To appear in *Quantitative Finance*, Vol. 00, No. 00, Month 20XX, 1–27

An Ensemble of LSTM Neural Networks for High-Frequency Stock Market Classification

Svetlana Borovkova and Ioannis Tsiamas

Vrije Universiteit Amsterdam

(*Received 00 Month 20XX; in final form 00 Month 20XX*)

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 non-stationarities 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: High-Frequency Trading, Deep Learning, LSTM Neural Networks, Ensemble Models

JEL Classification: C45, C53, C55, G17

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 is re-invigorated again 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 receive 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, at which even the collective 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 the 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", i.e., 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 previous patterns and prices are

Electronic copy available at: https://ssrn.com/abstract=3202313

fed into the networks - this is achieved by using the so-called *Long-Short Term Memory 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 non-stationarities in the data.

We apply our methodology to 22 large cap US stocks. We use one year of raw trade data, that was 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 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 datasets. 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 dataset was solely used for feature engineering. For the other 11 months, we operated in a rolling window way, where one month of data is used for training the networks, one 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 2-core 2.3GHz CPU and 8GB 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 section 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 D. E. Baestaens and Vaudrey (1995) and A. N. Refenes and Francis (1994), who used simple ANN (Artificial Neural Network) architectures and compared their performance to Logistic Regression(LR) models. W. Huanga (2005) found that SVMs (Support Vector Machines) can achieve better results than traditional statistical methods, while Pai and Lin (2005) proposed a hybrid ARIMA - SVM method for price forecasting.

Later contributions include O. Hegazy and Salam (2013), who used Particle Swarm Optimization [1] to fine-tune the hyperparameters of an SVM regressor, achieving significantly smaller Mean Squared Errors (MSE) on several US stocks. D. M. Q. Nelson (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

[1] Nature-inspired algorithm that uses a collection of candidate solutions as well as their respective positions and velocities to find the optimal solutions. (Kennedy and Eberhart 1995)

Electronic copy available at: https://ssrn.com/abstract=3202313

(2017) conducted a large-scale research, using daily S&P500 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 trigger periodically momentum and reversal, designed a new SVM kernel method [1] for forecasting high-frequency market directions and applied their method to the Chinese CSI300 index. Their results were significantly better compared to the Radial Basis Function (RBF) [2] kernel and the Sigmoid kernel.

One of the most remarkable contributions to deep learning for stock price prediction is that by W. Bao and Rao (2017). They proposed a predictive framework composed of three parts. First, they apply a Wavelet Transformation(WT) [3] to the financial dataset (prices, technical indicators, macroeconomic data). Then the de-noised data are passed through stacked-Autoencoders [4] 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 the 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 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, DTs), 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 seven months of 2012.

The main distinguishing features of this research when compared to the contributions outlined above, is 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, i.e., 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 one year (2014) of high-frequency historical data, for 22 large-cap [5] 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 [6], so we also construct sector datasets, by aggregating the stocks cross-sectionally. The summary of the selected stocks is presented in Table A1.

Our choice of using large-cap US stocks is motivated by two reasons: 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 is also the liquidity of a stock, as high liquidity ensures low trading costs. We selected the stocks in the way that keeps the ratio of the market capitalization of each

[1] Kernel functions enable higher dimensional operations without operating in the actual higher dimensional space.
[2] RBF maps inputs to higher dimensional space using the Euclidean distance.
[3] Signal processing technique, which transforms a series from time to frequency domain.
[4] Class of neural networks, which learn a representation of input x to a latent space Z and then reconstruct x_0, similarly to x (Bengio 2009).
[5] More than 10$ billion in market capitalization.
[6] According to market capitalization: Capital Goods, Consumer Non-Durables, Consumer Services, Energy, Finance, Health Care, Public Utilities and Technology.

Electronic copy available at: https://ssrn.com/abstract=3202313

sector close to that in SP500. We decided to avoid mega-cap stocks such as Google and Apple, since otherwise the technology sector would by far dominate the other sectors.

Data were obtained from the NYSE Trade-and-Quote(TAQ) database, which is accessible through the WRDS [1] interface. The TAQ database contains raw trade information for US stocks at a micro-second 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 contains 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 [2].

We processed the raw data to clean them from "bad" trades and aggregate them into 5-minute intervals. "Bad" are the trades 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 contains 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. 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 18252 observations per stock were produced.

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

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 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", while periods with zero or negative returns are labeled as "Sell".

In a high-frequency setting, a large number of time intervals is 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 1326 past observations. Given the fast-shifting probability distributions of our target variable, it becomes clear how important the use of multiple models becomes, in order to capture this variability.

[1] Wharton Research Data Services `https://wrds-web.wharton.upenn.edu/`

[2] Which let a trader to purchase stock from several exchanges until the given order size is satisfied.

