RESEARCH ARTICLE

An ensemble of LSTM neural networks for high-frequency stock market classification

Svetlana Borovkova [ID] | Ioannis Tsiamas

School of Business and Economics, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

Correspondence
Svetlana Borovkova, School of Business and Economics, Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands.
Email: s.a.borovkova@vu.nl

Abstract

We propose an ensemble of long–short-term memory (LSTM) neural networks for intraday stock predictions, using a large variety of technical analysis indicators as network inputs. The proposed ensemble operates in an online way, weighting the individual models proportionally to their recent performance, which allows us to deal with possible nonstationarities in an innovative way. The performance of the models is measured by area under the curve of the receiver operating characteristic. We evaluate the predictive power of our model on several US large-cap stocks and benchmark it against lasso and ridge logistic classifiers. The proposed model is found to perform better than the benchmark models or equally weighted ensembles.

KEYWORDS

deep learning, ensemble models, high-frequency trading, LSTM neural networks

1 | INTRODUCTION

The long-lasting debate on predictability of financial markets has led to volumes of research on this subject, but no consensus has been reached. With the emergence and development of efficient machine learning algorithms and powerful computers, this debate has been reinvigorated in the last few years.

The original, and probably most important theory related to that debate is the efficient market hypothesis (EMH; Fama, 1970), whose core idea is that all available information is already incorporated into market prices and thus the prices reflect assets' true values. In other words, no individual can profit by making predictions of future prices, since future information is not yet available. In that sense, asset prices are not predictable.

Criticisms of the EMH are plentiful. One of the key assumptions of EMH—the rationality of agents operating in the markets—is often challenged. Behavioral finance scholars argue that there are times when even the collec-

tive actions of people (and certainly individual decisions) are irrational.

In practice, a well-established field of technical analysis—studying price patterns and inferring future price developments from these patterns—is a direct challenge to the EMH. With the increased availability of high-frequency trade data and the development of machine learning algorithms that can handle such large amounts of data, technical analysis is currently undergoing a revival: Daily patterns are replaced by intraday ones, and algorithms, not humans, now learn price patterns and make forecasts on the basis of them.

This is also the focus of our paper. We compile a large set of "features"—that is, technical analysis indicators (on the basis of intraday trading data)—and feed them into recurrent neural networks. We use the so-called *deep learning*, where not only contemporary but also previous patterns and prices are fed into the networks; this is achieved by using so-called *long–short-term memory networks* (LSTM networks). Contrary to most research on this subject, we

train not one but an ensemble of neural networks, and for forecasting they are weighted in real time, according to the recent forecasting performance (so the network that produced best forecasts recently is weighted most heavily). In this way, we can flexibly deal with possible nonstationarities in the data.

We apply our methodology to 22 large cap US stocks. We use 1 year of raw trade data, which were cleaned and aggregated into 5-minute intervals, amounting to roughly 19,000 observations per stock. Furthermore, for every stock we also include information about its primary competitor, thus establishing a universe of 44 stocks (for example, if we want to forecast the direction of GM stock, we also use features of Ford stock). We forecast price direction for 22 stocks, but use price features for all 44. This is done to maximally utilize the available information and to obtain robust forecasts.

We are interested in price direction forecasts, so at every moment each stock is labeled as "Buy" or "Sell," according to the price direction. By cross-sectional aggregation, we additionally created eight sector data sets. Our feature engineering is done by constructing a large number of technical indicators, on different time frames and on stock as well as sector level.

The first month of our data set was solely used for feature engineering. For the other 11 months, we operated in a rolling window way, where 1 month of data is used for training the networks, 1 week for validating their performance and the predictions are done for the following week. This amounts to 21 training–validation–testing periods per stock.

For every period and for each stock, we trained 12 stacked LSTM networks. The predictions of each model were evaluated by the area under the curve (AUC) score of the receiver operating characteristic (ROC; Kent, 1989), which we explain below. The ensemble predictions for each testing period were obtained by the weighted combination of the 12 trained models. The weights assigned to each model were proportional to their AUC score on the past week of predictions. Finally, the overall performance of our predictive framework is measured by the average AUC score of the ensembles for all 21 testing periods.

All data processing is done in Python 3, using the packages NumPy and pandas. Training LSTM networks is done in TensorFlow, while the lasso and ridge logistic regression models were trained using the scikit-learn package. No special hardware was employed: we used a PC with a two-core 2.3 GHz CPU and 8 GB RAM.

The rest of the paper is structured as follows. The next section briefly describes related works on machine learning methods for predictive modeling in financial markets. Section 3 describes data collection and feature engineering. Section 4 describes the methods used and Section 5 presents and discusses the results. The last section is dedicated to conclusions and future research.

2 | LITERATURE

Most of the research on machine learning and deep learning applications for financial time series predictions is quite recent. Some early works include that of Baestaens, Van Den Bergh, and Vaudrey (1995) and Refenes, Zapranis, and Francis (1994), who used simple artificial neural network (ANN) architectures and compared their performance to logistic regression (LR) models. Huanga, Nakamoria, and Wang (2005) found that support vector machines (SVMs) can achieve better results than traditional statistical methods, while Pai and Lin (2005) proposed a hybrid autoregressive integrated moving average–SVM method for price forecasting.

Later contributions include Hegazy, Soliman, and Salam (2013), who used particle swarm optimization[1] to fine-tune the hyperparameters of an SVM regressor, achieving significantly smaller mean squared errors (MSEs) on several US stocks. Nelson, Pereira, and de Oliveira (2017) trained LSTM networks on 15-minute-interval observations, for several BOVESPA (Sao Paolo Stock Exchange) stocks, and reported accuracy metrics of 53–55% for the next direction price forecasts. Fischer and Krauss (2017) conducted a large-scale research project using daily S&P 500 data from 1992 to 2015. They used LSTMs, random forests, deep networks, and logistic regression, and found that a trading strategy based on the predictions of LSTM was the most profitable.

Qu and Zhang (2016), assuming that high-frequency returns periodically trigger momentum and reversal, designed a new SVM kernel method[2] for forecasting high-frequency market directions and applied their method to the Chinese CSI 300 index. Their results were significantly better compared to the radial basis function (RBF)[3] kernel and the sigmoid kernel.

One of the most remarkable contributions to deep learning for stock price prediction is that by Bao, Yue, and Rao (2017). They proposed a predictive framework composed of three parts. First, they apply a wavelet transformation (WT)[4] to the financial data set (prices, technical indicators,

---

[1] A nature-inspired algorithm that uses a collection of candidate solutions as well as their respective positions and velocities to find the optimal solutions (Kennedy & Eberhart, 1995).
[2] Kernel functions enable higher dimensional operations without operating in the actual higher dimensional space.
[3] RBF maps inputs to higher dimensional space using the Euclidean distance.
[4] A signal processing technique which transforms a series from time to frequency domain.

macroeconomic data). Then the de-noised data are passed through stacked autoencoders[5] to generate meaningful features, and finally LSTM networks are used for forecasting. They used daily observations from six stock indices and tested the profitability of their models by purchasing corresponding future indices, according to the predictions of the model. They showed that their proposed framework was better in terms of both accuracy and predictability when compared to simpler methods.

Another interesting work is that of Rechenthin (2014). He demonstrates the existence of predictability in high-frequency prices by analyzing price patterns. His proposed framework involves creating a pool of thousands of (relatively simple) base classifiers (SVMs, ANNs, decision trees), and interchanging between models depending on their performance. The target labels in this research are "strong sell," "hold," or "strong buy." He reported high predictive accuracy for 34 US energy stocks on 1-minute intervals, in the last 7 months of 2012.

The main distinguishing features of this research, when compared to the contributions outlined above, are that we

- do extensive feature engineering, using insights from technical analysis, and use these features as networks' inputs (including also features of competitor stocks);
- use deep learning—that is, LSTM networks—utilizing all the previous price information;
- train an ensemble of networks and dynamically weight their forecasts according to recent forecasting performance.

3 | DATA AND FEATURES ENGINEERING

We use 1 year (2014) of high-frequency historical data, for 22 large-cap[6] US stocks, traded either on NYSE or NASDAQ. For robustness of our forecasts, we include the primary competitor for each of the 22 stocks, taking advantage of strong cross-correlations between them. These stocks belong to one of the eight major sectors of the US stock markets,[7] so we also construct sector data sets, by aggregating the stocks cross-sectionally. A summary of the selected stocks is presented in Table A1 in Appendix A.

Our choice of using large-cap US stocks is motivated by two factors: lots of data and high liquidity. The abundance of data that characterizes US large-cap stocks, even in small time intervals, is essential for training our models, since a shortage or absence of trading data can disrupt

the learning process. Very important also is the liquidity of a stock, as high liquidity ensures low trading costs. We selected the stocks in such a way to keep the ratio of the market capitalization of each sector close to that in S&P 500. We decided to avoid mega-cap stocks such as Google and Apple, since otherwise the technology sector would dominate the other sectors by far.

Data were obtained from the NYSE Trade-and-Quote (TAQ) database, which is accessible through the WRDS[8] interface. The TAQ database contains raw trade information for US stocks at microsecond accuracy. We consider only trades that occurred between the regular opening hours of NYSE and NASDAQ—more specifically, between 09:30 and 16:00.

Raw trade data contain the date and time that the trade took place, the price of the trade, and its size (in stocks). Of importance are, additionally, the correction indicator and the trade conditions of the trade. A correction indicator can take several values, with 0 and 1 meaning that the trade was valid. Other values indicate that there was a complication with the trade and it was canceled. The trade condition specifies the existence and nature of special conditions, under which the trade took place. The vast majority of trades are completed under normal conditions or are intermarket sweep orders.[9]

We processed the raw data to clean them from "bad" trades and aggregate them into 5-minute intervals. "Bad" trades are those that are either out of sequence, outside of normal trade hours, corrected, or completed under special conditions. The removal of these trades ensures that data contain the least amount of noise. Thus, during the aggregation process, we took into account only trades that were correct, were completed under regular conditions, or were intermarket sweep orders. For each stock, 5–10% of the total trades were cleaned.

Furthermore, during data aggregation, we constructed some basic features such as price differences and trading volumes. An overview of these features is presented in Table A2 in Appendix A. In addition, we did not consider early closing days, in order to maintain a consistent number of observations per day (78). Prices were also adjusted for stock splits. After aggregation, the total of 18,252 observations per stock were produced.

We also constructed sector data sets and an overall data set, using the basic features of the 44 stocks. Market capitalization for each stock was used as weights within sectors.

---

[5]A class of neural networks which learn a representation of input x to a latent space $\mathcal{Z}$ and then reconstruct x_0, similarly to x (Bengio, 2009).
[6]More than $10 billion in market capitalization.
[7]According to market capitalization: Capital Goods, Consumer Non-Durables, Consumer Services, Energy, Finance, Health Care, Public Utilities and Technology.

