Finance Research Letters 46 (2022) 102280

Contents lists available at ScienceDirect

Finance Research Letters

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

Forecasting directional movements of stock prices for intraday trading using LSTM and random forests☆

Pushpendu Ghosh [a,1], Ariel Neufeld [b,*,1], Jajati Keshari Sahoo [c,1]

[a] Department of Computer Science & Information Systems, BITS Pilani K.K. Birla Goa campus, India
[b] Division of Mathematical Sciences, Nanyang Technological University, Singapore
[c] Department of Mathematics, BITS Pilani K.K. Birla Goa campus, India

ARTICLE INFO

Keywords:
Random forest
LSTM
Forecasting
Statistical arbitrage
Machine learning
Intraday trading

ABSTRACT

We employ both random forests and LSTM networks (more precisely CuDNNLSTM) as training methodologies to analyze their effectiveness in forecasting out-of-sample directional movements of constituent stocks of the S&P 500 from January 1993 till December 2018 for intraday trading. We introduce a multi-feature setting consisting not only of the returns with respect to the closing prices, but also with respect to the opening prices and intraday returns. As trading strategy, we use Krauss et al. (2017) and Fischer and Krauss (2018) as benchmark. On each trading day, we buy the 10 stocks with the highest probability and sell short the 10 stocks with the lowest probability to outperform the market in terms of intraday returns — all with equal monetary weight. Our empirical results [2] show that the multi-feature setting provides a daily return, prior to transaction costs, of 0.64% using LSTM networks, and 0.54% using random forests. Hence we outperform the single-feature setting in Fischer and Krauss (2018) and Krauss et al. (2017) consisting only of the daily returns with respect to the closing prices, having corresponding daily returns of 0.41% and of 0.39% with respect to LSTM and random forests, respectively.[3]

1. Introduction

In the last decade, machine learning methods have exhibited distinguished development in financial time series prediction. Huck (2009) and Huck (2010) construct statistical arbitrage strategies using Elman neural networks and a multi-criteria-decision method. Takeuchi and Lee (2013) evolve a momentum trading strategy. Papaioannou et al. (2017) develop a trend following trading strategy to forecast and trade S&P 500 futures contracts. Tran et al. (2018), Sezer and Ozbayoglu (2018), and Singh et al. (2020) use neural networks for predicting time series data, whereas Borovykh et al. (2018) and Xue et al. (2018) employ convolutional neural networks. Moreover, Mallqui and Fernandes (2019) employ artificial neural networks, support vector machines, and ensemble algorithms to predict the direction as well as the maximum, minimum and closing prices of Bitcoin. We also refer to Harikrishnan et al. (2021) for a survey paper regarding the application of various machine learning algorithms for the prediction of stock prices.

☆ Acknowledgment: The first author gratefully acknowledges the NTU-India Connect Research Internship Programme which allowed him to carry out part of this research project while visiting the Nanyang Technological University, Singapore. The second author gratefully acknowledges financial support by his NAP Grant *Machine Learning based Algorithms in Finance and Insurance*.

* Corresponding author.
E-mail addresses: f20150366@goa.bits-pilani.ac.in (P. Ghosh), ariel.neufeld@ntu.edu.sg (A. Neufeld), jksahoo@goa.bits-pilani.ac.in (J.K. Sahoo).

[1] All authors contributed equally to the paper. The author appearance is in alphabetical order.

[2] All the codes are available on https://github.com/pushpendughosh/Stock-market-forecasting

[3] Fischer and Krauss (2018) and Krauss et al. (2017) obtain 0.46% and 0.43%, as the period from November 2015 until December 2018 was not included in their backtesting.

https://doi.org/10.1016/j.frl.2021.102280
Received 28 June 2020; Received in revised form 23 May 2021; Accepted 29 June 2021
Available online 3 July 2021

In this paper, we focus on the application of long short-term memory networks (LSTM) and random forests to forecast directional movements of stock prices. Siami-Namini and Namin (2018) compare LSTM with an autoregressive integrated moving average (ARIMA) model. Their empirical results applied to financial data show that LSTM outperforms ARIMA in terms of lower forecast errors and higher accuracy. The empirical results in Qiu et al. (2020) using LSTM demonstrate an improvement for stock price predictions when an attention mechanism is employed. Sharma et al. (2021) observe that for both LSTM and autoregressive integrated moving average with exogenous variables (ARIMAX) models a considerable improvement of the prediction of stock price movements can be achieved when including a sentiment analysis. Moritz and Zimmermann (2014) employ random forests to predict returns of stocks using US CRSP data. As trading strategy, the top decile is bought, whereas the bottom one is sold short. Lohrmann and Luukka (2019) construct a classification model using random forest to predict open-to-close returns of stocks in the S&P 500. Basak et al. (2019) use random forests and gradient boosted decision trees (XGBoost), together with a selection of technical indicators, to analyze the performance for medium to long-run prediction of stock price returns, and Sadorsky (2021) employs random forests to forecast the directions of stock prices of clean energy exchange traded funds.

