
a). MAPEs of the AAPL test set.					
LMR	1.1958	2.9199	1.4223	2.1880	1.9315
KNN	**0.5546**	**0.8539**	**0.4884**	**0.4090**	**0.5765**
ARIMA	2.6741	1.4260	3.2027	2.6597	2.4906
LSTM	1.4061	4.5924	5.7139	5.2507	4.2408
GAN	0.7929	1.5085	1.2227	1.2868	1.2027
CGAN	0.7067	1.4688	0.6841	1.3076	1.0418
b). MAPEs of the SBUX test set.					
LMR	0.9019	1.1674	1.3397	1.2923	1.1753
KNN	0.5593	**0.4919**	0.6338	**0.5265**	**0.5529**
ARIMA	1.1861	1.0757	1.1539	1.0580	1.1184
LSTM	**0.5040**	3.8398	4.0186	3.7970	3.0399
GAN	0.6448	0.8664	0.8240	0.9556	0.8227
CGAN	0.6888	0.9053	**0.6280**	0.8969	0.7798
c). MAPEs of the EA test set.					
LMR	**0.5037**	0.8308	**0.9628**	0.9709	**0.8170**
KNN	0.5964	0.8414	0.9878	1.0162	0.8604
ARIMA	3.1441	3.8026	1.4712	2.4455	2.7158
LSTM	0.8987	16.9632	17.0647	16.4948	12.8554
GAN	0.9095	**0.8225**	1.0935	**0.7298**	0.8888
CGAN	1.2390	1.0105	1.3181	0.7448	1.0781
d). MAPEs of the ENDP test set.					
LMR	2.6631	2.892	**3.2137**	**2.8585**	**2.9068**
KNN	2.4014	**2.4446**	4.3038	3.4666	3.1541
ARIMA	7.5329	10.8514	7.8179	12.4049	9.6518
LSTM	**1.9243**	33.3106	27.6254	37.3947	25.0638
GAN	3.4240	53.2661	3.9620	4.8264	16.3696
CGAN	5.4597	3.6090	4.0107	3.1991	4.0696
e). MAPEs of the CSCO test set.					
LMR	1.1367	0.6497	0.8273	0.4343	0.7620
KNN	**0.3813**	**0.1555**	0.7219	**0.1840**	**0.3607**
ARIMA	1.3303	1.7011	2.7049	1.1942	1.7326
LSTM	0.5645	7.2849	7.9551	8.0766	5.9703
GAN	0.4081	0.2811	**0.6025**	0.2095	0.3753
CGAN	0.6318	0.4245	0.6968	0.2559	0.5022
f). MAPEs of the NVDA test set.					
LMR	2.4894	2.5058	2.0899	3.6247	2.6774
KNN	0.9341	1.0531	0.6399	1.5637	1.0477
ARIMA	2.1786	1.9938	1.5984	2.0929	1.9659
LSTM	**0.8402**	12.1243	10.8188	10.297	8.5201
GAN	1.0658	**0.7822**	**0.4776**	**1.2834**	**0.9022**
CGAN	1.1604	2.4538	0.4854	1.4711	1.3927
g). MAPEs of the SWKS test set.					
LMR	**1.3381**	1.6378	1.2815	**1.6442**	1.4754
KNN	1.5065	**1.1174**	1.2586	1.6564	**1.3847**
ARIMA	2.5976	2.0430	2.5419	3.4855	2.6670
LSTM	1.9081	6.2990	6.3457	7.1987	5.4379
GAN	1.9350	1.7313	1.2501	2.0234	1.7350
CGAN	1.7149	1.8520	**1.2180**	2.1641	1.7372
h). MAPEs of the EBAY test set.					
LMR	0.3222	1.0901	0.813	1.0681	0.8234
KNN	0.8402	0.5033	0.6336	0.7170	0.6735
ARIMA	1.1214	1.3163	1.5132	1.7715	1.4306
LSTM	**0.2329**	1.2547	**0.5792**	0.6469	0.6784
GAN	0.6034	0.5628	0.7878	**0.5867**	0.6352
CGAN	0.5781	**0.4940**	0.7231	0.6964	**0.6229**
i). MAPEs of the KHC test set.					
LMR	0.7283	0.7682	0.8998	1.4204	0.9542
KNN	**0.3715**	**0.1547**	0.4089	**0.4968**	**0.3580**
ARIMA	1.0659	1.0437	1.9594	0.6984	1.1919

(Continued in next column)

TABLE 3 | (*Continued*) Tables 3a–i present the MAPEs of forecasted O, H, L, and C prices from various models for nine stock symbols. The smallest MAPE is highlighted in bold across the models in each column for every stock price; the second smallest is underlined. The last column contains the average values of the MAPEs of the four price targets.