[8]Wharton Research Data Services (https://wrds-web.wharton.upenn.edu/).
[9]These allow a trader to purchase stock from several exchanges until the given order size is satisfied.
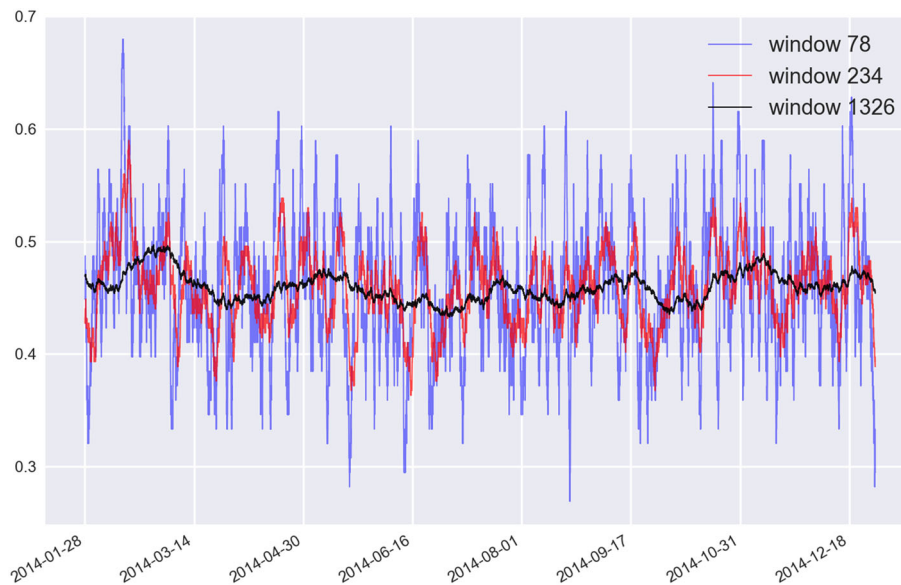
FIGURE 1 Probability distribution of class "Buy" [Colour figure can be viewed at wileyonlinelibrary.com]

Using the weighting factors, we constructed the basic features presented in Table A2 on sector and universe level. For example, price-related features were calculated as the weighted average of individual stock prices, while volume-related features were calculated by summing the individual volumes.

The objective of this exercise is to provide, at every instance, probabilities of an upward or downward move, regarding the price direction in the next 5 minutes. Thus periods of positive returns are labeled as "Buy," whereas periods with zero or negative returns are labeled as "Sell."

In a high-frequency setting, a large number of time intervals are characterized by zero returns, thus causing the class "Sell" to be overpopulated in comparison to the class "Buy." This phenomenon is called *class imbalance* and it may affect the training procedure if not treated. Every stock experiences an average ratio of 46% Buy and 54% Sell. In Figure 1 we provide an illustration of the class "Buy" probabilities for stock ABT, considering rolling windows of 78, 234, and 1,326 past observations. Given the fast-shifting probability distributions of our target variable, it becomes clear how important the use of multiple models is, in order to capture this variability.

To train the networks, we constructed a large number of technical indicators. Technical indicators are measures used in technical analysis that aim to identify "Buy" or "Sell" signals in charts and graphical representations of stocks. They can be categorized in trend, volatility, volume, and momentum indicators. People have been trading on the basis of these indicators by using heuristic rules. Although their true value has been argued, a model should be able to benefit from them, since they can provide a "summary" of a stock price development on differ-

ent time scales. Using technical indicators as features in machine learning is known as quantitative technical analysis. The indicators used in this research, along with their formulas, are presented in Table A3 in Appendix A and in Appendix B.

Aside from technical indicators, we used price predictions from rolling regressions as features based on the percentage change in prices (P_{close}, P_{high}, P_{low}, P_{vwap}). More specifically, we applied AR(1) models with a rolling window of 1,326 (approximately 1 month of observations).

Furthermore, the probability distributions and conditional probability distributions of target class "Buy" were included as features. We used rolling windows of 36, 72, 234, and 1,326 observations, for probabilities $\mathcal{P}(y_t = \text{Buy})$ and 72, 234, and 1,326 observations, for conditional probabilities $\mathcal{P}(y_t = \text{Buy}|y_{t-1} = \text{Sell})$, $\mathcal{P}(y_t = \text{Buy}|y_{t-1} = \text{Buy})$. Certain trends can be observed along the span of 1 year, which can potentially enhance our models' predictive power.

Trading intensity can vary greatly in stock markets, especially in a high-frequency setting. There are specific parts of the day that are more trading intensive than other. Figure 2 indicates the presence of time-specific patterns during a trading day. For that reason we included dummies that indicate the specific time of each trade (minutes, hours, days).

4 | METHODOLOGY

4.1 | Stacked LSTM network with layer normalization

Recurrent neural networks (RNNs) have been effectively used to forecast sequential, possibly nonstationary data.
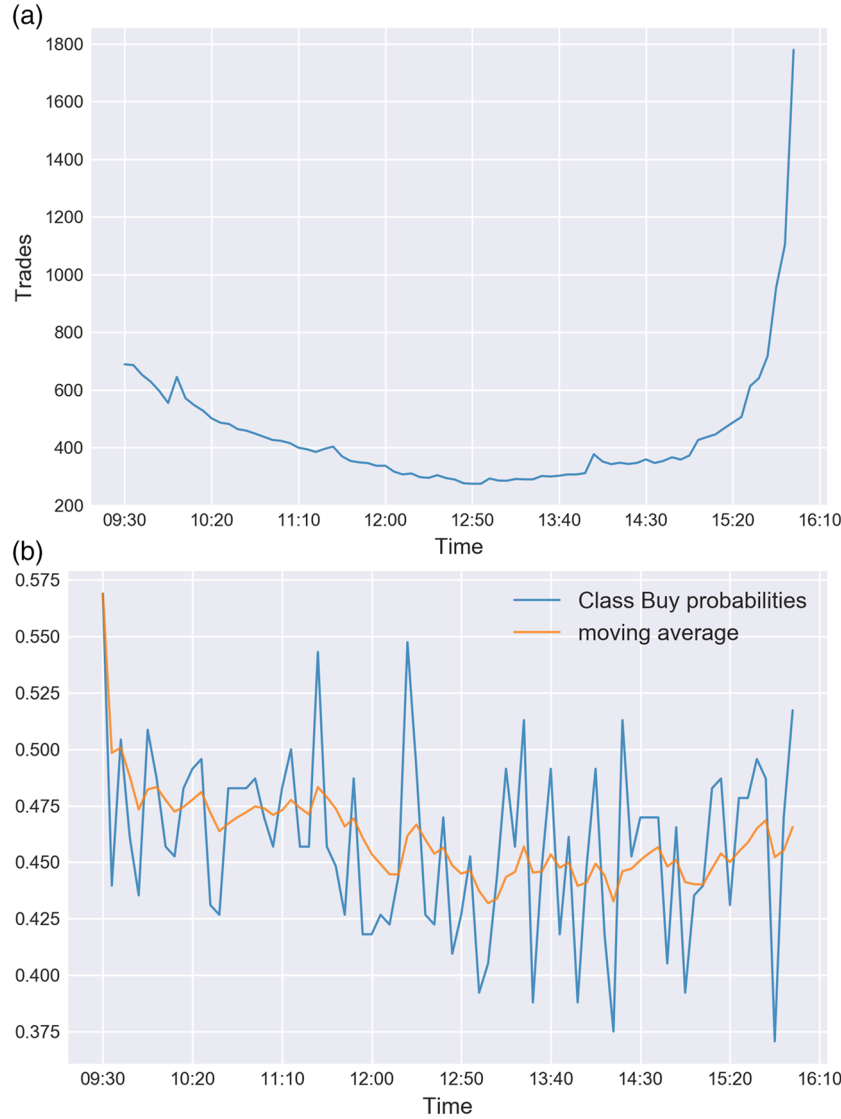
(a)

(b)

FIGURE 2 (a) Average number of trades during the day. A clear trend can be observed throughout the day, with trade intensity decreasing until 13:00 and then increasing exponentially when approaching closing time. The irregularity at 10:00 happens due to the "10 o'clock rule," according to which most traders avoid the first 30 minutes of the day due to highly volatile price movements. (b) Class "Buy" probabilities throughout the day, measured by taking into account all 44 stocks. We can identify a decreasing trend until 13:00, which is followed by an increasing one [Colour figure can be viewed at wileyonlinelibrary.com]

They have the ability to "remember" past observations, by maintaining a cell state c that gets updated at every time step t. Their mathematical configuration is fairly simple, with function f, connecting the cell states and function ϕ used to calculate the output h:

$$c^{(t)} = f(\mathbf{U}x^{(t)} + \mathbf{W}c^{(t-1)} + \mathbf{b}_c), \quad h^{(t)} = \phi(\mathbf{V}c^{(t)} + \mathbf{b}_o), \quad (1)$$

where $x^{(t)}$ is the input at time t, $\mathbf{U}$, $\mathbf{W}$ are weight matrices and $\mathbf{b}_c$, $\mathbf{b}_o$ are bias vectors.

The main problem of the vanilla RNN is that, during training, the error signals coming from past observations become weaker and weaker, and thus long-term dependencies cannot be modeled efficiently. The primary reason for the weak signals is the derivative of the cell states with respect to previous cell states. If $|\frac{\partial c_t}{\partial c_{t-1}}| < 1$ for consecutive states, the derivatives of the long-term states become really small, approaching zero:

$$\frac{\partial c_t}{\partial c_{t-j}} = \frac{\partial c_t}{\partial c_{t-1}} \cdot \frac{\partial c_{t-1}}{\partial c_{t-2}} \cdots \frac{\partial c_{t-j+1}}{\partial c_{t-j}} \approx 0. \quad (2)$$