Electronic copy available at: https://ssrn.com/abstract=3202313

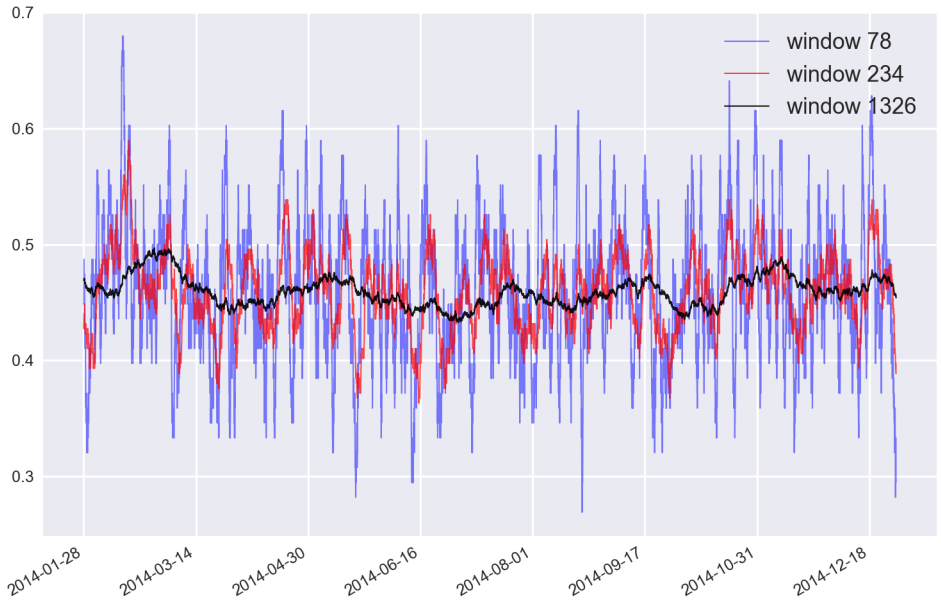


Figure 1. Probability distribution of class "Buy".

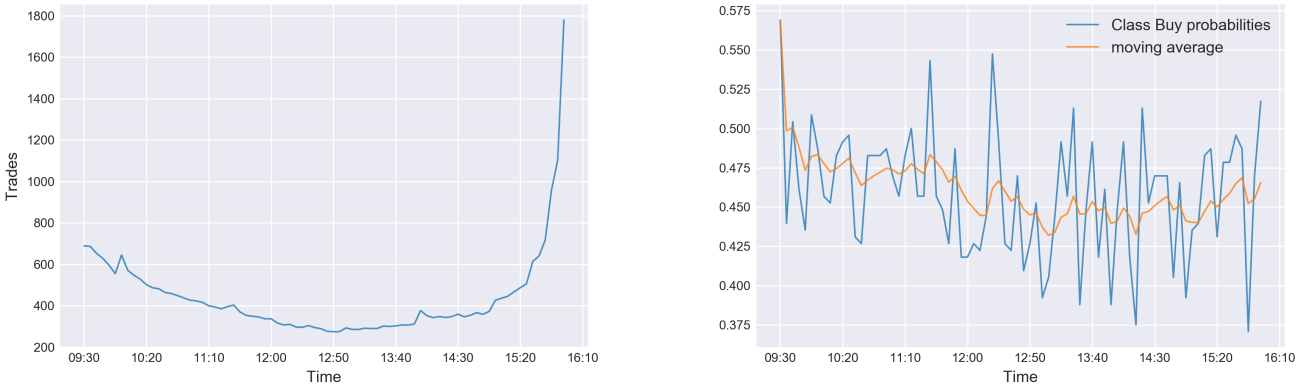
To train the networks, we constructed a large number of technical indicators. Technical indicators are measures used in technical analysis, that aim to identify "Buy" or "Sell" signals in charts and graphical representations of stocks. They can be categorized in trend, volatility, volume and momentum 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 different 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 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 1326 (approximately one 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, 1326 observations, for probabilities $\mathcal{P}(y_t = Buy)$ and 72, 234, 1326 observations, for conditional probabilities $\mathcal{P}(y_t = Buy|y_{t-1} = Sell)$, $\mathcal{P}(y_t = Buy|y_{t-1} = Buy)$. Certain trends can be observed along the span of one 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 3 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).

Electronic copy available at: https://ssrn.com/abstract=3202313



(a) Trading Intensity

(b) Price Movements

Figure 2. 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. 2(b): Class "Buy" probabilities throughout the day, measured by taking into account all the 44 stocks. We can identify a decreasing trend until 13:00, which is followed by an increasing one.

4. Methodology

4.1. *Stacked Long Short Term Memory Network with Layer Normalization*

Recurrent Neural Networks (RNNs) have been effectively used to forecast sequential, possibly non-stationary data. 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:

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

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 modelled 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:

$$\frac{\partial c_t}{\partial c_{t-j}} = \frac{\partial c_t}{\partial c_{t-1}} \cdot \frac{\partial c_{t-1}}{\partial c_{t-2}} \cdot \ldots \cdot \frac{\partial c_{t-j+1}}{\partial c_{t-j}} \approx 0 \tag{2}$$

The Long Short Term Memory(LSTM) (Hochreiter and 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 (J. L. Ba and Hinton 2016) is also introduced in our model to prevent neurons from

Electronic copy available at: https://ssrn.com/abstract=3202313

saturating [1], by keeping their inputs centered. It works by calculating the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of the inputs across the time-dimension and normalizing them accordingly. Following equations describe the LSTM with Layer Normalization:

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

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

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

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

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

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

were $\odot$ denotes the element-wise multiplication operator, $\boldsymbol{W}_{*x}$ input weight matrices, $\boldsymbol{W}_{*h}$ 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:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad sigm(x) = \frac{1}{1 + e^{-x}} \tag{9}$$

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

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, while an output of 1 means that it should retain this information in its entirety. Lastly, the output gate is of

[1] 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.