	Open	High	Low	Close	Mean/%
LSTM	0.8691	5.9214	6.3042	5.9614	4.7640
GAN	0.5725	0.2316	**0.3953**	0.7931	0.4981
CGAN	0.5304	0.3763	0.4749	0.6463	0.5070

TABLE 4 | Counts number of times a model is the best and second best in predicting the stock prices, and lists the mean of all mean values in the last column of Tables 3a–i.

Overall performance of all models

	Best	Second	Mean/%
LMR	6	4	1.5026
KNN	14	12	**0.9965**
ARIMA	0	1	2.7738
LSTM	5	1	7.8412
GAN	8	8	2.6033
CGAN	3	10	1.3035

TABLE 5 | Comparison of prediction results in different metrics on the AAPL and EBAY test sets using the average values of model errors on O, H, L, C prices. The smallest errors are displayed in bold font across the models we have built; the second smallest errors are underlined.

Model	RMSE	MSE	MAE	MAPE/%	SMAPE/%
a). Errors of the AAPL test set.					
LMR	2.2721	5.8504	1.9176	1.9315	1.9231
KNN	**0.7154**	**0.5680**	**0.5721**	**0.5764**	**0.5753**
ARIMA	2.6184	7.1359	2.4613	2.4906	2.4546
LSTM	5.1174	30.7145	4.2006	4.2408	4.1406
GAN	1.5321	2.5507	1.1970	1.2028	1.2152
CGAN	1.3464	2.0495	1.0378	1.0418	1.0511
b). Errors of the EBAY test set.					
LMR	0.2457	0.0689	0.1994	0.8234	0.8287
KNN	0.1970	0.0392	0.1628	0.6735	0.6765
ARIMA	0.4410	0.2071	0.3453	1.4306	1.4142
LSTM	0.2248	0.0617	0.1636	0.6784	0.6737
GAN	0.1903	0.0369	0.1534	0.6351	0.6368
CGAN	**0.1867**	**0.0353**	**0.1503**	**0.6229**	**0.6230**

results for the Open price. In the three groups of experiments, we chose three different (λ_1, λ_2) pairs in our CGAN model with $\lambda_2 := 1 - \lambda_1$. We also applied GAN as the baseline to highlight the influence of condition labels. As shown in **Table 6**, for the first test period, which follows immediately after the time span of the training data, the MAPEs from CGAN and GAN were least (see the first row of each group in **Table 6**). However, both models performed worse with increasingly larger MAPEs during the subsequent test periods except for GAN in the last training scenario. Thus, the learning models should be trained periodically, or whenever the test errors are larger than user-desired tolerances, especially when they are used in real-time prediction tasks.

TABLE 6 | Comparison of the MAPEs for AAPL-Open dataset from CGAN and GAN over three different testing periods for each training time span. The smallest error in each test group for each model is highlighted in bold.

Time span for training set	Time span for testing set	λ_1	CGAN/%	GAN/%
2014-01-01–2014-03-31	2014-04-01–2014-04-10	0.2	**3.81**	**3.73**
	2014-04-11–2014-04-20		5.52	4.93
	2014-05-01–2014-05-10		3.88	3.77
2014-07-01–2014-09-30	2014-10-01–2014-10-10	0.2	**0.77**	**0.71**
	2014-10-11–2014-10-20		1.03	0.97
	2014-11-01–2014-11-10		0.95	0.88
2015-01-01–2015-03-31	2015-04-01–2015-04-10	0.9	**2.90**	8.98
	2015-04-11–2015-04-20		9.58	**8.71**
	2015-05-01–2015-05-10		10.65	9.79

6 CONCLUSIONS AND FUTURE WORK

For individual investors and investment banks, rational prediction through statistical modeling helps decide stock trading schemes and increase their expected profits. However, it is well known that human trading decisions are not purely rational and the irrational drivers are often hard to observe. Introducing proxies of irrational decision factors such as relevant online discussion from Twitter followed by sentiment analysis has become an advanced approach in financial price modeling.