The LSTM (Hochreiter & Schmidhuber, 1997) network is a particular class of RNNs which solves this issue by connecting the cell states in a linear way, thus ensuring strong and stable gradients. Another difference of the LSTM compared to the vanilla RNN is the use of multiple sigmoid gates that control the information flow through the model. Layer normalization (Lei Ba, Kiros, & Hinton, 2016) is
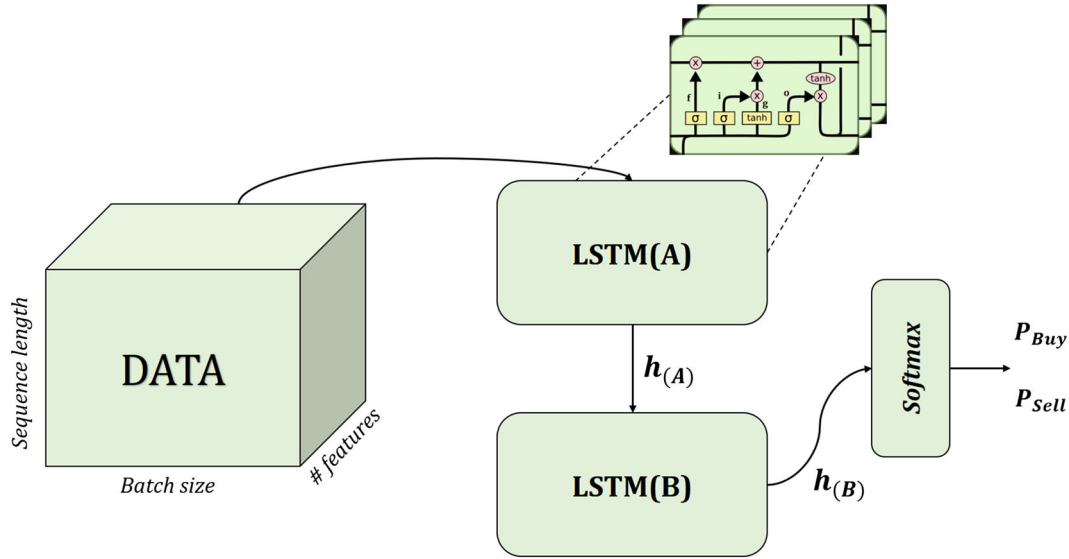
FIGURE 3 Stacked-LSTM with a softmax layer. The "DATA" block represents the data as it is fed into LSTM(A), in the form of a three-dimensional tensor, with dimensions batch size, sequence length and number of features. The output h(A) of the LSTM(A) is passed into LSTM(B) and consequently its output, h(B) is transformed into probabilities of "Buy" and "Sell" class through the softmax layer, for each instance of the batch. The subfigure (source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/), located above LSTM(A), provides a visual representation of the flow of information inside an LSTM network per time-step. Note that the subfigure represents a normal version of the LSTM model, while the implemented one includes layer normalization, as well as dropout [Colour figure can be viewed at wileyonlinelibrary.com]

also introduced in our model to prevent neurons from saturating,[10] by keeping their inputs centered. It works by calculating the mean μ and standard deviation σ of the inputs across the time dimension and normalizing them accordingly. The following equations describe the LSTM with layer normalization:

$$g^{(t)} = \tanh[\text{LayerNorm}(\mathbf{W}_{gx}\mathbf{x}^{(t)}; \boldsymbol{\alpha}_g, \boldsymbol{\beta}_g) + \text{LayerNorm}(\mathbf{W}_{gh}\mathbf{h}^{(t-1)}; \boldsymbol{\alpha}_g, \boldsymbol{\beta}_g)], \quad (3)$$

$$i^{(t)} = \text{sigm}[\text{LayerNorm}(\mathbf{W}_{ix}\mathbf{x}^{(t)}; \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i) + \text{LayerNorm}(\mathbf{W}_{ih}\mathbf{h}^{(t-1)}; \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i)], \quad (4)$$

$$f^{(t)} = \text{sigm}[\text{LayerNorm}(\mathbf{W}_{fx}\mathbf{x}^{(t)}; \boldsymbol{\alpha}_f, \boldsymbol{\beta}_f) + \text{LayerNorm}(\mathbf{W}_{fh}\mathbf{h}^{(t-1)}; \boldsymbol{\alpha}_f, \boldsymbol{\beta}_f)], \quad (5)$$

$$o^{(t)} = sigm[\text{LayerNorm}(\mathbf{W}_{ox}\mathbf{x}^{(t)}; \boldsymbol{\alpha}_o, \boldsymbol{\beta}_o) + \text{LayerNorm}(\mathbf{W}_{oh}\mathbf{h}^{(t-1)}; \boldsymbol{\alpha}_o, \boldsymbol{\beta}_o)], \quad (6)$$

$$c^{(t)} = g^{(t)} \odot i^{(t)} + c^{(t-1)} \odot f^{(t)}, \quad (7)$$

$$h^{(t)} = \tanh[\text{LayerNorm}(c^{(t)}; \boldsymbol{\alpha}_c, \boldsymbol{\beta}_c)] \odot o^{(t)}, \quad (8)$$

where $\odot$ denotes the element-wise multiplication operator, $\mathbf{W}_{*x}$ input weight matrices, $\mathbf{W}_{*h}$ recurrent weight matrices and $\boldsymbol{\alpha}_*$ (gain), $\boldsymbol{\beta}_*$ are trainable vectors of the layer normalization, used to fit the distribution of the output. Furthermore, tanh is the hyperbolic tangent function,

sigm is the sigmoid function, and LayerNorm is the layer normalization function:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \text{sigm}(x) = \frac{1}{1 + e^{-x}}, \quad (9)$$

$$\text{LayerNorm}(\mathbf{Z}; \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\mathbf{Z} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \odot \boldsymbol{\alpha} + \boldsymbol{\beta}. \quad (10)$$

Equations 3–6 refer to the four basic gates of the LSTM network, which are the modulation gate $g^{(t)}$, the input gate $i^{(t)}$, the forget gate $f^{(t)}$, and the output gate $o^{(t)}$. The modulation gate transforms the previous state and the new input via a tanh activation function and represents the new candidate values that could be added to the cell state. The input gate weights the new candidate values coming from the modulation gate, before being added to the cell state. The sigmoid function, with outputs in $[0, 1]$, provides weight according to the relevance of the information. The forget gate serves the cause of filtering the current information contained in the cell state. An output of 0 from the sigmoid function means that the model should forget everything, whereas an output of 1 means that it should retain this information in its entirety. Lastly, the output gate is of similar function to the other two sigmoids. It highlights the information that will be passed to the next hidden state. Equations 7 and 8 are then used to compute the output of the LSTM layer. The choice of not including bias terms in the equations of the four gates

---

[10]The result of extreme compression of the input between bounds $[-1, 1]$ for tanh and $[0, 1]$ for σ is small derivatives, a saturated neuron, which passes a really weak error signal and does not contribute to training.

(Equations 3–6) is motivated by the use of layer normalization. Although not mentioned in Lei Ba, et al. (2016), the function of the bias terms in the LSTM is taken over by the shift terms β_* of LayerNorm and thus their presence is considered redundant.

In this research we implemented a stacked-LSTM, comprising two LSTM networks (A) and (B), illustrated in Figure 3. The two LSTMs are of different size, with the first one being larger than the one that follows it. This is a common practice, aiming to identify different features in terms of specificity. The first layer is able to recognize more general features, while the second one targets more specific ones. For example, in the computer vision task of animal classification, first layers detect features like edges and lines, while the latter ones detect more specific ones, like eyes or ears. In our implementation, the respective sizes of the two LSTMs were set, after tuning, to 64 and 32.

The output of LSTM(A) is fed to LSTM(B) and accordingly its output is passed through a softmax layer that transforms the raw output to class probabilities:

$$p^{(t)} = \text{softmax}(h_{(B)}^{(t)}\boldsymbol{W}_{\text{softmax}} + \boldsymbol{b}_{\text{softmax}}), \qquad (11)$$

$$\text{softmax}(z_{buy}) = \frac{e^{z_{buy}}}{e^{z_{buy}} + e^{z_{sell}}}. \qquad (12)$$

4.2 | Training

The cross-entropy loss function is used to measure the divergence of the probability estimates output by the softmax layer with respect to the true labels "Sell" and "Buy." Note that, only during training (not during inference), we additionally weight the $h_{(B)}^{(t)}$ to account for the class imbalance[11] observed in our data, resulting in $p_{\text{weighted}}^{(t)}$. Thus the loss function for a batch size of d_1, sequence length of d_2, class label y and regularization term $\mathcal{L}_{\text{reg}}$, the loss function is the following:

$$\mathcal{L}(\boldsymbol{Y}, \boldsymbol{P}_{\text{weighted}}) = -\frac{1}{d_1 d_2}\sum_{i=1}^{d_1}\sum_{t=1}^{d_2}\sum_{k\in[\text{Sell,Buy}]} y_k^{(i,t)} \\ \times \log(p_{k,\text{weighted}}^{(i,t)}) + \mathcal{L}_{\text{reg}}. \qquad (13)$$

The regularization term used in this research is the $\ell2$ weight decay, which employs the Euclidean norm of the regularized parameters to penalize complex weights and avoid overfitting. For the collection of regularized parameters Θ, which can be found in Table 1, and a regularization magnitude λ, which was tuned to 0.1, we have

$$\mathcal{L}_{\text{reg}} = \frac{\lambda}{d_1 d_2}\sum_{\theta} \in \Theta||\theta||_2. \qquad (14)$$

The choices for batch size $d1$ and sequence length $d2$ were 32 and 5 accordingly. Using a batch size of 32 is suggested by Bengio (2012), as it is usually large enough to take advantage of the computational speed-up of matrix multiplications. On the other hand, we found that a sequence length of 5 was the best choice, since the extra computational complexity of larger sequences was not justified by the small gains.

The Glorot initialization (Glorot & Bengio, 2010) is used for sigmoid layers, to ensure the perseverance of the signal variance, through both forward propagation and backpropagation. In Table 1 we provide an overview of the model parameters, along with their dimensions, initialization and regularization.

Except for weight decay, dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is also used as a measure to counter overfitting and increase the generality of our model. Although dropout may also be applied to recurrent layers (Semeniuta, Severyn, & Barth, 2016), we found that it was decreasing the memory capacity of the network. Thus dropout is applied only to nonrecurrent layers, namely before data enter the first LSTM cell and before it enters the softmax layer.

Features are split into two subgroups at the beginning of each training iteration: the stock and the nonstock[12] features. We biased the information flow through the model towards stock features, by applying a softer dropout rate on them. More specifically, 50% is applied to stock features, while 70% is applied to the rest. The two data streams are concatenated before entering the first LSTM cell. Finally, a dropout rate of 50% is applied before the softmax layer. The dropout rates where selected via means of trial-and-error experiments.

As mentioned earlier, we engineer a great number of features for each time step t. These can be categorized in the data sets of stock, competitor, sector, universe, correlated stocks, and time dummies, and amount to a total of 400 features. Feeding all these to the model would cause a significant decrease in training time and most probably disrupt the learning process. Unfortunately, feature selection algorithms are too computationally expensive and time consuming, especially if they were to be applied hundreds of times, considering the great number of different networks that we are training in this research We therefore devised a procedure to randomize the data feed for each model, resulting in a different training set every time. This contributes to the variability needed to produce an efficient ensemble model by combining the different stacked LSTMs. Similarly to dropout, we biased the feature selection towards stock-specific features, and let a probabilistic

---

[11]Recall that, in high-frequency settings, the sell class usually dominates the buy class with a ratio of approximately 54%:46%.

[12]Competitor, sector, correlated stocks, time features.