Electronic copy available at: https://ssrn.com/abstract=3202313

similar function, as the other two sigmoids. It highlights the information that will be passed to the next hidden state. Then equations (7) and (8) are used to compute the output of the LSTM layer. The choice of not including bias terms in the equations of the four gates [(3)-(6)] is motivated by the use of layer normalization. Although not mentioned in J. L. Ba and Hinton (2016), the function of the bias terms in the LSTM is taken over by the shift terms $\boldsymbol{\beta}_*$ of $LayerNorm$ and thus their presence is considered redundant.

In this research we implemented a stacked-LSTM, comprising of two LSTM networks (A) and (B). 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 at 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 the LSTM(A) is fed to the LSTM(B) and accordingly its output is passed through a softmax layer that transforms the raw output to class probabilities:

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

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

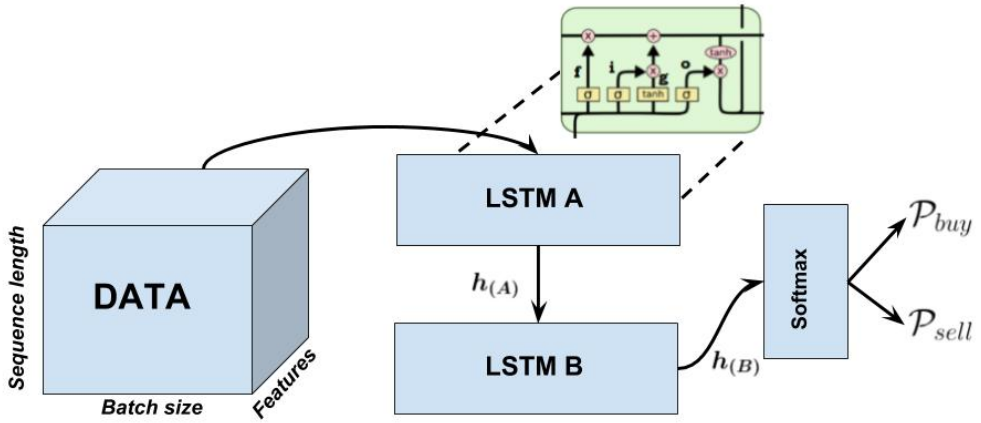


Figure 3. Stacked-LSTM with a softmax layer. During training, data flows into the model in a form of 3-dimensional tensor, with d_1 being the batch size, d_2 the sequence length and d_3 the features. Output of the LSTM(A) $h_{(A)}$ enters LSTM(B) and then the its output $h_{(B)}$ gets transformed into probabilities of the two classes "Buy" and "Sell". In the secondary figure (source: http://colah.github.com), the inside process of an LSTM can be visualized, denoting the four gates and the transfer functions.

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

Electronic copy available at: https://ssrn.com/abstract=3202313

imbalance [1] observed in our data, resulting in $p^{(t)}_{weighted}$. Thus, the loss function for a batch size of d_1, sequence length of d_2, class label y and regularization term $\mathcal{L}_{reg}$, the loss function is the following:

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

The regularization term used in this research is the $\ell 2$ weight decay, that employs the euclidean norm of the regularized parameters to penalize complex weights and avoid overfitting. For the collection of regularized parameters $\boldsymbol{\Theta}$, which can be found in 4.2, and a regularization magnitude λ, which was tuned to 0.1, we have:

$$\mathcal{L}_{reg} = \frac{\lambda}{d_1 d_2} \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} ||\boldsymbol{\theta}||_2 \tag{14}$$

The choices for batch size $d1$ and sequence length $d2$ were 32 and 5 accordingly. Using a batch size of 32 is suggested by in 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 were not justified by the small gains.

The Glorot initialization (Glorot and 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.

Parameter	**Dimensions**	**Initialization**	**Regularized**
$\boldsymbol{W}^{(A)}_{gx}$, $\boldsymbol{W}^{(A)}_{ix}$, $\boldsymbol{W}^{(A)}_{fx}$, $\boldsymbol{W}^{(A)}_{ox}$	250×64	$\mathcal{N}(0, \sqrt{\frac{2}{250+64}})$	True
$\boldsymbol{W}^{(A)}_{gh}$, $\boldsymbol{W}^{(A)}_{ih}$, $\boldsymbol{W}^{(A)}_{fh}$, $\boldsymbol{W}^{(A)}_{oh}$	64×64	$\mathcal{N}(0, \sqrt{\frac{2}{64+64}})$	True
$\boldsymbol{W}^{(B)}_{gx}$, $\boldsymbol{W}^{(B)}_{ix}$, $\boldsymbol{W}^{(B)}_{fx}$, $\boldsymbol{W}^{(B)}_{ox}$	64×32	$\mathcal{N}(0, \sqrt{\frac{2}{64+32}})$	True
$\boldsymbol{W}^{(B)}_{gh}$, $\boldsymbol{W}^{(B)}_{ih}$, $\boldsymbol{W}^{(B)}_{fh}$, $\boldsymbol{W}^{(B)}_{oh}$	32×32	$\mathcal{N}(0, \sqrt{\frac{2}{32+32}})$	True
$\boldsymbol{\alpha}^{(A)}_{g}$, $\boldsymbol{\alpha}^{(A)}_{i}$, $\boldsymbol{\alpha}^{(A)}_{f}$, $\boldsymbol{\alpha}^{(A)}_{o}$, $\boldsymbol{\alpha}^{(A)}_{c}$	64×1	$\mathbf{1}$	True
$\boldsymbol{\beta}^{(A)}_{g}$, $\boldsymbol{\beta}^{(A)}_{i}$, $\boldsymbol{\beta}^{(A)}_{f}$, $\boldsymbol{\beta}^{(A)}_{o}$, $\boldsymbol{\beta}^{(A)}_{c}$	64×1	$\mathbf{0}$	False
$\boldsymbol{\alpha}^{(B)}_{g}$, $\boldsymbol{\alpha}^{(B)}_{i}$, $\boldsymbol{\alpha}^{(B)}_{f}$, $\boldsymbol{\alpha}^{(B)}_{o}$, $\boldsymbol{\alpha}^{(B)}_{c}$	32×1	$\mathbf{1}$	True
$\boldsymbol{\beta}^{(B)}_{g}$, $\boldsymbol{\beta}^{(B)}_{i}$, $\boldsymbol{\beta}^{(B)}_{f}$, $\boldsymbol{\beta}^{(B)}_{o}$, $\boldsymbol{\beta}^{(B)}_{c}$	32×1	$\mathbf{0}$	False
$\boldsymbol{W}_{softmax}$	32×2	$\mathcal{N}(0, 0.01)$	True
$\boldsymbol{b}_{softmax}$	1×2	$\mathbf{0}$	False
Trainable Parameters = 93,698			