Moreover, Krauss et al. (2017) compare different deep learning methods such as deep neural networks, gradient-boosted-trees and random forests. In a single-feature setting, the daily returns with respect to the closing prices of the S&P 500 from December 1992 until October 2015 are provided to forecast one-day-ahead for every stock the probability of outperforming the market. As trading strategy, the 10 stocks with the highest probability are bought and the 10 stocks with the lowest probability are sold short — all with equal monetary weight. It turns out that random forests achieve the highest return of each of the above deep learning methods with returns of 0.43% per day, prior to transaction costs. Fischer and Krauss (2018) continue the study of Krauss et al. (2017) by employing LSTM networks as deep-learning methodology and obtain returns of 0.46% per day prior to transaction costs, therefore outperforming all the memory-free methods in Krauss et al. (2017).

In our work, we use the results in Krauss et al. (2017) and Fischer and Krauss (2018) as benchmark for the ease of comparison. We introduce a multi-feature setting consisting not only of the returns with respect to the closing prices, but also with respect to the opening prices and intraday returns to predict for each stock, at the beginning of each day, the probability to outperform the market in terms of intraday returns. As data set we use all stocks of the S&P 500 from the period of January 1990 until December 2018. We employ both random forests on the one hand and LSTM networks (more precisely CuDNNLSTM) on the other hand as training methodology and apply the same trading strategy as in Krauss et al. (2017) and Fischer and Krauss (2018). Our empirical results show that the multi-feature setting provides a daily return, prior to transaction costs, of 0.64% for the LSTM network, and 0.54% for the random forest, hence outperforming the single-feature setting in Fischer and Krauss (2018) and Krauss et al. (2017), having corresponding daily returns of 0.41% and of 0.39%, respectively. [4]

The remainder of this paper is organized as follows. In Section 2 we explain the data sample as well as the software and hardware we use. In Section 3 we discuss the methodology we employ. The empirical results are then presented in Section 4.

2. Data and technology

We collected adjusted closing prices and opening prices of all constituent stocks of the S&P 500 from the period of January 1990 until December 2018 using Bloomberg. For each day, stocks with zero volume were not considered for trading at this day.

All experiments were executed in a NVIDIA Tesla V100 with 30 GB memory. The codes and simulations were implemented using Python 3.6.5 with a dependency of TensorFlow 1.14.0 and scikit-learn 0.20.4. Visualization and statistical values were produced and calculated using the financial toolbox of MATLAB R2016b.

3. Methodology

Our methodology is composed of five steps. In the first step, we divide our raw data into study periods, where each study period is divided into a training part (for in-sample trading), and a trading part (for out-of-sample predictions). In the second step, we introduce our features, whereas in the third step we set up our targets. In the forth step, we define the setup of our two machine learning methods we employ, namely random forest and CuDNNLSTM. Finally, in the fifth step, we establish a trading strategy for the trading part.

3.1. Dataset creation with non-overlapping testing period

We follow the procedure of Krauss et al. (2017) & Fischer and Krauss (2018) and divide the dataset consisting of 29 years starting from January 1990 till December 2018, using a 4-year window and 1-year stride, where each study period is divided into a training period of approximately 756 days ($\approx$ 3 years) and a trading period of approximately 252 days ($\approx$ 1 year). As a consequence, we obtain 26 study periods with non-overlapping trading part (see Fig. 1).

[4] Fischer and Krauss (2018) and Krauss et al. (2017) obtain 0.46% and 0.43%, as the period from November 2015 until December 2018 was not included in their backtesting.