TABLE 1 All trainable parameters, along with their dimensions, initialization method, and regularization

Parameter	Dimensions	Initialization	Regularized
$W_{gx}^{(A)}, W_{ix}^{(A)}, W_{fx}^{(A)}, W_{ox}^{(A)}$	250×64	$\mathcal{N}(0, \sqrt{\frac{2}{250+64}})$	True
$W_{gh}^{(A)}, W_{ih}^{(A)}, W_{fh}^{(A)}, W_{oh}^{(A)}$	64×64	$\mathcal{N}(0, \sqrt{\frac{2}{64+64}})$	True
$W_{gx}^{(B)}, W_{ix}^{(B)}, W_{fx}^{(B)}, W_{ox}^{(B)}$	64×32	$\mathcal{N}(0, \sqrt{\frac{2}{64+32}})$	True
$W_{gh}^{(B)}, W_{ih}^{(B)}, W_{fh}^{(B)}, W_{oh}^{(B)}$	32×32	$\mathcal{N}(0, \sqrt{\frac{2}{32+32}})$	True
$\alpha_g^{(A)}, \alpha_i^{(A)}, \alpha_f^{(A)}, \alpha_o^{(A)}, \alpha_c^{(A)}$	64×1	$\mathbf{1}$	True
$\beta_g^{(A)}, \beta_i^{(A)}, \beta_f^{(A)}, \beta_o^{(A)}, \beta_c^{(A)}$	64×1	$\mathbf{0}$	False
$\alpha_g^{(B)}, \alpha_i^{(B)}, \alpha_f^{(B)}, \alpha_o^{(B)}, \alpha_c^{(B)}$	32×1	$\mathbf{1}$	True
$\beta_g^{(B)}, \beta_i^{(B)}, \beta_f^{(B)}, \beta_o^{(B)}, \beta_c^{(B)}$	32×1	$\mathbf{0}$	False
W_{softmax}	32×2	$\mathcal{N}(0, 0.01)$	True
b_{softmax}	1×2	$\mathbf{0}$	False
Trainable parameters = 93, 698			

Note. (A) indicates the first LSTM and (B) the second one. The first size of the input weight matrices of LSTM(A) indicates the number of features fed into the network. We assume that the number of features is 250, while it may slightly vary from model to model since we randomized the procedure.

algorithm handle the selection of features from within the other data sets, aiming to cover all the different look-back windows used to produce the features. As a result, approximately 250 features are used for every model.

The training algorithm used in this research is the RMSProp (Hinton, Srivastava, & Swersky, 2012) with momentum. RMSProp works with adaptive learning, by maintaining a moving average of the square of the gradient for each parameter, which is used to normalize the gradients. For a decay rate d and for gradients $G^{(t)}$, the update rule for the moving average of the squared gradient (MASG) is defined as

$$\mathbf{MASG}^{(t)} = d\,\mathbf{MASG}^{(t-1)} + (1 - d)(G^{(t)})^2. \quad (15)$$

Now, using also a momentum of m and learning rate η, we can define the update rule for parameters θ as

$$\theta^{(t+1)} = \theta^{(t)} - mG^{(t-1)} + \eta\frac{G^{(t)}}{\sqrt{\mathbf{MASG}^{(t)} + \epsilon}}, \quad (16)$$

where ϵ is a small positive value to avoid division by zero, set to 10^{-8}. The rest of the parameters of the RMSProp were set by means of trial and error. We used an initial learning rate of $\eta = 0.001$, a decay rate $d = 0.9$, and a momentum of $m = 0.9$.

As discussed in Section 4.1, although the LSTM solves the problem of the vanishing gradient, it does not address the issue of the exploding gradient. The problem of the exploding gradient arises in the backpropagation through time, when sequential error signals from the state derivatives are much greater than 1. Accordingly, their product creates a large gradient that can potentially lead to unsta-

ble training. The solution for this problem is to define a threshold $G_{\max}$, above which, all gradients are reduced to it.

$$G^{(t)} = \begin{cases} G_{\max}, & G^{(t)} > G_{\max} \\ G^{(t)}, & G^{(t)} \leq G_{\max}. \end{cases}$$

Although, by observing the training process, we found that gradients were rarely greater than 1, we set a maximum gradient $G_{\max} = 5$, to ensure stable training.

4.3 | Evaluation

Evaluation of the performance of each model is done by the area under the curve (AUC) score. AUC is calculated as the area defined by the receiver operating characteristic (ROC). The ROC curve is obtained by evaluating the true positive rate (TPR) and false positive rate (FPR) at different decision thresholds.

Using the probabilities that were output by the softmax layer in our model, and given a threshold parameter t, we can define TPR and FPR as follows:

$$\text{TPR}(t) = \int_t^1 f_1(x)dx, \quad \text{FPR}(t) = \int_t^1 f_0(x)dx, \quad (17)$$

where $f_1(x)$ and $f_0(x)$ are the probability density functions of x if it belongs to class "Buy" or "Sell", respectively. The density functions and their intersections, as defined by the varying threshold t, can be observed in the example of Figure 4.

By plotting the TPR against FPR, for the varying decision threshold t, the ROC curve is obtained and the AUC
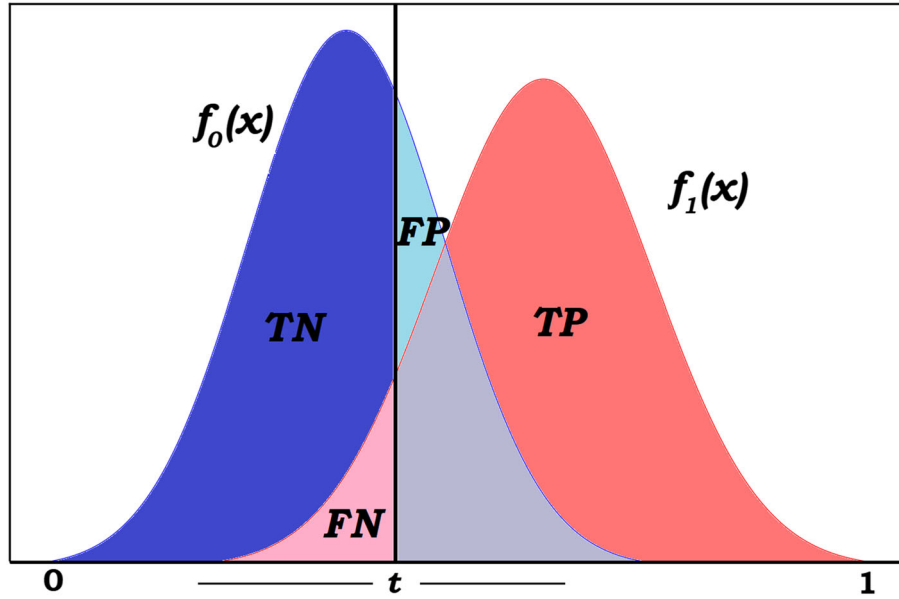
FIGURE 4 False positive rate and true positive rate. By varying threshold t over the range $[0, 1]$, the FPR is the ratio of the area covered by FP to the area defined by $f_0(x)$. On the other hand, the TPR is the ratio of the area covered by TP to the area defined by $f_1(x)$. Note that the gray area is part of both FP and TP. Also, it has to be noted that this example is not something we derived from our methods. In our case, given its difficulty, the density functions are most probably blended with one another and are not clearly separable, like those in this example [Colour figure can be viewed at wileyonlinelibrary.com]

score is the area defined by this curve. This translates to the following integration:

$$\text{AUC} = \int_0^1 \text{TPR}(t)(-\text{FPR}'(t))dt, \qquad (18)$$

where the prime indicates the reversal of the integration limits. The AUC score is the probability that a model ranks a random chosen "Buy" example higher than a random chosen "Sell" example. Being a probability, the AUC score is in the range of $[0, 1]$, with a probability of 0.5 indicating a random model.

AUC is arguably the most used evaluation metric when it comes to classification tasks. One of its advantages over other evaluation metrics is its robustness to class imbalances. Class imbalance, in a classification task, is the problem of one class being the majority of the total population. As previously discussed, in high-frequency market classification the positive class is systematically lower than 50% of total instances—usually around 46%. Evaluating performance using plain accuracy would not make clear what the actual predictive power of the model is, since it assumes that all misclassification errors are the same. An accuracy score of 54% could just be that the model predicts the majority of instances as "Sell," which obviously does not add any real value, and furthermore cannot be used as a means of model comparison.

Other measures, such as sensitivity or precision, or their harmonic mean, the f1-score, may be used in order to tackle class imbalance and provide a better view on the predictive power of a model. Yet again, there are some shortcomings. Firstly, the true negatives (TN) are left out of the equation. Secondly, these measures still evaluate performance over one decision threshold $t = t_0$.

The AUC score is robust to both class imbalance and decision thresholds, and thus it is an unbiased performance metric that can be used to capture in full the performance of our models. Furthermore, in high-difficulty classification tasks, like ours, it is a benchmark by itself. The weak form of market efficiency suggests that past market information (like technical indicators) hold no predictability power over future events. According to this statement, predictions of a classifier should approach random performance, meaning an AUC score of 0.5. Thus any classifier that provides AUC scores greater than 0.5 would be evidence of predictability.

4.4 | Ensemble

The final stage of our predictive framework involves the use of an ensemble algorithm to combine the predictions of 12 LSTMs, for every predictive period. A simple way to implement an ensemble is to take the average of the individual predictions. Thus for stock S and for instance t, the prediction of the ensemble is

$$p_{S,t}^{(\text{ensemble})} = \frac{1}{12} \sum_{i=1}^{12} p_{S,t}^{(\text{model}_i)}, \qquad (19)$$

where $p_{S,t}^{(\text{model}_i)}$ is the probability for class "Buy" as was output by the softmax layer of model$_i$. For the rest of this research, we are referring to this model as the equally weighted ensemble.

Except for the equally weighted ensemble, we considered two additional methods. The first one is an ensemble method, where the weights of each model in the ensemble change according to the performance of the model in the n_{valid} past observations. Thus, after training the models on the n_{train} first observations of the data set, the process works in an online way, by evaluating the models and then predicting the next observation:

$$p_{S,t}^{(\text{ensemble})} = \sum_{i=1}^{12} \frac{\text{AUC}_{S,t}^{(\text{model}_i)}}{\sum_{k=1}^{12} \text{AUC}_{S,t}^{(\text{model}_k)}} p_{S,t}^{(\text{model}_i)}, \qquad (20)$$

where $\text{AUC}_{S,t}^{(\text{model}_i)}$ indicates the AUC score of model$_i$ in period $(t - n_{\text{valid}} - 1, t - 1)$. This is illustrated in Figure 5. This type of ensemble will be referred to as the performance weighted ensemble.

To identify whether predictability comes from a single good model or from their collective interactions, we furthermore considered an ensemble, which takes into account only the best model, according to its performance in the last n_{valid} observations, and will be referred to as best model ensemble. Essentially the best model ensemble is an extreme case of performance weighted ensemble, where the weight is 1 for the model with the highest AUC score and 0 for all the rest.

$$p_{S,t}^{(\text{ensemble})} = \underset{\text{AUC}_{S,t}}{\arg\max}, \quad \boldsymbol{p}_{S,t}. \qquad (21)$$

Our intuition is that each one of the 12 models has identified and learned certain patterns from the training set. Whether these patterns can be found also in the test set or have disappeared, we are not in a position to know. But what we know is that if they emerge they are going to be preserved at least for some period T. Thus the ensemble

can work in an online way and learn to update the weighting of the models according to their ability to identify some good patterns in the last n_{valid} instances.

As a look-back window for validating the models, we chose $n_{\text{valid}} = 390$, which corresponds to 1 week of data. Given that predictions are applied for 2 weeks, 1 week provides enough time to validate and identify the appropriate weights, while the remaining week can be used for forecasting. Thus, for a 4-week period, the first 2 weeks are used for training and predictions are made on the last week.