Table 1. All trainable parameters, along with their dimensions, initialization method and regularization. (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.

Except for weight decay, dropout (N. Srivastava and Salakhutdinov 2014) is also used as a measure to counter overfitting and increase the generality of our model. Although dropout may

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

Electronic copy available at: https://ssrn.com/abstract=3202313

also be applied to recurrent layers (S. Semeniuta and Barth 2016), we found that it was decreasing the memory capacity of the network. Thus, dropout is applied only to non-recurrent layers, namely before data enters the first LSTM cell and before it enters the softmax layer.

Features are split into two subgroups in the beginning of each training iteration, the stock and the non-stock [1] 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 datasets of stock, competitor, sector, universe, correlated stocks, 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 time, 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 algorithm handle the selection of features from within the other datasets, 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 (G. Hinton and 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 $\boldsymbol{G^{(t)}}$, the update rule for the Moving Average of the Squared Gradient (MASG) is defined as:

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

Now using also a momentum of m and learning rate η, we can define the update rule for parameters $\boldsymbol{\theta}$ as:

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

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 $d = 0.9$, a momentum of $m = 0.9$.

As discussed in 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 back-propagation 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 unstable training. The solution for this problem is to define a threshold G_{max}, above which, all gradients are reduced to it.

[1] Competitor, sector, correlated stocks, time-features

Electronic copy available at: https://ssrn.com/abstract=3202313

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

Although, by observing the training process, we found that gradients were rarely greater than 1, we set a max 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:

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

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

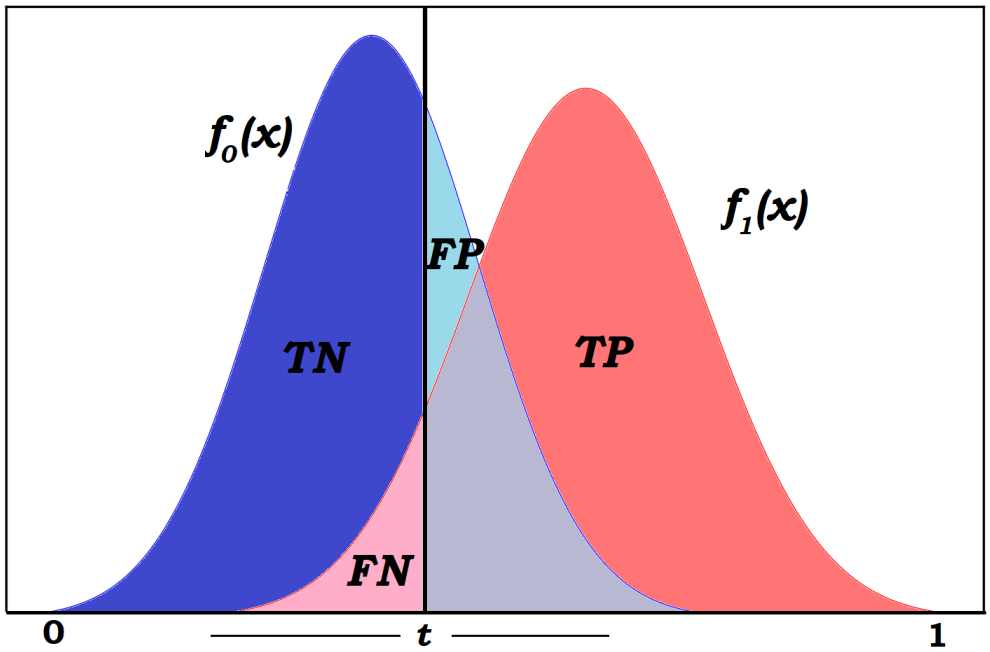


Figure 4. False Positive Rate and True Positive Rate. By varying threshold t over the range $[0, 1]$, the FPR is the ratio of the area covered by FP to the area defined by $f_0(x)$. On the other hand, the TPR is the ratio of the area covered by TP to the area defined by $f_1(x)$. Note that the grey area is part of both FP and TP. Also, it has to be noted that this example and 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.