In this work, we have successfully built a sentiment-conditional GAN network in which LSTM served as the generator. We encoded Twitter messages, extracted sentiment information from the tweets, and utilized it to conduct our stock prediction experiments. The experiments showed, for about a third of our models, superior properties of our GAN and CGAN models compared to LMR and ARIMA models, KNN method, as well as LSTM networks. Our GAN and CGAN models could better adapt to the variations in the stock market and fluctuation trends of some real price data. Hence they could play a role in ensemble methods that combine strong models with small prediction errors to achieve better accuracies than any individual model when no one model would always perform best; see, for example, [38].

Even though our neural network models could outperform the simpler baseline models at times, it could still be enhanced. The main obstacle in our work was that we failed to find tweets' dataset that was longer than 70 days, which might largely weaken the training effect. For instance, we initially planned to add more labels to our input data, since it would help represent varying degrees of sentiment tendencies instead of only 'up', 'down', and 'almost unchanged'. Nevertheless, the GAN-based models could not learn better with more labels in such a short time-series dataset. Therefore, we finally chose only three classes to make sure that our model would be fully trained. The same consideration went when we created the sentiment variable, which was the reason why we only selected 'positive' and 'negative' lists when adding new words to the dictionary. If we could get a dataset with a longer time coverage, we would be able to do experiments with sentiment labels in more categories and potentially further improve our models.

Another shortcoming in our current approach is that we only took sentiment factors into account. The stock market is very complicated and there are potentially a great many factors to be considered. There are some other factors like economic growth, interest rates, stability, investor confidence and expectations. We also noticed that political events would have a huge impact on the variation of the stock market. We can extract political information from newspapers and news websites. If we added these factors to the input, our model may better learn the features of the stock market and make the prediction tendencies close to that in the real world.

In this work, we have assumed that the tweets are more or less truthful. However, social media sources could be contaminated with fake news or groundless comments, that are hard to be distinguished from the good ones with real signals. A fruitful area of future research is to look into GAN models for alleviating the problem.

It is also known that neural network models often need much bigger datasets to beat simpler models. To this end, there are multiple ways we could explore: consider more than nine stocks; join all stock prices into one single dataset, i.e., to train one model on panel data grouped by stocks and prices (O, H, L, C) instead of a single time series; sample our datasets at hourly frequency.

Lastly, many properties of neural networks are still active areas of research. For example, in LSTM, GAN, and CGAN models, the loss function values and the solution quality often appear to be sensitive to small changes in inputs, hyperparameters, or stopping conditions. Creating stable yet efficient numerical algorithms are necessary for reliable solutions. In addition, the success of neural networks are often associated with computer vision problems. Adopting them in finance may require different techniques or transformations.

In the future, we would like to explore the topic further in the following directions:

1) Obtaining larger datasets and creating labels in more classes to improve our model by examining hourly data, for example.
2) Building GAN models for detecting overhyped or deceitful messages about a stock before incorporating them into our models.
3) Attempting to extract more stock-related factors and adding them as predictors in our models.
4) Experimenting with more stocks or other financial assets, and considering having one model for all stock price data.

5) Utilizing more sophisticated natural-language processing (NLP) methods to analyze the financial or political information from news media and assessing the impact and the role they play in the stock market.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found here: https://data.world/kike/nasdaq-100-tweets.

AUTHOR CONTRIBUTIONS

YZ: Drafted the manuscript; developed code for data preprocessing and sentiment analysis; designed modeling methods and experiments. JL: Modified the manuscript and cooperated in the design of data processing. HW: Modified the manuscript and cooperated in the design of sentiment analysis. S-CC: Provided directions and supervision on scientific principles and methodology for the whole project; reviewed and modified code for computational scalability and reproducibility; edited manuscript for content accuracy and clarity.