(i) Generate data sets $X_i \in \mathscr{X}$, with $i \in [1, 12]$. All data sets refer to the same time period as $\mathscr{X}$, but the feature selection is randomized. Approximately 250 features, from the available 600, are selected for each X_i. Data sets X_i are split into $X_i^{(\text{train})}$ and $X_i^{(\text{test})}$.

(ii) Train models $M_i, i \in [1, 12]$ using data sets $X_i^{(\text{train})}$. The fact that each training set is different from one another and the huge number of parameters (roughly 90,000) that have to be tuned by the RMSprop optimizer ensures that the resulting trained models will highly dissimilar. This is a critical requirement for the ensemble to work, since it will benefit from the diversity within the models and their respective predictions.

(iii) Models M_i performance is evaluated on the n_{valid} first observations of $\mathscr{X}^{(\text{test})}$ and the $\text{AUC}_i^{(n_{\text{valid}}+1)}$ is calculated.

(iv) The weighting factors $W_i^{(n_{\text{valid}}+1)}$ of each model M_i in the ensemble are calculated. As indicated in Equation 20, the weight of each model for predicting $\mathscr{Y}_{n_{\text{valid}}+1}^{(\text{test})}$ is a fraction of $\text{AUC}_i^{(n_{\text{valid}}+1)}$ to the summation of the 12 AUC scores calculated for the exact same period.

(v) The probabilities for "Buy" and "Sell" classes are estimated. Each model M_i estimates the probabilities of observation $\mathscr{Y}_{n_{\text{valid}}+1}^{(\text{test})}$, thus $\mathcal{P}_i$. Then the prediction of the ensemble is produced by the weighted sum of these probabilities, using the weighting factors of the previous step.
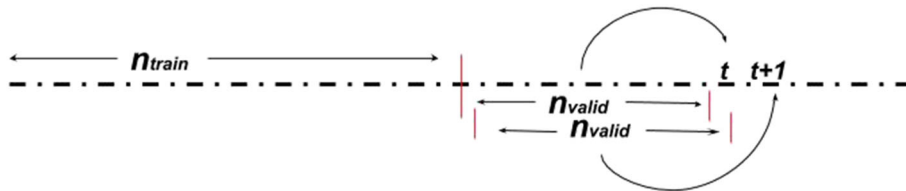

FIGURE 5 Predictions in an online way. The performance-weighted ensemble and best model ensemble work in an online way, by evaluating the 12 models on n_{valid} and adjusting the weights of the models for predicting the next observation [Colour figure can be viewed at wileyonlinelibrary.com]

Steps (iii)–(v) are repeated for every observation in $[n_{\text{valid}} + 1, n_{[(i)]\text{valid}} + n_{\text{test}}]$, with performance of models being evaluated on the n_{valid} previous observations of the observation that we want to predict and then weighting their predictions for the point accordingly. This process is illustrated in Figure 6.

4.5 | Benchmarks

To provide a benchmark for the performance of the ensemble, we used two additional methods: the lasso and the ridge classifiers. Essentially they are regularized logistic regressions, with $\ell 1$ weight decay applied for lasso and $\ell 2$ applied for ridge. The selection of these methods for benchmarking stems from the fact that they are simple (compared to LSTMs) and, furthermore, they output probabilities that can be used to calculate the AUC scores. The

TABLE 2 AUC scores for the LSTM ensembles. Average AUC scores for the 22 stocks

Stock	Equally weighted ensemble	Best model ensemble	Performance-weighted ensemble
BA	0.5153 (4.13)***	0.5017 (0.30)	**0.5224** (4.24)***
F	0.5232 (4.70)***	0.5179 (2.95)***	**0.5355** (5.86)***
DHR	0.5144 (2.77)***	0.5141 (2.14)**	**0.5231** (3.52)***
KO	0.5173 (4.20)***	0.5138 (2.41)**	**0.5251** (3.22)***
MO	0.5110 (2.61)***	0.5042 (0.71)	**0.5201** (3.08)***
MAR	0.5251 (7.17)***	0.5346 (4.29)***	**0.5346** (5.30)***
AMT	0.5086 (1.72)**	0.4959 (−0.53)	**0.5132** (1.99)**
MCD	0.5223 (3.68)***	0.5222 (3.52)***	**0.5248** (3.62)***
DIS	0.5150 (5.03)***	0.5185 (2.72)***	**0.5196** (3.09)***
GE	0.5179 (3.51)***	0.5129 (2.09)**	**0.5251** (3.93)***
EOG	0.5155 (3.48)***	0.5082 (1.15)	**0.5193** (2.96)***
GS	0.5149 (3.28)***	0.5194 (3.28)***	**0.5362** (5.38)***
MET	0.5178 (3.33)***	0.5090 (1.30)	**0.5218** (3.47)***
BK	0.5173 (4.78)***	0.5102 (1.76)**	**0.5254** (3.92)***
AMGN	0.5161 (2.88)***	0.5094 (2.06)**	**0.5224** (2.91)***
ABT	0.5057 (1.55)*	0.5111 (2.07)**	**0.5190** (3.12)***
AET	0.5133 (2.95)***	0.5255 (5.40)***	**0.5263** (3.25)***
NEE	0.5207 (4.73)***	0.5103 (1.47)*	**0.5304** (4.76)***
EXC	0.5102 (2.23)**	0.4920 (−1.26)	**0.5166** (2.19)**
IBM	0.5112 (2.47)**	0.5064 (1.05)	**0.5167** (2.43)**
ATVI	0.5106 (3.39)***	0.5066 (0.94)	**0.5141** (2.40)**
NVDA	0.5186 (4.27)***	0.5065 (1.17)	**0.5242** (4.45)***

Note. The *t*-statistic is used to measure whether a AUC score is significantly better than random (0.50). In bold is the best model for the particular stock. Nearly all AUC scores for the equally weighted and performance weighted ensembles are significantly better than random at $\alpha = 0.01$. Asterisks indicate significance only at *$\alpha = 0.1$, **$\alpha = 0.05$, and ***$\alpha = 0.01$. The ensemble, which is comprised from only the best model at each time, produced inferior results compared to the other two, in both absolute terms and significance.

TABLE 3 AUC scores for lasso and ridge logistic regressions

Stock	Lasso LR	Ridge LR
BA	0.5176 (3.27)***	**0.5182** (3.30)***
F	0.5198 (2.90)***	**0.5216** (3.04)***
DHR	**0.5176** (1.99)**	0.5167 (1.88)**
KO	**0.5236** (4.48)***	0.5228 (3.77)***
MO	**0.5190** (2.16)**	0.5142 (1.65)*
MAR	0.5159 (2.14)**	**0.5171** (2.09)**
AMT	**0.5097** (1.42)**	0.5087 (1.20)
MCD	0.5127 (2.63)***	**0.5150** (2.99)***
DIS	0.5104 (1.91)**	**0.5126** (2.24)**
GE	0.5155 (1.83)**	**0.5161** (1.93)**
EOG	0.5069 (1.07)	**0.5070** (1.17)
GS	**0.5270** (3.64)***	0.5236 (3.39)***
MET	0.5168 (3.03)***	**0.5197** (3.58)***
BK	0.5157 (2.62)***	**0.5199** (2.85)***
AMGN	**0.5158** (2.49)**	0.5132 (2.30)**
ABT	0.5099 (2.07)**	**0.5117** (2.40)**
AET	**0.5104** (1.44)*	0.5086 (1.19)
NEE	0.5273 (3.41)***	**0.5276** (3.64)***
EXC	**0.5239** (3.45)***	0.5221 (3.07)***
IBM	0.5095 (1.36)*	**0.5106** (1.59)*
ATVI	**0.5088** (1.41)*	0.5056 (1.00)
NVDA	0.5123 (2.34)**	**0.5138** (2.73)***

Note. Average AUC performance for the 22 stocks by employing lasso and ridge logistic regressions with a weight decay of $\lambda = 0.1$. In bold is the best model for the particular stock. The *t*-statistic in parentheses indicates whether the result is significantly better than random (AUC = 0.5). Asterisks indicate significance only at *$\alpha = 0.1$, **$\alpha = 0.05$, and ***$\alpha = 0.01$. The majority of the scores are significant at $\alpha = 0.05$ or $\alpha = 0.01$. Furthermore, we can say that the two methods produce comparable results.

unregularized logistic regression can be defined by the following equation:

$$p(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}\boldsymbol{x}+b}}, \qquad (22)$$

where $\boldsymbol{x}$ is the feature vector for one instance, $\boldsymbol{w}$ is a vector of weights, and b is a bias term. By also including regularization of λ, the function for training the model is as follows:

$$\boldsymbol{w}^*, b^* = \underset{\boldsymbol{w},b}{\text{argmin}} \sum_{i=1}^{n} \log(e^{-y_i X_i^T \boldsymbol{w}+b} + 1) + \mathcal{L}_{\text{reg}}, \qquad (23)$$

where n is the number of training examples and $\boldsymbol{y}_i$ is the vector of target values, for example i. Finally, the regularization loss is $\mathcal{L}_{\text{reg}} = \lambda||\boldsymbol{w}||_1$ for lasso and $\mathcal{L}_{\text{reg}} = \lambda||\boldsymbol{w}||_2$ for ridge.
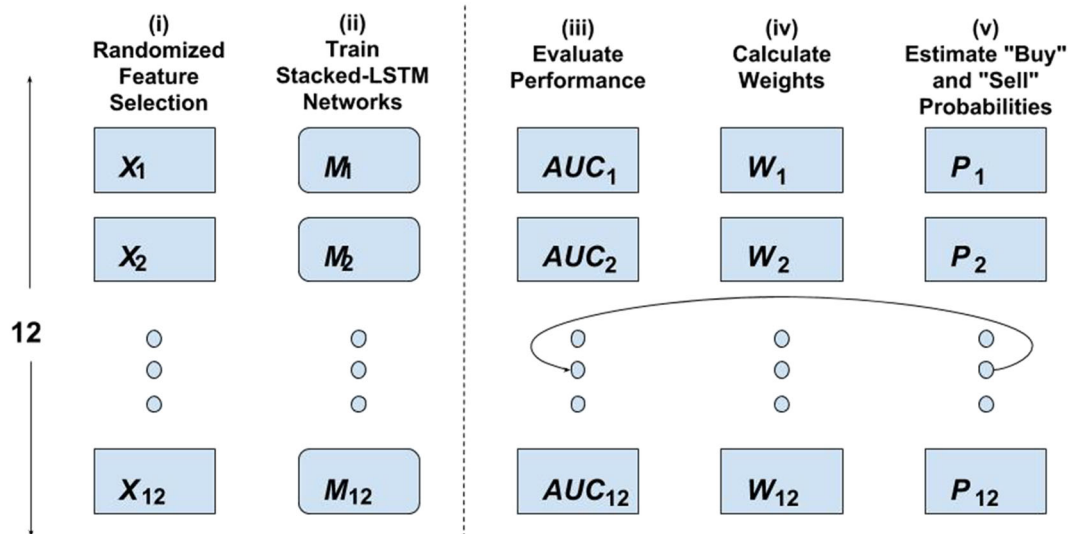
FIGURE 6 Ensemble [Colour figure can be viewed at wileyonlinelibrary.com]
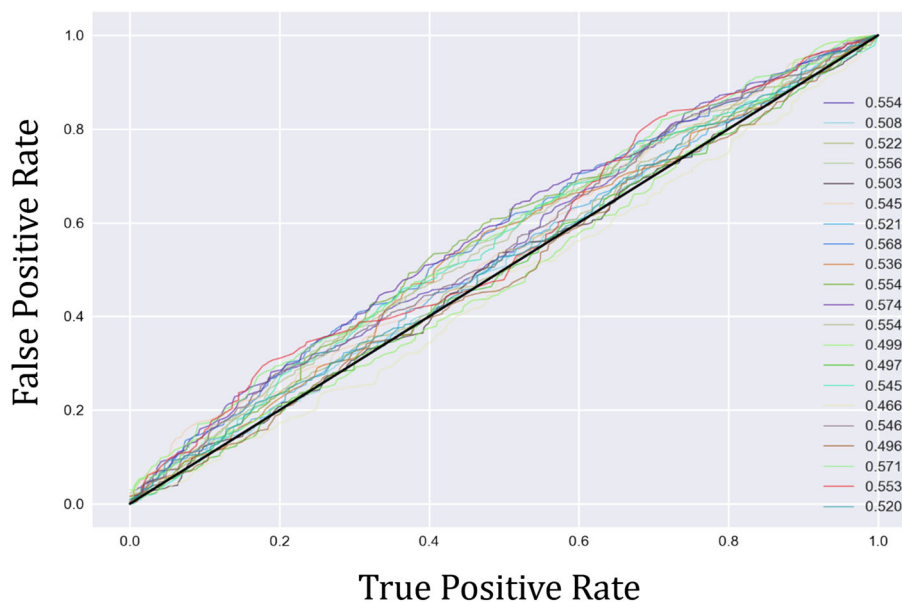