By plotting the TPR against FPR, for the varying decision threshold t, the ROC curve is obtained and the AUC score is the area defined by this curve. This translates to following integration:

Electronic copy available at: https://ssrn.com/abstract=3202313

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

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 miss-classification 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 at 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

$$p_{S,t}^{(ensemble)} = \frac{1}{12} \sum_{i=1}^{12} p_{S,t}^{(model_i)}, \tag{19}$$

where $p_{S,t}^{(model_i)}$ 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 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 dataset, the process works in an online way, by evaluating the models and then predicting the next observation:

Electronic copy available at: https://ssrn.com/abstract=3202313

$$p_{S,t}^{(ensemble)} = \sum_{i=1}^{12} \frac{AUC_{S,t}^{(model_i)}}{\sum_{k=1}^{12} AUC_{S,t}^{(model_k)}} p_{S,t}^{(model_i)}, \tag{20}$$

where $AUC_{S,t}^{(model_i)}$ indicates the AUC score of $model_i$ in period $(t - n_{valid} - 1, t - 1)$. This type of ensemble will be refereed to as 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 he highest AUC score and 0 for all the rest.

$$p_{S,t}^{(ensemble)} = argmax_{\boldsymbol{AUC}_{S,t}} \ \boldsymbol{p}_{S,t} \tag{21}$$

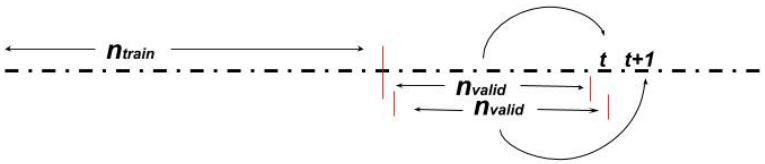


Figure 5. Predictions in an online way. The Performance Weighted Ensemble and Best Model Ensemble work in an online way, by evaluating the 12 models on n_valid and adjusting the weights of the models for predicting the next observation.

Our intuition is that each one of the 12 models for has identified and learned certain patterns from the training set. Whether these patterns can be found also in the test set, or they 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 $n_{valid} = 390$, which corresponds to one 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, first two weeks are used for training and predictions are made on the last week.

Electronic copy available at: https://ssrn.com/abstract=3202313

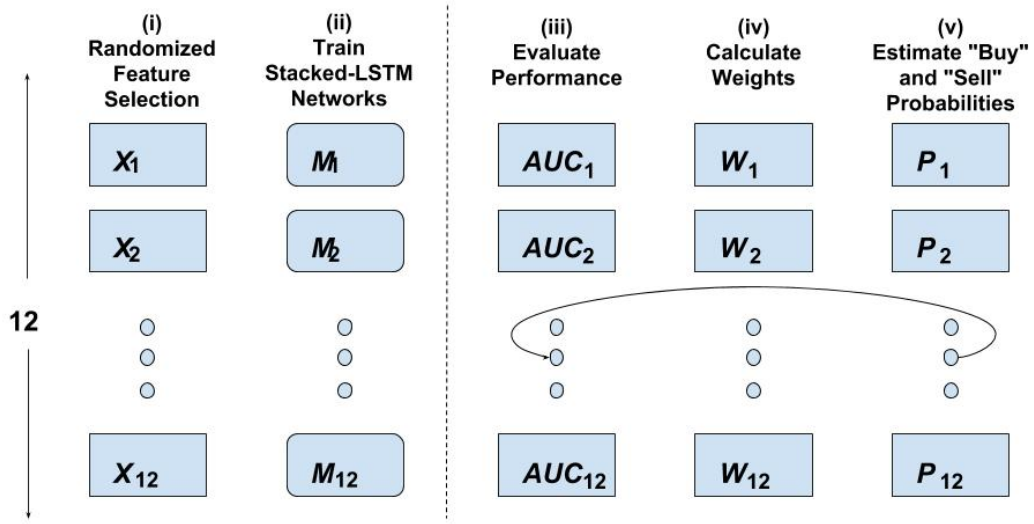


Figure 6. Ensemble.

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

(ii) Train models $M_i, i \in [1, 12]$ using datasets $X_i^{(train)}$. The fact that each train 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 $\boldsymbol{\mathcal{X}^{(test)}}$ and the $AUC_i^{(n_{valid}+1)}$ is calculated.

(iv) The weighting factors $W_i^{(n_{valid}+1)}$ of each model M_i in the ensemble are calculated. As indicated in equation 20, the weight of each model for predicting $\boldsymbol{\mathcal{Y}}_{\boldsymbol{n_{valid}+1}}^{\boldsymbol{(test)}}$ is fraction of $AUC_i^{(n_{valid}+1)}$ 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 $\boldsymbol{\mathcal{Y}}_{\boldsymbol{n_{valid}+1}}^{\boldsymbol{(test)}}$, 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.

Steps (iii) to (v) are repeated for every observation in $[n_{valid}+1, n_{valid}+n_{test}]$, 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.