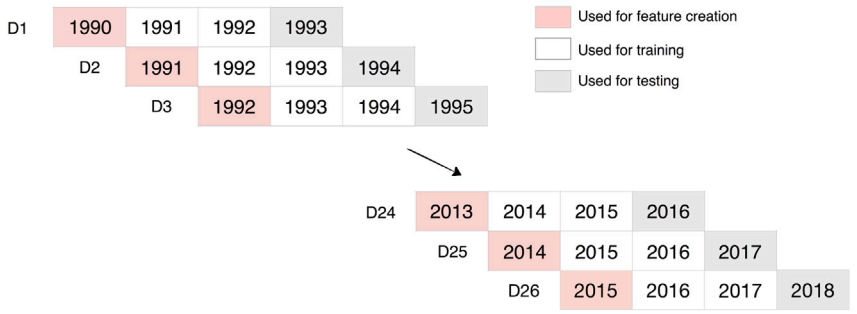

Fig. 1. Dataset creation with non-overlapping testing period.

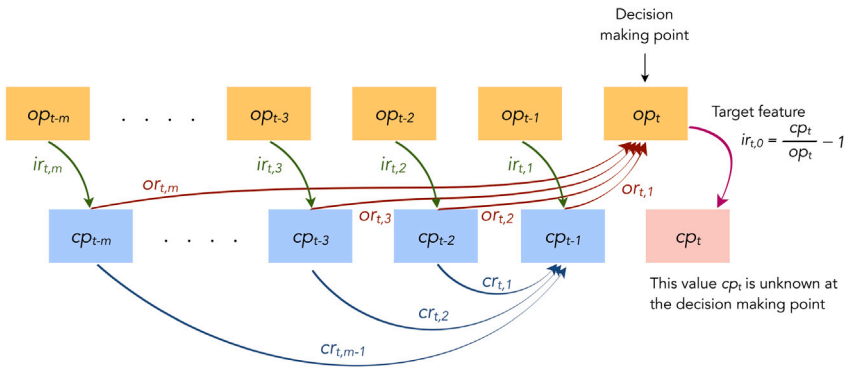

Fig. 2. Feature generation for random forest.

3.2. Features selection

Let T_{study} denote the amount of days in a study period and let n_i be the number of stocks[5] s in $\boldsymbol{S}$ having complete historical data available at the end of each study period i. Moreover, we define the adjusted closing price and opening price of any stock $s \in \boldsymbol{S}$ at time t by $cp_t^{(s)}$ and $op_t^{(s)}$ respectively.

Given a prediction day $t := \tau$, we have the following inputs and prediction tasks:

Input: We have the historical opening prices, $op_t^{(s)}$, $t \in \{0, 1, \dots, \tau - 1, \tau\}$, (including the prediction day's opening price $op_\tau^{(s)}$) as well as the historical adjusted closing prices $cp_t^{(s)}$, $t \in \{0, 1, \dots, \tau - 1\}$, (excluding the prediction day's closing price, $cp_\tau^{(s)}$).

Task: Out of all n stocks, predict k stocks with the highest and k stocks with the lowest intraday return $ir_{\tau,0} := \frac{cp_\tau}{op_\tau} - 1$.

3.2.1. Feature generation for random forest

For any stock $s \in \boldsymbol{S}$ and any time $t \in \{241, 242, \dots, T_{study}\}$, the feature set we provide to the random forest comprises of the following three signals:

1. Intraday returns: $ir_{t,m}^{(s)} := \frac{cp_{t-m}^{(s)}}{op_{t-m}^{(s)}} - 1$,
2. Returns with respect to last closing price: $cr_{t,m}^{(s)} := \frac{cp_{t-1}^{(s)}}{cp_{t-1-m}^{(s)}} - 1$,
3. Returns with respect to opening price: $or_{t,m}^{(s)} := \frac{op_t^{(s)}}{cp_{t-m}^{(s)}} - 1$,

where $m \in \{1, 2, 3, \dots, 20\} \cup \{40, 60, 80, \dots, 240\}$, obtaining 93 features, similar to Takeuchi and Lee (2013) & Krauss et al. (2017), who considered the single feature case consisting of the simple return $cr_{t,m}^{(s)} = \frac{cp_{t-1}^{(s)}}{cp_{t-1-m}^{(s)}} - 1$. By the choice of $m \in \{1, 2, 3, \dots, 20\} \cup \{40, 60, 80, \dots, 240\}$, we consider in the first month the corresponding returns of each trading day, whereas for the subsequent 11 months we only consider the corresponding multi-period returns of each month. No time series transformation such as, e.g., scaling or centering, is performed for the random forest (see Fig. 2).

[5] We include stock prices of all S&P500 constituents at the last day of the training data.