ACKNOWLEDGMENTS

The authors would like to thank the editor, Prof. Qingtang Jiang, and reviewers, Prof. Qi Ye and Prof. Qiang Wu, for their invaluable support and feedback during the anonymous review process. In addition, we are grateful to the following seminar speakers for inspiration during the development of this work in the 2019 IIT Elevate Summer Program's research course, *SCI 498-106 Machine Learning Algorithms on Heterogeneous Big Data:* Victoria Belotti, Adam Ginensky, Joshua Herman, Rebecca Jones, Sunit Kajarekar, Hongxuan Liu, Lawrence K. H. Ma, Qiao Qiao, Jagadeeswaran Rathinavel, Aleksei Sorokin, Huayang (Charlie) Xie, and Zhenghao Zhao. In addition, the last author would like to thank the following colleagues for discussion and encouragement: Prof. Robert Ellis, Prof. Robert Jarrow, Jackson Kwan, Wilson Lee, Prof. Xiaofan Li, Prof. Lek-Heng Lim, Aleksei Sorokin, Don van Deventer, and Martin Zorn.

REFERENCES

1. Devi KN, and Bhaskaran VM. Impact of social media sentiments and economic indicators in stock market prediction. *Int J Comput Sci Eng Technol* (2015) 6.
2. Zhang X, Zhang Y, Wang S, Yao Y, Fang B, and Yu PS. Improving stock market prediction via heterogeneous information fusion. *Knowledge-Based Syst* (2018) 143:236–47. doi:10.1016/j.knosys.2017.12.025
3. Bollen J, Mao H, and Zeng X. Twitter mood predicts the stock market. *J Comput Sci* (2011) 2:1–8. doi:10.1016/j.jocs.2010.12.007
4. Pagolu VS, Reddy KN, Panda G, and Majhi B. Sentiment analysis of Twitter data for predicting stock market movements. In: International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES); 2016 Oct 3–5; Paralakhemundi, India. New York, US: IEEE (2016) p. 1345–50.
5. Xu Y, and Cohen SB. Stock movement prediction from tweets and historical prices. In Proceedings of the 56th annual meeting of the association for computational linguistics (Vol. 1: Long Papers); 2018 July; Melbourne, Australia. Stroudsburg, Pennsylvania: Association for Computational Linguistics (2018) 1970–9.
6. Hutto CJ, and Gilbert E. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In Eighth International AAAI Conference on Weblogs and Social Media; 2014 June 1-4; Ann Arbor, MI. The AAAI Press (2014).
7. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, and Ozair S. Generative adversarial nets. *Adv Neural Inf Process Syst* (2014) 2672–80.
8. Goodfellow IJ, Shlens J, and Szegedy C. Explaining and harnessing adversarial examples. In: Bengio A, LeCun Y, editors. 3rd International Conference on Learning Representations, {ICLR} 2015; 2015 May 7–9; San Diego, CA (2015). Available from: http://arxiv.org/abs/1412.6572.
9. Kurakin A, Goodfellow I, and Bengio S. (2017) Adversarial Machine Learning at Scale, 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings, Toulon, France, Apr 24–26, 2017, OpenReview.net. Available from: https://openreview.net/forum?id=BJm4T4Kgx
10. Miyato T, Dai AM, and Goodfellow I. Adversarial Training Methods for Semi-Supervised Text Classification, 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings Toulon, France, Apr 24–26, 2017, OpenReview.net. Available from: https://arxiv.org/abs/1605.07725
11. He X, He Z, Du X, and Chua TS. Adversarial personalized ranking for recommendation. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval; 2008 June. New York, US: ACM (2018) p. 355–64.
12. Li C, Wang S, Wang Y, Yu P, Liang Y, and Liu Y. Adversarial learning for weakly-supervised social network alignment. In: Proceedings of the AAAI Conference on Artificial Intelligence; 2019 Jan 27-Feb 1; Honolulu, Hawaii. Menlo park, California: AAAI (2019) 33:996–1003. doi:10.1609/aaai.v33i01.3301996
13. Mirza M, and Osindero S. Conditional generative adversarial nets. *CoRR* abs/1411.1784 (2014). Available from: http://arxiv.org/abs/1411.1784
14. Antipov G, Baccouche M, and Dugelay JL. Face aging with conditional generative adversarial networks. In: IEEE International Conference on Image Processing (ICIP); 2017 Sept 17–20; Beijing, China. New York, US: IEEE (2017) p. 2089–93.
15. Yang Z, Chen W, Wang F, and Xu B. Improving neural machine translation with conditional sequence generative adversarial nets. *CoRR* abs/1703.04887 (2017). Available from: http://arxiv.org/abs/1703.04887
16. Yu L, Zhang W, Wang J, and Yu Y. SeqGAN: Sequence generative adversarial nets with policy gradient. In: Thirty-First AAAI conference on artificial intelligence. New York, US: ACM (2017) 2852–8.
17. Mogren O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *Constructive Machine Learning Workshop at NIPS 2016 in Barcelona, Spain*. Available from: https://arxiv.org/pdf/1611.09904.
18. Esteban C, Hyland SL, and Rätsch G. Real-valued (medical) time series generation with recurrent conditional GANs. *CoRR* (2017). Available from: http://arxiv.org/abs/1706.02633.
19. Zhu F, Ye F, Fu Y, Liu Q, and Shen B. Electrocardiogram generation with a bidirectional LSTM-CNN generative adversarial network. *Sci Rep* (2019) 9:6734. doi:10.1038/s41598-019-42516-z
20. Wang Z, Chai J, and Xia S. Combining recurrent neural networks and adversarial training for human motion synthesis and control. *IEEE Trans Visualization Comput Graphics* (2019) 27:14–28. doi:10.1109/TVCG.2019.2938520
21. Zhou X, Pan Z, Hu G, Tang S, and Zhao C. Stock market prediction on high-frequency data using generative adversarial nets. *Math Probl Eng* (2018) 2018:1–11. doi:10.1155/2018/4907423
22. Wang JJ, Wang JZ, Zhang ZG, and Guo SP. Stock index forecasting based on a hybrid model. *Omega* (2012) 40:758–66. doi:10.1016/j.omega.2011.07.008
23. Huang W, Nakamori Y, and Wang S-Y. Forecasting stock market movement direction with support vector machine. *Comput Operations Res* (2005) 32:2513–22. doi:10.1016/j.cor.2004.03.016