4.5. *Benchmarks*

To provide a benchmark for the performance of the ensemble, we used two additional methods, the Lasso and the Ridge classifiers. They essential are regularized logistic regressions, with $\ell 1$ weight decay applied for Lasso and $\ell 2$ applied for the 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 unregularized logistic regression can be defined in the following equation:

Electronic copy available at: https://ssrn.com/abstract=3202313

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

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 are the following:

$$\boldsymbol{w}^*, b^* = argmin_{\boldsymbol{w},b} \sum_{i=1}^{n} log(e^{-\boldsymbol{y}_i \boldsymbol{X}_i^T \boldsymbol{w}+b} + 1) + \mathcal{L}_{reg}, \tag{23}$$

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 $\mathcal{L}_{reg} = \lambda||\boldsymbol{w}||_1$ for Lasso and $\mathcal{L}_{reg} = \lambda||\boldsymbol{w}||_2$ for Ridge.

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 need for the algorithm to hypothetically [1] run over all the instances of the trains set. Given a mini-batch B and a training set D, one epoch is equal to $\frac{|D|}{|B|} = \frac{1560}{32} \approx 49$. Consequently, 15 epoch amounts to $15 \cdot 49 = 735$ training iterations. The training iterations is the only stopping criteria. 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 weighting scheme of the ensemble is responsible for selecting "good" and "bad" models.

12 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 ensembles. The first two are the Equally Weighted Ensemble and the Performance Weighted Ensemble, as described in 4.4. To demonstrate that the predictability comes from the collective presence of the models, we also produced an ensemble, where for predicting each observation, we considered only the best model (according to the AUC of the past 390 observations). In the following figure, 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, or 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.

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

Electronic copy available at: https://ssrn.com/abstract=3202313

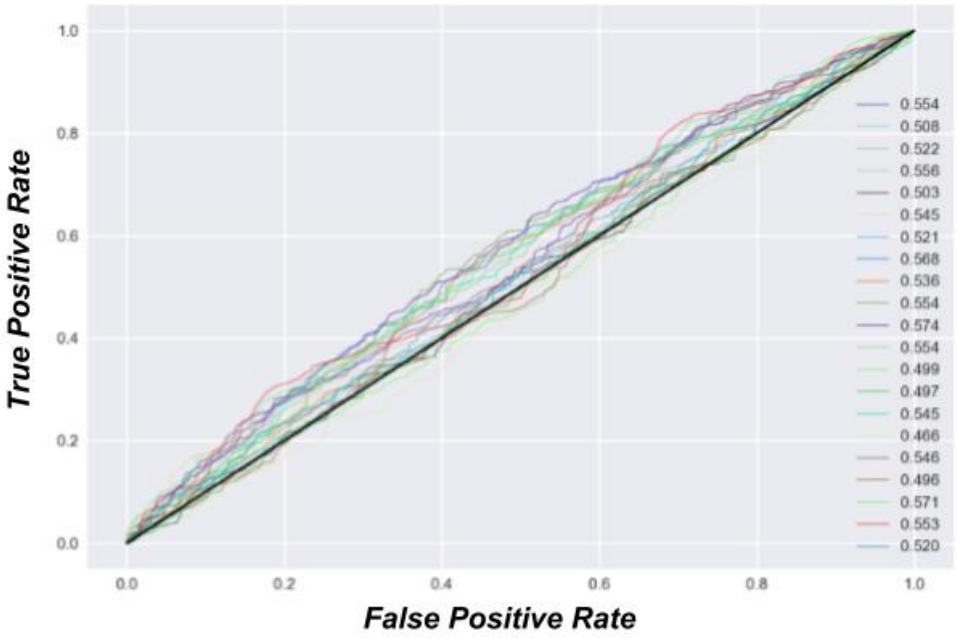


Figure 7. The Receiver Operating Characteristic (ROC) curve for 21 periods for stock NEE. Each line is obtained by plotting the False Positive Rate(FPR) against the True Positive Rate(TPR) for a varying decision threshold for each period, thus resulting to 21 ROC curves. The identity line stands for the random classifier (area of 0.5). The majority of the lines remain consistently above the identity line and indicate a better-than-random classifier. Additionally, in most models, the area defined by the ROC curve increases in middle thresholds (0.2 - 0.8), suggesting that the optimal decision threshold exists in that area.

5.2. *Lasso and Ridge Logistic Regressions*

To provide another benchmark for the Performance Weighted Ensemble, other than the random guessing, the Lasso logistic regression and the 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 vs Benchmarks*

In the following table, we provide the 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 [1] and tested for improvement in the produced AUC scores. The results with the inclusion of the extra models, along with the paired t-test can be be found in Table 5.

[1]Now the ensemble is comprised of the 12 stacked-LSTMs, the Lasso LR and the Ridge LR.

Electronic copy available at: https://ssrn.com/abstract=3202313

Table 2. AUC scores for the LSTM ensembles. Average AUC scores for the 22 stocks. 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 $\alpha = 0.01$. One asterisk indicates significance only at $\alpha = 0.1$, double asterisk indicates significance at $\alpha = 0.05$, while the triple one indicates significance at $\alpha = 0.01$. The ensemble, which is comprised from only the best model at each time, produced inferior results than the other two, in both absolute terms and significance.

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)***

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, that 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.

Electronic copy available at: https://ssrn.com/abstract=3202313

Table 3. AUC scores for Lasso and Ridge Logistic Regressions. 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 the parenthesis indicates whether the result is significantly better than random (AUC = 0.5). One asterisk indicates significance only at $\alpha = 0.1$, double asterisk indicates significance at $\alpha = 0.05$, while the triple one indicates significance at $\alpha = 0.01$. 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.