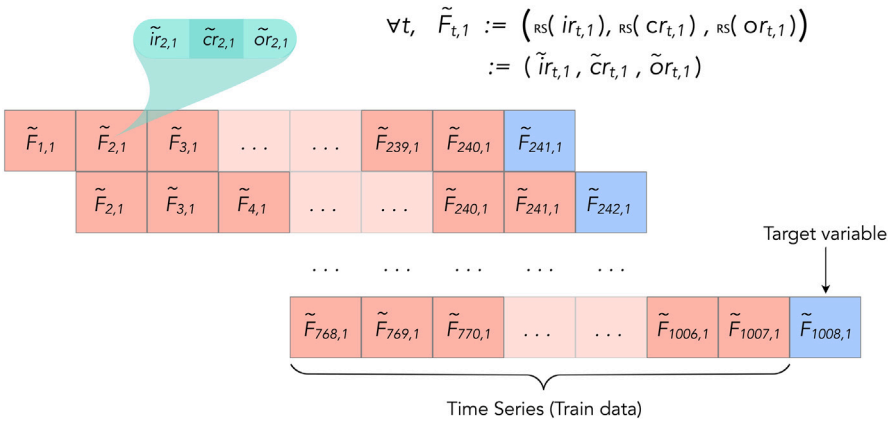

Fig. 3. Feature generation for LSTM.

3.2.2. Feature generation for LSTM

Following the approach of Fischer and Krauss (2018), but in a multi-feature setting rather than their single feature approach, we input the model with 240 time steps and 3 features, and train it to predict the direction of the 241st intraday return.

More precisely, for each stock s at time t, we first consider the following three features, $ir^{(s)}_{t,1}$, $cr^{(s)}_{t,1}$, $or^{(s)}_{t,1}$ defined above in Section 3.2.1. Then we apply the Robust Scaler standardization

$$\widetilde{f}^{(s)}_{t,1} := \frac{f^{(s)}_{t,1} - \mathrm{Q}_2(f^{(s)}_{\cdot,1})}{\mathrm{Q}_3(f^{(s)}_{\cdot,1}) - \mathrm{Q}_1(f^{(s)}_{\cdot,1})},$$

where $\mathrm{Q}_1(f^{(s)}_{\cdot,1})$, $\mathrm{Q}_2(f^{(s)}_{\cdot,1})$ and $\mathrm{Q}_3(f^{(s)}_{\cdot,1})$ are the first, second, and third quartile of $f^{(s)}_{\cdot,1}$, for each feature $f^{(s)}_{\cdot,1} \in \{ir^{(s)}_{\cdot,1}, cr^{(s)}_{\cdot,1}, or^{(s)}_{\cdot,1}\}$ in the respective training period.

The Robust Scaler standardization (Pedregosa et al., 2011) first subtracts (and hence removes) the median and then scales the data using the inter-quantile range, making it robust to outliers.

Next, for each time $t \in \{241, 242, \ldots, T_{study}\}$, we generate overlapping sequences of 240 consecutive, three-dimensional standardized features $\{\, \widetilde{F}^{(s)}_{t-239,1}, \widetilde{F}^{(s)}_{t-238,1}, \ldots, \widetilde{F}^{(s)}_{t,1} \,\}$, where $\widetilde{F}^{(s)}_{t-i,1} := (\widetilde{ir}^{(s)}_{t-i,1}, \widetilde{cr}^{(s)}_{t-i,1}, \widetilde{or}^{(s)}_{t-i,1})$, $i \in \{239, 238, \ldots, 0\}$ (see Fig. 3).

3.2.3. Train-test split

For each stock $s \in \boldsymbol{S}$, we create a matrix with M columns and T_{study} rows, where M is the number of features. Hence M is equal to 93 and (240, 3) when using random forest and LSTM, respectively. We fill the matrix with respective M features as defined above. Since by definition, $ir_{t,m}$ is not defined when $t \le m$, columns of the top 240 rows are partially filled and are hence removed. This removal leaves $T_{study} - 240$ rows (i.e. for $t = \{241, 242, 243, \ldots, T_{study}\}$) which is split into two parts, namely approximately from $t = 241$ to $t = 756$, and from $t = 757$ to $t = T_{study}$, for training and testing purposes, respectively. Note that typically $T_{study} = 1008$.

At the end, we concatenate the training data of all stocks in $\boldsymbol{S}$ to get the collective training set. Hence the training set is a matrix with approximate size of $500 \times 516 = 258000$ rows (instances) and M columns (features), along with their corresponding target, whereas the trading set is a matrix with approximate size of $500 \times 252 = 126000$ rows (instances) and M columns (features) (see Fig. 4).