24. Chong E, Han C, and Park FC. Deep learning networks for stock market analysis and prediction: methodology, data representations, and case studies. *Expert Syst Appl* (2017) 83:187–205. doi:10.1016/j.eswa.2017.04.030
25. Adebiyi AA, Adewumi AO, and Ayo CK. Comparison of ARIMA and artificial neural networks models for stock price prediction. *J Appl Math* (2014) 2014: 1–7. doi:10.1155/2014/614342
26. Nelson DM, Pereira AC, and de Oliveira RA. Stock market's price movement prediction with LSTM neural networks. In: International Joint Conference on Neural Networks (IJCNN); 2017 May 14–19; Anchorage, AK, USA. New York, US: IEEE (2017) p. 1419–26.
27. Loughran T, and McDonald B. The use of word lists in textual analysis. *J Behav Finance* (2015) 16:1–11. doi:10.1080/15427560.2015.1000335
28. Li FT, Ting KM, and Zhou ZH. Isolation-based anomaly detection. *ACM Trans Knowledge Discov Data (TKDD)* (2012) 6:3. doi:10.1145/2133360.2133363
29. Bracewell RN. *The Fourier transform and its applications*. New York: McGraw-Hill (1986).
30. Tsay RS. *Analysis of financial time series*. New York, US: John Wiley & Sons (2005) Vol. 543.
31. Hochreiter S, and Schmidhuber J. Long short-term memory. *Neural Comput* (1997) 9:1735–80. doi:10.1162/neco.1997.9.8.1735
32. Tieleman T, and Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks Machine Learn* (2012) 4:26–31.
33. Chollet F. *Keras* (2015). Available from: https://keras.io.
34. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, and Citro C. TensorFlow: large-scale machine learning on heterogeneous distributed systems (2015). Available from: tensorflow.org.
35. Cover T, and Hart P. Nearest neighbor pattern classification. *IEEE Trans Inform Theor* (1967). 13:21–7. doi:10.1109/tit.1967.1053964
36. Hyndman RJ, and Khandakar Y. Automatic time series forecasting: The forecast Package for R. *J Stat Soft* (2008) 27:1–22. doi:10.18637/jss.v027.i03
37. Smith TG. *pmdarima: ARIMA estimators for Python* (2017).
38. Clemen RT. Combining forecasts: A review and annotated bibliography. *Int J Forecast* (1989) 5:559–83. doi:10.1016/0169-2070(89)90012-5

Conflict of Interest: Author S-CC was employed by the company Kamakura Corporation.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.