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)***

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 Long Short Term Memory neural networks through a robust, performance-evaluated, weighting method.

We conducted a large empirical study on the basis 44 US large cap stocks, which can be translated to 30GB 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 datasets that were used in the predictive modeling. The data were used to train a total of 5544 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. It furthermore outperformed two simpler models, the Lasso and Ridge logistic regressions. Additionally, by showing that incorporating the simpler models into the ensemble, no significant gain was achieved, 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 guess, 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

Electronic copy available at: https://ssrn.com/abstract=3202313

Table 4. Performance Weighted Ensemble against the other methods. 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 parenthesis 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 3 methods. One plus sign indicates that the Performance Weighted Ensemble is significantly better than the benchmark at $\alpha = 0.1$, double plus sign indicates that it is significantly better at $\alpha = 0.05$, while the triple one indicates that it is significantly better at $\alpha = 0.01$.

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**

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, while the 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 used computational time, 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, but also fast predictions. 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, for which an optimal value was shown to exist between 0.2 and 0.8.

Electronic copy available at: https://ssrn.com/abstract=3202313

Table 5. Inclusion of Lasso and Ridge in the ensemble. AUC scores of the Performance Weighted Ensemble containing only stacked-LSTMs and the the Performance Weighted Ensemble containing skacked-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 $\alpha = 0.25$.

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

References

A. N. Refenes, A.Z. and Francis, G., Stock Performance Modeling Using Neural Networks: A Comparative Study With Regression Models. *Neural Networks*, 1994, **7**, 375–388.

Bengio, Y., Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2009, **2**.

Bengio, Y., Practical recommendations for gradient-based training of deep architectures. *CoRR*, 2012, **abs/1206.5533**.

D. E. Baestaens, W.M.V.D.B. and Vaudrey, H., Options as a predictor of common stock price changes. *The European Journal of Finance*, 1995, **1**, 325–343.

D. M. Q. Nelson, A. C. M. Pereira, R.A.d.O., Stock Markets Price Movement Prediction With LSTM Neural Networks. *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017.

Fama, E.F., Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, 1969 (May, 1970), **25**, 383–417.

Fischer, T. and Krauss, C., Deep learning with long short-term memory networks for Financial market predictions. *AU Discussion Papers in Economics*, 2017, **11**.

G. Hinton, N.S. and Swersky, K., Neural Networks for Machine Learning, Lecture 6, University of Toronto. , 2012.

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

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

J. L. Ba, J.R.K. and Hinton, G.E., Layer Normalization. *CoRR*, 2016, **abs/1607.06450v1**.

Kennedy, J. and Eberhart, R., Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 1995, **4**, 1942–1948 vol.4.

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

Electronic copy available at: https://ssrn.com/abstract=3202313

N. Srivastava, G. Hinton, A.K.I.S. and Salakhutdinov, R., Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 2014, **15**, 1929–1958.

O. Hegazy, O.S.S. and Salam, M.A., A Machine Learning Model for Stock Market Prediction. *International Journal of Computer Science and Telecommunications*, 2013, **4**.

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

Qu, H. and Zhang, Y., A New Kernel of Support Vector Regression for Forecasting High-Frequency Stock Returns. *Mathematical Problems in Engineering*, 2016.

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

S. Semeniuta, A.S. and Barth, E., Recurrent Dropout without Memory Loss. *CoRR*, 2016, **abs/1603.05118**.

W. Bao, J.Y. and Rao, Y., A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 2017, **12**, 1–24.

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

Electronic copy available at: https://ssrn.com/abstract=3202313

Appendix A: Tables

Table A1. Stock Universe. Stocks used for this research , along with the sector they belong, their primary competitor and the exchange that they are traded at.

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
McDonalds 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

Table A2. Basic features. Features extracted from the raw trade data. Formulas indicate how these features are extracted from the split into 5-minute intervals datasets. 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.

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 ($StdPriceD$)	$\sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}(\Delta P_i - MeanPriceD)^2}$
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 ($Volume^0$)	$\sum_{i=1}^N \boldsymbol{I}_{\Delta P_i=0} S_i$

Electronic copy available at: https://ssrn.com/abstract=3202313

Table A3. Advanced features. Here we present a summary of all the features in each dataset and how many different versions of them are included on stock level and on sector/index level.

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

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 $\mathrm{SMA}(x_t, n)$ is composed from the mean of the last n observations of time-series x_t.

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

The Exponential Weighed Moving Average $\mathrm{EWMA}(x, \alpha)$ is calculated by exponential decreasing the weight of observations x_i with respect to their distance from x_t.

$$\mathrm{EWMA}(x_t, \alpha) = \alpha x_t + (1-\alpha)\mathrm{EWMA}(x_{t-1}, \alpha) \tag{B2}$$

B.2. *Sharpe Ratio*

Sharpe ratio is 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 $n = [6, 12, 36, 78, 234]$.