3.3. Target selection

Following Takeuchi and Lee (2013) and Fischer and Krauss (2018) we divide each stock at time t into 2 classes of equal size, based on their intraday returns $ir^{(s)}_{t,0}$. Class 0 is realized if $ir^{(s)}_{t,0}$ of stock s is smaller than the cross-sectional median intraday return of all stocks at time t, whereas Class 1 is realized if $ir^{(s)}_{t,0}$ of stock s is bigger than the cross-sectional median intraday return of all stocks at time t.

3.4. Model training specification

3.4.1. Model specification for random forest

As first model, we use random forests introduced by Ho (1995) and expanded by Breiman (2001), with the following parameters:

- Number of decision trees in the forest = 1000
- Maximum depth of each tree[6] = 10

[6] Our empirical study from hyperparameter tuning suggests that forests with maximum depth of 10 give the highest accuracy.

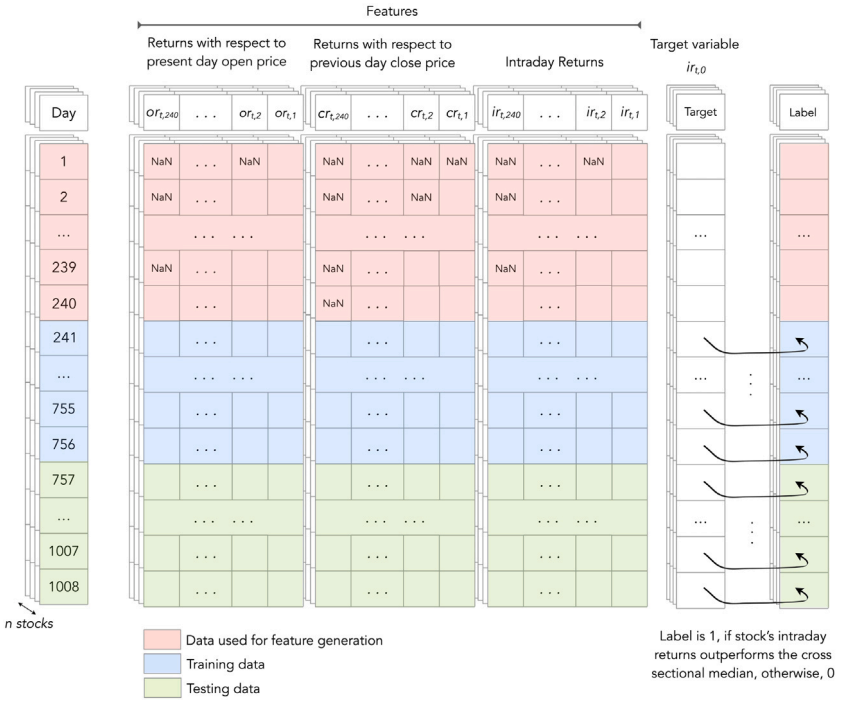

Fig. 4. Train-test split matrix.

- For every split, we select $m := \lfloor \sqrt{p} \rfloor$ features randomly from the $p = 93$ features in the data, see Pedregosa et al. (2011).

We refer to Krauss et al. (2017, Subsection 4.3.3) and Fischer and Krauss (2018, Subsection 3.4) for further details regarding random forests.

3.4.2. Model specification for LSTM

LSTM is a recurrent neural network introduced by Schmidhuber and Hochreiter (1997); we refer to Fischer and Krauss (2018) for a detailed description. Since the training of LSTMs is very time consuming, and also to efficiently utilize the power of GPUs, we perform our experiments using CuDNNLSTMs (Chetlur et al., 2014). CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library for deep neural networks. We gain immense speedup (up to 7.2x, see Braun (2018)) in training and predicting time. For the ease of comparison[7] we follow the network architecture used in Fischer and Krauss (2018). We create a model with 25 cells of CuDNNLSTM, followed by a dropout layer of 0.1 and then a dense layer of 2 output nodes with softmax activation function.

- Loss function: categorical cross-entropy
- Optimizer: RMSProp (with the keras default learning rate of 0.001)
- Batch size: 512
- Early stopping: patience of 10 epochs, monitoring the validation loss
- Validation split: 0.2.

3.5. Prediction and trading methodology

We forecast the probability $\mathcal{P}_t^{(s)}$ for each stock s to outperform the median intraday return $ir_{t,0}^{(s)}$. Next, as trading strategy, we follow Krauss et al. (2017) and Fischer and Krauss (2018) and go long the top $k = 10$ stocks with highest $\mathcal{P}_t^{(s)}$ and go short the worst $k = 10$ stocks with lowest $\mathcal{P}_t^{(s)}$ — all with equal monetary weight. Each long and short transaction are subjected to 0.05% slippage cost on each half-turn, as suggested by Avellaneda and Lee (2010), so each day's transaction is penalized with a total of 0.2%.