FIGURE 7 Visualization of the Receiver Operating Characteristic (ROC) curves of the Performance-Weighted Ensemble models on the stock NEE. Each line represents the ROC curve of a particular model learned for a specific period. We evaluated our models on 21 distinct periods of time, and thus 21 different models were learned. The identity line defines the performance of a random classifier model and can be considered as a baseline for comparison. The greater the Area Under the Curve (AUC) of the ROC, the better the performance. The vast majority of the classifiers learned for stock NEE, perform better than random, as indicated by their curves being consistently above the identity line, thus having a greater AUC score [Colour figure can be viewed at wileyonlinelibrary.com]

5 | RESULTS

5.1 | LSTM ensembles

The results of our predictive framework are evaluated on the 22 tested stocks. For every stock, we make predictions during 21 weeks of the year 2014 (every 5 minutes).

Each stacked LSTM model is trained for a total of 15 epochs. One epoch is equal to the number of training iterations needed for the algorithm to hypothetically[13] run over all the instances of the training set. Given a mini-batch B and a training set D, one epoch is equal to $\frac{|D|}{|B|} = \frac{1,560}{32} \approx 49$. Consequently, 15 epochs amount to $15 \cdot 49 = 735$ training iterations. The training iterations are the only stopping criterion. We do not monitor the accuracy at every epoch, nor the loss function. We solely focus on producing models that have identified patterns in the training set, and

---

[13] Each mini-batch is constructed via a randomized process with replacement.

the weighting scheme of the ensemble is responsible for selecting "good" and "bad" models.

Twelve models were trained for each period and for each stock. The trained models were used for inference on the testing periods, outputting the probabilities of the softmax layer. Thus, at each period, we have at our disposal the probabilities for each class, according to 12 different models. Using these probabilities, we produced three different type of ensemble. The first two are the equally weighted ensemble and the performance-weighted ensemble, as described in Section 4.4. To demonstrate that the predictability comes from the collective presence of the models, we also produced an ensemble in which, for predicting each observation, we considered only the best model (according to the AUC of the past 390 observations). In Figure 7, we present a visualization of the AUC scores for stock NEE, as obtained by the performance-weighted ensemble.

The combined AUC scores of the three ensemble methods can be found in Table 2. The *t*-statistics in the table indicate that all our scores are significantly greater than an AUC score of 0.50—in other words, our predictions are better than a random guess. This provides evidence for predictability, since we are able to make better than random predictions using past market information. The results in Table 2 demonstrate that using the AUC score of past observations to weight the models in the ensemble produces a better performing model. Additionally, this provides evidence on the existence of patterns in the high-frequency US stock market that are not only durable but also discoverable.

5.2 | Lasso and ridge logistic regressions

To provide another benchmark for the performance-weighted ensemble other than random guessing, lasso logistic regression and ridge logistic regression are used. A weight decay of $\lambda = 0.1$ was applied for the regularization. We employ both of them on the same periods that the ensemble was tested. The AUC scores for each stock using the lasso and ridge logistic regression can be observed in Table 3. The two models produce similar results to one another, which are also significantly better than random, as indicated by the *t*-statistic. Since both of them are fairly simplistic models compared to the LSTM, we may infer that an important part of the predictability comes from the features used in this research.

5.3 | Performance-weighted ensemble versus benchmarks

In Table 4 we provide a comparison (in terms of AUC score) of the performance-weighted ensemble, against the other methods.

With the exception of one stock (EXC), the performance-weighted ensemble produces better results than the other methods.

Finally, we want to test the hypothesis that all the information contained in lasso and ridge classifiers is also present in the performance-weighted ensemble. Since both models are fairly simple, we want to examine the possibility that the performance-weighted ensemble lost some basic information due to its advanced complexity. To do so, we included the lasso and ridge classifiers in the ensemble[14] and tested for improvement in the AUC scores produced. Results with the inclusion of the extra models, along with the paired *t*-test, can be be found in Table 5.

The inclusion produced minor changes in the AUC scores of the performance-weighted ensemble, which are also insignificant. As expected, the largest change was for stock EXC, for which both lasso and ridge had outperformed the performance-weighted ensemble. Considering the results of Table 5, we can conclude that no real value can be added by including the lasso and ridge classifiers in the ensemble, and thus all the information is already incorporated in the performance-weighted ensemble.

6 | CONCLUSIONS AND FURTHER RESEARCH

Here we proposed a machine learning framework for obtaining intraday directional price forecasts. Our framework is based on an ensemble model that combines multiple LSTM neural networks through a robust, performance-evaluated weighting method.

We conducted a large empirical study on the basis of 44 US large cap stocks, which can be translated to 30 GB in raw data or 420 million in number of trades. The raw trade data of every stock were aggregated into 5-minute intervals, which were used to create a large number of technical indicators on different time frames. From the aggregated stock data, we further created sector data sets that were used in predictive modeling. The data were used to train a total of 5,544 LSTM networks that were merged to create 462 ensemble models.

By testing the predictive power of the proposed ensemble on 22 stocks, it was shown that the proposed ensemble is a significant upgrade of regular ensemble weighting methods, such as an equally weighted method or one that considers only the best individual model. Furthermore, it outperformed two simpler models: the lasso and ridge logistic regressions. Additionally, by showing that incorpo-

---

[14]Now the ensemble comprises the 12 stacked LSTMs, the lasso LR and the ridge LR.

ration of the simpler models into the ensemble achieved no significant gain, we confirmed that the ensemble dominates them in terms of information included. Finally, all of the trained models were found to be statistically better than random guessing for the majority of the tested stocks, thus suggesting evidence against the weak form of market efficiency.

Research in artificial intelligence (AI) is growing rapidly, and most likely, along with blockchain technologies, it will shape the world of business and finance in the near future. Although AI methods such as deep neural networks are incredibly powerful (some estimate that they will reach the computational capacity of a human brain by 2025), they fall back in terms of efficiency. Despite the fact that our proposed model outperforms a set of benchmarks, it is much more computationally expensive. The training of the ensemble on a single stock required approximately 150 minutes, whereas estimation of the lasso or ridge logistic regression needs only a few seconds. With

this in mind, we believe that further research should be aimed at maximizing the ratio of predictive accuracy to computational time used, instead of predictive accuracy alone.

This is also apparent when it comes to high-frequency trading. One should be able to make not just good predictions but also fast ones. We believe that our method can be further optimized by tuning the models. As our research was limited in terms of hardware, most of the hyperparameters were chosen by means of trial-and-error experiments or heuristic rules found in the literature. Correct tuning can be carried out by setting up a grid search through candidate values, or even by an evolutionary algorithm.

Finally, we would like to point out that this research was aimed at achieving a general predictability by estimating class probabilities on a varying decision threshold. Profitability of this framework should also be investigated. This can be accomplished by optimizing the decision threshold.

TABLE 4 Performance-weighted ensemble against the other methods

Stock	Lasso LR	Ridge LR	Equally weighted ensemble	Performance-weighted ensemble
BA	0.5176 (0.63)	0.5182 (0.54)	0.5153 (1.09)	**0.5224**
F	0.5198 (1.72)	0.5216 (1.49)[+]	0.5232 (1.57)[+]	**0.5355**
DHR	0.5176 (0.50)	0.5167 (0.58)	0.5144 (1.03)	**0.5231**
KO	0.5236 (0.15)	0.5228 (0.23)	0.5173 (0.88)	**0.5251**
MO	0.5190 (0.09)	0.5142 (0.55)	0.5110 (1.17)	**0.5201**
MAR	0.5159 (1.89)[++]	0.5171 (1.67)[+]	0.5251 (1.28)	**0.5346**
AMT	0.5097 (0.37)	0.5087 (0.46)	0.5086 (0.56)	**0.5132**
MCD	0.5127 (1.45)[+]	0.5150 (1.15)	0.5223 (0.27)	**0.5248**
DIS	0.5104 (1.10)	0.5126 (0.83)	0.5150 (0.65)	**0.5196**
GE	0.5155 (0.90)	0.5161 (0.85)	0.5179 (0.87)	**0.5251**
EOG	0.5069 (1.36)[+]	0.5070 (1.40)[+]	0.5155 (0.48)	**0.5193**
GS	0.5270 (0.91)	0.5236 (1.29)	0.5149 (2.63)[+++]	**0.5362**
MET	0.5168 (0.59)	0.5197 (0.25)	0.5178 (0.48)	**0.5218**
BK	0.5157 (1.09)	0.5199 (0.57)	0.5173 (1.09)	**0.5254**
AMGN	0.5158 (0.67)	0.5132 (0.96)	0.5161 (0.67)	**0.5224**
ABT	0.5099 (1.18)	0.5117 (0.93)	0.5057 (1.86)[++]	**0.5190**
AET	0.5104 (1.47)[+]	0.5086 (1.63)[+]	0.5133 (1.41)[+]	**0.5263**
NEE	0.5273 (0.30)	0.5276 (0.28)	0.5207 (1.24)	**0.5304**
EXC	**0.5239** (-0.71)	0.5221 (-0.52)	0.5102 (0.72)	0.5166
IBM	0.5095 (0.74)	0.5106 (0.64)	0.5112 (0.67)	**0.5167**
ATVI	0.5088 (0.62)	0.5056 (1.04)	0.5106 (0.52)	**0.5141**
NVDA	0.5123 (1.58)[+]	0.5138 (1.39)[+]	0.5186 (0.81)	**0.5242**

Note. The performance weighted ensemble produces consistently better forecasts than the other methods, with the only exception being for stock EXC, where it underperformed with respect to lasso and ridge classifiers. In bold is the best model for the particular stock and the values in parentheses indicate the one-side paired *t*-test between each method and the performance-weighted ensemble for 21 periods, indicating whether the performance-weighted ensemble is significantly better than the other three methods. Superscript plus signs indicate that the performance-weighted ensemble is significantly better than the benchmark at [+]$\alpha = 0.1$, [++]$\alpha = 0.05$, and [+++]$\alpha = 0.01$.