Electronic copy available at: https://ssrn.com/abstract=3202313

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

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.

$$\text{CLV}_t = \frac{(P_{close,t} - P_{low,t}) - (P_{high,t} - P_{close,t})}{P_{high,t} - P_{low,t}} \qquad \text{ADI}_t = \sum_{i=1}^{t} \text{CLV}_t \, Volume_i \tag{B4}$$

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 $n = [6, 12, 36, 78, 234]$.

$$MiddleBB_t = \text{SMA}(P_{typical,t}, n) \tag{B5}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i}^{n} P_{typical,t-i} - MiddleBB_t} \tag{B6}$$

$$UpperBB = MiddleBB_t + d\,\sigma \qquad LowerBB = MiddleBB_t - d\,\sigma \tag{B7}$$

$$\%b = \frac{P_{close,t} - LowerBB_t}{UpperBB_t - LowerBB_t} \qquad Bandwith_t = \frac{UpperBB_t - LowerBB_t}{MiddleBB_t} \tag{B8}$$

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.

$$K_t = \frac{P_{close,t} - L_n}{H_n - L_n} \tag{B9}$$

Where $L_n = min(P_{low,t-n}, ..., P_{low,t})$ and $H_n = min(P_{high,t-n}, ..., P_{high,t})$.
Finally, the Stochastic Oscillator is a Simple Moving Average of length m.

Electronic copy available at: https://ssrn.com/abstract=3202313

$$StochOsci_t = 100\,\mathrm{SMA}(K_t, m) \tag{B10}$$

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

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 $n = [6, 12, 36, 78, 234]$.

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

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

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

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, while large hint the opposite. The CCI is calculate for $n = [6, 12, 36, 78, 234]$.

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

Where MAD is the Mean Absolute Deviation, calculated as:

$$\mathrm{MAD}(x_t, n) = \frac{1}{n} \sum_{i=1}^{n} |x_{t-i} - \frac{1}{n} \sum_{k=1}^{n} x_{t-k}| \tag{B13}$$

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 used a measure of trend strength. The ADX is calculate for $n = [6, 12, 36, 78, 234]$.

$$UpMove_t = P_{high,t} - P_{high,t-1} \qquad DownMove_t = P_{low,t-1} - P_{low,t} \tag{B14}$$

Electronic copy available at: https://ssrn.com/abstract=3202313

$$DM_t^+ = \begin{cases} UpMove_t & , UpMove_t > DownMove_t \text{ and } UpMove_t > 0 \\ 0 & , \text{otherwise} \end{cases}$$

$$DM_t^- = \begin{cases} DownMove_t & , DownMove_t > UpMove_t \text{ and } DownMove_t > 0 \\ 0 & , \text{otherwise} \end{cases}$$

$$DI_t^+ = 100\frac{\text{EWMA}(DM_t^+, 1/n)}{\text{SMA}(TR_t, n)} \qquad DI_t^- = 100\frac{\text{EWMA}(DM_t^-, 1/n)}{\text{SMA}(TR_t, n)} \tag{B15}$$

Where TR stands for True Range and is defined as

$$TR_t = max(P_{high,t} - P_{low,t}, |P_{high,t} - P_{close,t-1}|, |P_{low,t} - P_{close,t-1}|) \tag{B16}$$

Finally, ADX is defined as

$$ADX_t = 100\ \text{EWMA}(\frac{|DI_t^+ - DI_t^-|}{|DI_t^+ + DI_t^-|}, 1/n) \tag{B17}$$

B.9. *Double and Triple Exponentially Smoothed Returns*

Both these indicators are calculate as the percentage change of the double DIX or triple TRIX exponentially smoothed prices. DIX is calculated for $n = 234$, while TRIX for $n = [6, 12, 36, 78]$.

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

B.10. *Moving Average Convergence-Divergence (MACD)*

MACD is a technical indicator composed from 3 separate time-series, used for detection changes in the trend of a price, with respect to strength, momentum, duration nd direction. First, the divergence of two EWMA 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 $n_3 = [2, 4, 12, 26, 78]$.

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

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

Electronic copy available at: https://ssrn.com/abstract=3202313

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 $n = [6, 12, 36, 78, 234]$.

$$MF_t = P_{typical,t} Volume_t \tag{B21}$$

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.

$$MF_t^+ = \sum_{i=1}^{n} MF_i \boldsymbol{I}_{P_{typical,t-i} - P_{typicalt-i-1} > 0} \tag{B22}$$

$$MF_t^- = \sum_{i=1}^{n} MF_i \boldsymbol{I}_{P_{typical,t-i} - P_{typicalt-i-1} < 0} \tag{B23}$$

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.

$$MFI_t = 100 \frac{MF_t^+}{MF_t^+ + MF_t^-} \tag{B24}$$

B.12. *Price Disagreement and Polarity*

Disagreement and Polarity are measures frequently used in sentiment analysis, calculated by the amount 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 A2. These indicators are calculated for $n = [0, 6, 12, 36, 78, 234]$.

$$Disagreement_t = |1 - |\frac{\sum_{i=1}^{n} Volume_{t-i}^+ - \sum_{i=1}^{n} Volume_{t-i}^-}{\sum_{i=1}^{n} Volume_{t-i}^+ + \sum_{i=1}^{n} Volume_{t-i}^-}|| \tag{B25}$$

$$Polarity_t = \frac{\sum_{i=1}^{n} Volume_{t-i}^+ - \sum_{i=1}^{n} Volume_{t-i}^-}{\sum_{i=1}^{n} Volume_{t-i}^0} \tag{B26}$$

Electronic copy available at: https://ssrn.com/abstract=3202313