[7] We tested several values for the amount of cells, as well as several LSTM architectures. They only marginally influence the empirical results.

Table 1
Time comparison.

Metric	3-Feature IntraDay LSTM	3-Feature IntraDay RF	1-Feature NextDay LSTM	1-Feature NextDay RF	1-Feature IntraDay LSTM	1-Feature IntraDay RF
Time per epoch (in sec)	33.1	–	166	–	13.8	–
Training time (in min)	24.21	7.21	112.3	2.59	10.4	2.56
Decision making time (in sec)	0.086924	0.419563	0.180778	0.380040	0.036128	0.374121

Table 2
Average performance metrics of daily returns before transaction cost.

Metric of daily returns	3-Feature IntraDay LSTM	3-Feature IntraDay RF	1-Feature NextDay LSTM	1-Feature NextDay RF	1-Feature IntraDay LSTM	1-Feature IntraDay RF	SP500 Index
Mean (long)	0.00332	0.00273	0.00257	0.00259	0.00094	0.00104	0.00033
Mean (short)	0.00312	0.00266	0.00158	0.00130	0.00180	0.00187	0.00000
Mean return	0.00644	0.00539	0.00414	0.00389	0.00274	0.00290	0.00033
Standard error	0.00019	0.00020	0.00024	0.00023	0.00021	0.00021	0.00014
Minimum	−0.1464	−0.1046	−0.1713	−0.1342	−0.1565	−0.1487	−0.0903
Quartile 1	−0.0017	−0.0028	−0.0052	−0.0051	−0.0054	−0.0050	−0.0044
Median	0.00559	0.00462	0.00352	0.00287	0.00242	0.00221	0.00056
Quartile 3	0.01433	0.01306	0.01294	0.01161	0.01086	0.01036	0.00560
Maximum	0.14101	0.14153	0.19884	0.28139	0.13896	0.16064	0.11580
Share > 0	0.69663	0.65857	0.60598	0.59479	0.58405	0.58937	0.53681
Std. deviation	0.01572	0.01597	0.01961	0.01831	0.01713	0.01683	0.01133
Skewness	0.15599	0.28900	0.36822	1.41199	−0.1828	0.12051	−0.1007
Kurtosis	9.71987	8.32627	10.8793	19.8349	10.1893	11.7758	11.9396
1-percent VaR	−0.0352	−0.0364	−0.0492	−0.0432	−0.0461	−0.0448	−0.0313
1-percent CVaR	−0.0519	−0.0528	−0.0712	−0.0592	−0.0678	−0.0660	−0.0451
5-percent VaR	−0.0157	−0.0170	−0.0234	−0.0208	−0.0214	−0.0197	−0.0177
5-percent CVaR	−0.0284	−0.0297	−0.0401	−0.0345	−0.0377	−0.0357	−0.0270
Max. drawdown	0.22345	0.19779	0.42551	0.23155	0.35645	0.43885	0.56775
Avg return p.a.	3.84750	2.75103	1.68883	1.53806	0.91483	1.00281	0.06975
Std dev. p.a.	0.24957	0.25358	0.31135	0.29071	0.27193	0.26719	0.17990
Down dev. p.a.	0.17144	0.17301	0.21204	0.18690	0.19270	0.18530	0.12970
Sharpe ratio	6.34253	5.20303	3.22732	3.23339	2.39560	2.59188	0.24867
Sortino ratio	62.7403	49.6764	27.8835	30.0753	19.0217	21.2964	1.77234

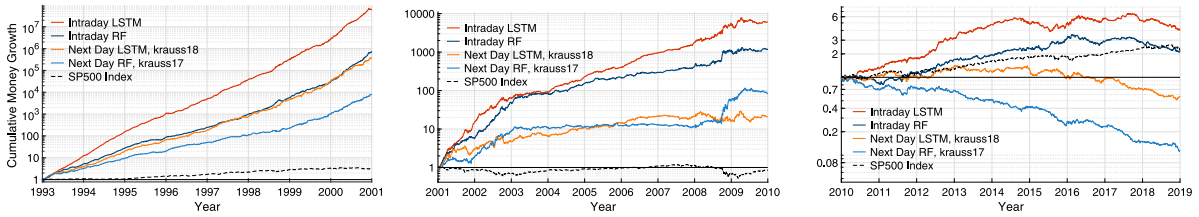

Fig. 5. Cumulative money growth with US$1 initial investment, after deducting transaction cost.