TABLE 5 Inclusion of lasso and ridge in the ensemble

Stock	Performance-weighted ensemble	Performance-weighted ensemble + lasso + ridge	Paired *t*-test
BA	0.5224	**0.5237**	0.18
F	**0.5355**	0.5354	0.00
DHR	0.5231	**0.5242**	0.12
KO	0.5251	**0.5254**	0.03
MO	0.5201	**0.5203**	0.02
MAR	**0.5346**	0.5355	0.10
AMT	0.5132	0.5132	0.00
MCD	0.5248	**0.5252**	0.04
DIS	0.5196	**0.5197**	0.02
GE	0.5251	**0.5254**	0.03
EOG	**0.5193**	0.5191	−0.03
GS	0.5362	**0.5368**	0.07
MET	0.5218	**0.5226**	0.10
BK	0.5254	0.5254	0.01
AMGN	0.5224	0.5224	−0.01
ABT	0.5190	**0.5195**	0.06
AET	0.5263	**0.5269**	0.05
NEE	0.5304	**0.5310**	0.07
EXC	0.5166	**0.5191**	0.23
IBM	0.5167	**0.5168**	0.01
ATVI	0.5141	**0.5145**	0.05
NVDA	**0.5242**	0.5236	−0.07

Note. AUC scores of the performance-weighted ensemble containing only stacked LSTMs and the the performance-weighted ensemble containing stacked LSTMs and the lasso and ridge logistic classifiers. In bold is the best model for the particular stock. We can only observe minor differences between the two test models, and all of them are insignificant even at $\alpha = 0.25$.

ORCID

Svetlana Borovkova https://orcid.org/0000-0001-6585-999X

REFERENCES

Baestaens, D. E., Van Den Bergh, W. M., & Vaudrey, H. (1995). Options as a predictor of common stock price changes. *European Journal of Finance*, *1*, 325–343.

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*, 1–24.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. Available at: http://arxiv.org/abs/1206.5533

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, *25*, 383–417.

Fischer, T., & Krauss, C. (2017). Deep learning with long short-term memory networks for financial market predictions. (*AU Discussion Papers in Economics*). Erlangen, Germany: Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, *9*, 249–256.

Hegazy, O., Soliman, O. S., & Salam, M. A. (2013). A machine learning model for stock market prediction. *International Journal of Computer Science and Telecommunications*, *4*(12), 16–23.

Hinton, G., Srivastava, N., & Swersky, K. (2012). Neural networks for machine learning. (*Lecture 6*). Toronto, Canada: University of Toronto.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *8*, 1735–1780.

Huanga, W., Nakamoria, Y., & Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers and Operations Research*, *32*, 2513–2522.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization, In. *Proceedings, IEEE International Conference on Neural Networks*. Piscataway, NJ: IEEE, Vol. *4*, pp. 1942–1948.

Kent, A. (1989). Signal detection theory: Valuable tools for evaluating inductive learning, *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 160–163). Amsterdam, Netherlands: Elsevier.

Lei Ba, J., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. Retrieved from arXiv:1607.06450.

Nelson, M. Q., Pereira, A. C. M., & de Oliveira, R. A. (2017). Stock markets price movement prediction with LSTM neural networks,

2017 International Joint Conference on Neural Networks (IJCNN). Piscataway, NJ: IEEE, pp. 1419–1426.

Pai, P. F., & Lin, C. S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*, *33*, 497–505.

Qu, H., & Zhang, Y. (2016). A new kernel of support vector regression for forecasting high-frequency stock returns. *Mathematical Problems in Engineering*, 2016, 9, Article ID 4907654. https://doi.org/10.1155/2016/4907654

Rechenthin, M. D. (2014). Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction. (PhD thesis), University of Iowa.

Refenes, A. N., Zapranis, A., & Francis, G. (1994). Stock performance modeling using neural networks: A comparative study with regression models. *Neural Networks*, *7*, 375–388.

Semeniuta, S., Severyn, A., & Barth, E. (2016). Recurrent dropout without memory loss. Retrieved from arXiv:1603.05118.

Srivastava, N., Hinton, G., Krizhevsky, A, Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

AUTHOR BIOGRAPHIES

Svetlana Borovkova is currently an Associate Professor of Quantitative Finance at the Vrije Universiteit Amsterdam, where she is the program director of the Quantitative Risk Management honours program. Dr. Borovkova's research ranges from commodity finance to exotic derivatives, news analytics and various aspects of financial risk and FinTech. She is also a partner in a risk advisory firm Probability and Partners, which specializes in market risk, data analytics, and quantitative investment strategies. Dr. Borovkova is a frequently invited speaker on international finance and energy conferences, such as Global Derivatives, Bachelier Congress for Mathematical Finance, Behavioral Finance and Sentiment Analysis. Previously she held positions in Delft University of Technology and in Shell Trading, London. She got her Ph.D. in 1998 from the University of Groningen, The Netherlands, and Oregon State University, USA, for analysis of nonlinear and chaotic time series, and MSc degrees in computer science and in mathematics from Moscow and Utrecht.

Ioannis Tsiamas is currently studying Artificial Intelligence at the University of Amsterdam. In 2018, he received his masters in Quantitative Risk Management at the Vrije Universiteit Amsterdam.

APPENDIX A: TABLES

TABLE A1 Stock universe

Stock name (symbol)	Sector	Competitor name (symbol)	Exchange
The Boeing Company (BA)	Capital Goods	United Technologies Corporation (UTX)	NYSE
Ford Motor Company (F)	Capital Goods	General Motors Company (GM)	NYSE
Danaher Corporation (DHR)	Capital Goods	Thermo Fisher Scientific Inc (TMO)	NYSE
The Coca-Cola Company (KO)	Consumer Non-Durables	Pepsico, Inc. (PEP)	NYSE
Altria Group (MO)	Consumer Non-Durables	Philip Morris International Inc. (PM)	NYSE
Marriott International (MAR)	Consumer Services	Las Vegas Sands Corp. (LVS)	NASDAQ
American Tower Corporation, REIT (AMT)	Consumer Services	Simon Property Group, Inc. (SPG)	NYSE
McDonald's Corporation (MCD)	Consumer Services	Starbucks Corporation (SBUX)	NASDAQ
The Walt Disney Company (DIS)	Consumer Services	Time Warner Inc. (TWX)	NYSE
General Electric Company (GE)	Energy	Emerson Electric Company (EMR)	NYSE
EOG Resources, Inc. (EOG)	Energy	Occidental Petroleum Corporation (OXY)	NYSE
The Goldman Sachs Group, Inc. (GS)	Finance	Morgan Stanley (MS)	NYSE
MetLife, Inc. (MET)	Finance	Prudential Financial, Inc. (PRU)	NYSE
The Bank Of New York Mellon Corporation (BK)	Finance	U.S. Bancorp (USB)	NYSE
Amgen Inc. (AMGN)	Health Care	Gilead Sciences, Inc. (GILD)	NASDAQ
Abbott Laboratories (ABT)	Health Care	Merck & Company, Inc. (MRK)	NYSE
Aetna Inc. (AET)	Health Care	Cigna Corporation (CI)	NYSE
NextEra Energy, Inc. (NEE)	Public Utilities	Dominion Energy, Inc. (D)	NYSE
Exelon Corporation (EXC)	Public Utilities	Duke Energy Corporation (DUK)	NYSE
International Business Machines Corporation (IBM)	Technology	HP Inc. (HPQ)	NYSE
Activision Blizzard, Inc (ATVI)	Technology	Adobe, Inc (ADBE)	NASDAQ
NVIDIA Corporation (NVDA)	Technology	Intel Corporation (INTC)	NASDAQ

Note. Stocks used for this research are shown, along with the sector they belong to, their primary competitor and the exchange that they are traded at.

TABLE A2 Basic features

Features	Formula
Open (P_{open})	P_1
Close (P_{close})	P_N
Low (P_{low})	$\min(\boldsymbol{P})$
High (P_{high})	$\max(\boldsymbol{P})$
Volume-weighted average price (P_{vwap})	$\sum_i^N P_i S_i / \sum_i^N S_i$
Mean price difference (MeanPriceD)	$(P_1 - P_N)/(N - 1)$
Max. price difference (MaxPriceD)	$\max(\Delta \boldsymbol{P})$
Standard deviation of price differences (StdPriceD)	$\sqrt{\frac{1}{N-1} \sum_{i=1}^{N-1} (\Delta P_i - \text{MeanPriceD})^2}$
Total number of trades	N
Number of market sweep trades	$\sum_{i=1}^{N} \boldsymbol{I}_{cond=F}$
Max. trade size	$\max(\boldsymbol{S})$
Mean trade size	$\sum_{i=1}^{N} S_i / N$
Volume	$\sum_i^N S_i$
Positive volume (volume$^+$)	$\sum_{i=1}^{N} \boldsymbol{I}_{\Delta P_i > 0} S_i$
Negative volume (volume$^-$)	$\sum_{i=1}^{N} \boldsymbol{I}_{\Delta P_i < 0} S_i$
Neutral volume (volume^0)	$\sum_{i=1}^{N} \boldsymbol{I}_{\Delta P_i = 0} S_i$

Note. Features are extracted from the raw trade data. Formulas indicate how these features are extracted from the split into 5-minute interval data sets. N is the number of trades that took place in this interval, P_i is the price of trade at moment i, S_i is the size of the trade, and $\Delta \boldsymbol{P}$ denotes the price differences vector. $\boldsymbol{I}$ stands for indicator function.

TABLE A3 Advanced features

Feature	Versions in stock	Versions in sector/index
Percentage change in price	7	6
Price difference (on raw level)	3	0
Std. of returns (cross-sectionally)	0	1
Volume	4	3
Accumulation/distribution index	6	6
True range	1	1
Probability and conditional probability	10	10
Rolling regression	4	1
Sharpe ratio	5	5
Price disagreement	6	6
Price polarity	6	6
Bollinger bands	10	10
Stochastic oscillator	5	5
Relative strength index	5	5
Commodity channel index	5	5
Average directional index	5	5
Double and triple exponentially smoothed returns	5	5
Moving average convergence/divergence	5	5
Money flow index	5	5
	97	90

Note. In this table we present a summary of all the features in each data set and how many different versions of them are included at stock level and at sector/index level.

APPENDIX B: TECHNICAL INDICATORS

B.1 | Moving averages

Many of the technical indicators used in this research involve the calculation of several moving averages. Moving averages are used to smooth noisy time series and highlight short or long-term trends.