4. Results and discussion

The empirical results[8] show that our multi-feature setting consisting not only of the returns with respect to the closing prices, but also with respect to the opening prices and intraday returns, outperforms the single feature setting of Krauss et al. (2017) and Fischer and Krauss (2018), both with respect to random forests and LSTM. We refer to "IntraDay" for our setting and "NextDay" for the setting in Krauss et al. (2017) and Fischer and Krauss (2018) in Tables 1–3 and Figs. 5–7.

Indeed, our setting involving LSTM obtains, prior to transaction costs, a daily return of 0.64%, compared to the setting in Fischer and Krauss (2018) obtaining a 0.41% daily return. Also for random forests, our setting obtains a higher daily return of 0.54%, compared to 0.39% when using the setting as in Krauss et al. (2017). The share of positive returns is at 69.67% and 65.85% for LSTM and random forests. In addition, our setting obtains higher sharpe ratio and lower standard deviation (i.e. typical annualized risk-return metrics) in comparison with the one in Krauss et al. (2017) and Fischer and Krauss (2018). Furthermore, our setting produces a lower maximum drawdown and lower daily value at risk (VaR); we refer to Tables 2 and 3.

We also see that in our setting LSTM outperforms random forests, which is in line with the results of Fischer and Krauss (2018) showing that LSTM has an advantage compared to the memory-free methods analyzed in Krauss et al. (2017).

[8] All the codes are available on https://github.com/pushpendughosh/Stock-market-forecasting.

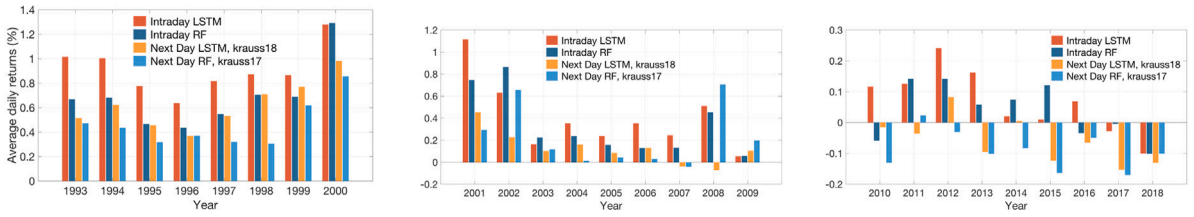

Fig. 6. Average of daily mean returns, after deducting transaction cost.

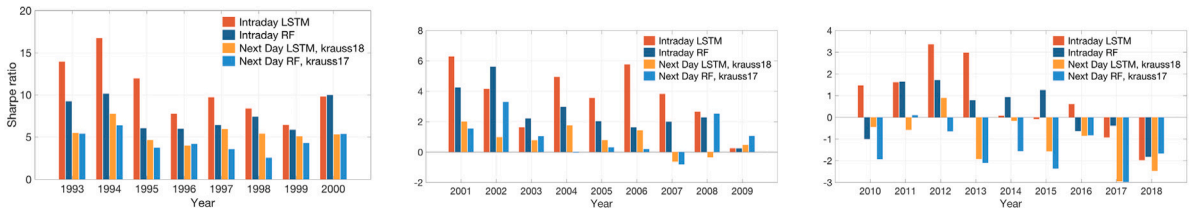

Fig. 7. Annualized sharpe ratio, after deducting transaction cost.

Table 3
Average performance metrics of daily returns after transaction cost.