The simple moving average $\text{SMA}(x_t, n)$ is composed from the mean of the last n observations of time-series x_t.

$$\text{SMA}(x_t, n) = \frac{1}{n}\sum_{i=1}^{n} x_{t-i}. \quad (B1)$$

The exponential weighed moving average $\text{EWMA}(x, \alpha)$ is calculated by exponential decreasing the weight of observations x_i with respect to their distance from x_t.

$$\text{EWMA}(x_t, \alpha) = \alpha x_t + (1-\alpha)\text{EWMA}(x_{t-1}, \alpha). \quad (B2)$$

B.2 | Sharpe ratio

Sharpe ratio is a measure that summarizes the trade-off between return and risk. Although mostly used for evaluating investment strategies, it can also be used as a technical indicator. Sharpe ratio is calculated for $n = $

[6, 12, 36, 78, 234].

$$\text{Sharpe}_t = \frac{(\Pi_{i=1}^{n}(1 + R_{t+1-i}) - 1) - R_{rf,t}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(R_i - \frac{1}{n}\sum_{k=1}^{n}(1 + R_{t+1-k}))^2}}, \quad (B3)$$

where R_{rf} is the risk free rate.

B.3 | Accumulation/distribution index (ADI)

ADI is a popular volume indicator, which is calculated as the cumulative sum of the product of close location value (CLV) and the trading volume.

$$\text{CLV}_t = \frac{(P_{\text{close},t} - P_{\text{low},t}) - (P_{\text{high},t} - P_{\text{close},t})}{P_{\text{high},t} - P_{\text{low},t}},$$

$$\text{ADI}_t = \sum_{i=1}^{t} \text{CLV}_t \, \text{Volume}_i. \quad (B4)$$

B.4 | Bollinger bands (BB)

Bollinger Bands are used to construct volatility indicators by evaluating the distances of current typical price to the previous n typical prices. For the purpose of this research

we are using $d = 3$ and $n = [6, 12, 36, 78, 234]$.

$$\text{MiddleBB}_t = \text{SMA}(P_{\text{typical},t}, n), \qquad (B5)$$

$$\sigma = \sqrt{\frac{1}{n}\sum_i^n P_{\text{typical},t-i} - \text{MiddleBB}_t}, \qquad (B6)$$

$$\text{UpperBB} = \text{MiddleBB}_t + d\sigma \text{LowerBB} = \text{MiddleBB}_t - d\sigma, \qquad (B7)$$

$$\%b = \frac{P_{\text{close},t} - \text{LowerBB}_t}{\text{UpperBB}_t - \text{LowerBB}_t},$$
$$\text{Bandwith}_t = \frac{\text{UpperBB}_t - \text{LowerBB}_t}{\text{MiddleBB}_t}. \qquad (B8)$$

B.5 | Stochastic oscillator

The stochastic oscillator is a momentum indicator that aims to predict the oscillations of the price by taking into account the price range during the last n observations.

$$K_t = \frac{P_{\text{close},t} - L_n}{H_n - L_n}, \qquad (B9)$$

where $L_n = \min(P_{\text{low},t-n}, \dots, P_{\text{low},t})$ and $H_n = \min(P_{\text{high},t-n}, \dots, P_{\text{high},t})$.

Finally, the stochastic oscillator is a simple moving average of length m.

$$\text{StochOsci}_t = 100\,\text{SMA}(K_t, m). \qquad (B10)$$

The values used for this research are $n = [6, 12, 36, 78, 234]$ and $m = [3, 3, 2, 2, 2]$ respectively.

B.6 | Relative strength index (RSI)

The relative strength index is another momentum indicator that uses the velocity and magnitude of price movements to identify if an asset is overbought or oversold. It ranges from 0 to 100, with low values indicating that the asset is oversold, while high values indicate that is overbought. RSI is calculate for $n = [6, 12, 36, 78, 234]$.

$$U_t = \begin{cases} P_{\text{close},t} - P_{\text{close},t-1}, & P_{\text{close},t} > P_{\text{close},t-1} \\ 0, & P_{\text{close},t} \leq P_{\text{close},t-1}, \end{cases}$$

$$D_t = \begin{cases} 0, & P_{\text{close},t} > P_{\text{close},t-1} \\ P_{\text{close},t} - P_{\text{close},t-1}, & P_{\text{close},t} \leq P_{\text{close},t-1}. \end{cases}$$

$$\text{RSI}_t = 100 - \frac{100}{1 + \frac{\text{EWMA}(U_t, 1/n)}{\text{EWMA}(D_t, 1/n)}}. \qquad (B11)$$

B.7 | Commodity channel index (CCI)

The commodity channel index functions similarly to the RSI, indicating whether an asset is overbought or oversold. The main difference is that CCI operates on the typical price. It ranges from -100 to 100, with small values hinting that the stock is oversold, whereas large values suggest the opposite. The CCI is calculate for $n = [6, 12, 36, 78, 234]$.

$$\text{CCI}_t = \frac{1}{0.015}\frac{P_{\text{typical},t} - \text{SMA}(P_{\text{typical},t}, n)}{\text{MAD}(P_{\text{typical},t}, n)}, \qquad (B12)$$

where MAD is the mean absolute deviation, calculated as

$$\text{MAD}(x_t, n) = \frac{1}{n}\sum_{i=1}^{n}\left|x_{t-i} - \frac{1}{n}\sum_{k=1}^{n}x_{t-k}\right|. \qquad (B13)$$

B.8 | Average directional moving index (ADX)

The average directional moving index is a combination of two other indicators—the positive directional DI^+ and the negative directional DI^-, and it is used as a measure of trend strength. The ADX is calculated for $n = [6, 12, 36, 78, 234]$.

$$\begin{aligned} \text{UpMove}_t &= P_{\text{high},t} - P_{\text{high},t-1}, \\ \text{DownMove}_t &= P_{\text{low},t-1} - P_{\text{low},t}, \end{aligned} \qquad (B14)$$

$$\text{DM}_t^+ = \begin{cases} \text{UpMove}_t, & \text{UpMove}_t > \text{DownMove}_t \text{ and } \text{UpMove}_t > 0 \\ 0, & \text{otherwise}, \end{cases}$$

$$\text{DM}_t^- = \begin{cases} \text{DownMove}_t, & \text{DownMove}_t > \text{UpMove}_t \text{ and } \text{DownMove}_t > 0 \\ 0, & \text{otherwise}, \end{cases}$$

$$\begin{aligned} \text{DI}_t^+ &= 100\frac{\text{EWMA}(\text{DM}_t^+, 1/n)}{\text{SMA}(\text{TR}_t, n)}, \\ \text{DI}_t^- &= 100\frac{\text{EWMA}(\text{DM}_t^-, 1/n)}{\text{SMA}(\text{TR}_t, n)}, \end{aligned} \qquad (B15)$$

where TR stands for true range and is defined as

$$\begin{aligned} TR_t = \max(&P_{\text{high},t} - P_{\text{low},t}, |P_{\text{high},t} \\ &- P_{\text{close},t-1}|, |P_{\text{low},t} - P_{\text{close},t-1}|). \end{aligned} \qquad (B16)$$

Finally, ADX is defined as

$$\text{ADX}_t = 100\,\text{EWMA}\left(\frac{|\text{DI}_t^+ - \text{DI}_t^-|}{|\text{DI}_t^+ + \text{DI}_t^-|}, 1/n\right). \qquad (B17)$$

B.9 | Double and triple exponentially smoothed returns

Both these indicators are calculated as the percentage change of the double DIX or triple TRIX exponentially

smoothed prices. DIX is calculated for $n = 234$, whereas TRIX is calculated for $n = [6, 12, 36, 78]$.

$$DIX_t = \frac{\text{EWMA}(\text{EWMA}(P_{\text{close},t}, 1/n), 1/n) - \text{EWMA}(\text{EWMA}(P_{\text{open},t}, 1/n), 1/n)}{\text{EWMA}(\text{EWMA}(P_{\text{open},t}, 1/n), 1/n)}. \tag{B18}$$

B.10 | Moving average convergence–divergence (MACD)

MACD is a technical indicator composed from three separate time series, used for detection changes in the trend of a price, with respect to strength, momentum, duration, and direction. First, the divergence of two EWMAs on the typical price is measured and then MACD is calculated as the difference of the divergence and EWMA of the divergence. MACD is calculated for the following sets of n, $n_1 = [3, 6, 18, 36, 117]$, $n_2 = [6, 12, 36, 78, 234]$ and $n_3 = [2, 4, 12, 26, 78]$.

$$\text{Divergence}_t = \text{EWMA}(P_{\text{typical},t}, 1/n_1) \\ - \text{EWMA}(P_{\text{typical},t}, 1/n_2), \tag{B19}$$

$$\text{MACD}_t = \text{Divergence}_t - \text{EWMA}(\text{Divergence}_t, 1/n_3). \tag{B20}$$

B.11 | Money flow index (MFI)

MFI is a momentum indicator based on the product of price and volume, which is defined as money flow. MFI is calculated for $n = [6, 12, 36, 78, 234]$.

$$\text{MF}_t = P_{\text{typical},t} \text{Volume}_t. \tag{B21}$$

Then positive money flow is defined as the cumulative sum of all the instances where there has been a positive change in the price. Negative money flow is calculated using the opposite logic.

$$\text{MF}_t^+ = \sum_{i=1}^{n} \text{MF}_i \, \boldsymbol{I}_{P_{\text{typical},t-i} - P_{\text{typical},t-i-1} > 0}, \tag{B22}$$

$$\text{MF}_t^- = \sum_{i=1}^{n} \text{MF}_i \, \boldsymbol{I}_{P_{\text{typical},t-i} - P_{\text{typical},t-i-1} < 0}, \tag{B23}$$

where $\boldsymbol{I}$ is the indicator function.

Finally, MFI is calculated as the ratio of the positive money flow to the sum of positive and negative flows.

$$\text{MFI}_t = 100 \frac{\text{MF}_t^+}{\text{MF}_t^+ + \text{MF}_t^-}. \tag{B24}$$

B.12 | Price disagreement and polarity

Disagreement and polarity are measures frequently used in sentiment analysis, calculated by the amount of positive, negative, and neutral news for a period of n. In this research we are calculating disagreement and polarity using the positive, negative, and neutral trading volume, as defined in Table A2. These indicators are calculated for $n = [0, 6, 12, 36, 78, 234]$.

$$\text{Disagreement}_t = \left| 1 - \left| \frac{\sum_{i=1}^{n} \text{Volume}_{t-i}^+ - \sum_{i=1}^{n} \text{Volume}_{t-i}^-}{\sum_{i=1}^{n} \text{Volume}_{t-i}^+ + \sum_{i=1}^{n} \text{Volume}_{t-i}^-} \right| \right|, \tag{B25}$$

$$\text{Polarity}_t = \frac{\sum_{i=1}^{n} \text{Volume}_{t-i}^+ - \sum_{i=1}^{n} \text{Volume}_{t-i}^-}{\sum_{i=1}^{n} \text{Volume}_{t-i}^0}. \tag{B26}$$