Metric of daily returns	3-Feature IntraDay LSTM	3-Feature IntraDay RF	1-Feature NextDay LSTM	1-Feature NextDay RF	1-Feature IntraDay LSTM	1-Feature IntraDay RF	SP500 Index
Mean (long)	0.00232	0.00173	0.00157	0.00159	−0.0000	0.00004	0.00033
Mean (short)	0.00212	0.00166	0.00058	0.00030	0.00080	0.00087	0.00000
Mean return	0.00444	0.00339	0.00214	0.00189	0.00074	0.00090	0.00033
Standard error	0.00019	0.00020	0.00024	0.00023	0.00021	0.00021	0.00014
Minimum	−0.1484	−0.1066	−0.1733	−0.1362	−0.1585	−0.1507	−0.0903
Quartile 1	−0.0037	−0.0048	−0.0072	−0.0071	−0.0074	−0.0070	−0.0044
Median	0.00359	0.00262	0.00152	0.00087	0.00042	0.00021	0.00056
Quartile 3	0.01233	0.01106	0.01094	0.00961	0.00886	0.00836	0.00560
Maximum	0.13901	0.13953	0.19684	0.27939	0.13696	0.15864	0.11580
Share > 0	0.63129	0.59319	0.54279	0.53006	0.51534	0.50810	0.53681
Std. deviation	0.01572	0.01597	0.01961	0.01831	0.01713	0.01683	0.01133
Skewness	0.15599	0.28900	0.36822	1.41199	−0.1828	0.12051	−0.1007
Kurtosis	9.71987	8.32627	10.8793	19.8349	10.1893	11.7758	11.9396
1-percent VaR	−0.0372	−0.0384	−0.0512	−0.0452	−0.0481	−0.0468	−0.0313
1-percent CVaR	−0.0539	−0.0548	−0.0732	−0.0612	−0.0698	−0.0680	−0.0451
5-percent VaR	−0.0177	−0.0190	−0.0254	−0.0228	−0.0234	−0.0217	−0.0177
5-percent CVaR	−0.0304	−0.0317	−0.0421	−0.0365	−0.0397	−0.0377	−0.0270
Max. drawdown	0.39227	0.40139	0.87232	0.92312	0.97263	0.87123	0.56775
Avg return p.a.	1.94325	1.27179	0.63060	0.53901	0.16046	0.21146	0.06975
Std dev. p.a.	0.24957	0.25358	0.31135	0.29071	0.27193	0.26719	0.17990
Down dev. p.a.	0.17144	0.17301	0.21204	0.18690	0.19270	0.18530	0.12970
Sharpe ratio	4.32307	3.21553	1.60856	1.49969	0.54218	0.70557	0.24867
Sorting ratio	39.0873	27.9810	12.9274	12.7950	3.98288	5.34773	1.77234

In Figs. 5–7, we have divided the time period from January 1993 until December 2018 into three time-periods, analog to Fischer and Krauss (2018) and similar to Krauss et al. (2017). Roughly speaking, the first time-period corresponds to a strong performance caused by, among others, the dot-com-bubble, followed by the time-period of moderation with the bursting of the dot-com bubble and the financial crisis of 2008, ending with the time-period of deterioration; probably since by that time on, machine learning algorithms are broadly available and hence diminishes the opportunity of creating statistical arbitrage having a technological advantage. We refer to Krauss et al. (2017) and Fischer and Krauss (2018) for a detailed discussion of these sub-periods. We see in Figs. 5–7 that in each of these sub-periods, our setting outperforms the one in Krauss et al. (2017) and Fischer and Krauss (2018).

To show the importance of using three features instead of having a single feature, we additionally analyze in Tables 2 and 3 the performance in the case of intraday-trading, but where only intraday returns $ir_{t,m}^{(s)}$ as a single feature is used. The experimental results show massive improvement in all metrics when using the three features introduced in Section 3.2.

As an outlook for future research, we remark that the problem of forecasting directional movements of stocks (or currencies) can be formulated as a reinforcement learning problem, both model-free and model-based. An interesting analysis would be to compare these different reinforcement learning methodologies to both LSTM and random forests, particularly applied to cryptocurrencies. We leave this open for future studies.

References

Avellaneda, M., Lee, J.-H., 2010. Statistical arbitrage in the us equities market. Quant. Finance 10, 761–782.

Basak, S., Kar, S., Saha, S., Khaidem, L., Dey, S.R., 2019. Predicting the direction of stock market prices using tree-based classifiers. N. Am. J. Econ. Financ. 47, 552–567.

Borovykh, A., Bohte, S., Oosterlee, C.W., 2018. Dilated convolutional neural networks for time series forecasting. J. Comput. Finance 22 (4), 73–101.

Braun, S., 2018. LSTM Benchmarks for deep learning frameworks. preprint, arXiv:1806.01818.

Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E., 2014. Cudnn: Efficient primitives for deep learning. preprint, arXiv:1410.0759.

Fischer, T., Krauss, C., 2018. Deep learning with long short-term memory networks for financial market predictions. European J. Oper. Res. 270, 654–669.

Harikrishnan, R., Gupta, A., Tadanki, N., Berry, N., Bardae, R., 2021. Machine learning based model to predict stock prices: A survey. IOP Conf. Ser.: Mater. Sci. Eng. 1084, 012019.

Ho, T.K., 1995. Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition, Vol. 1. IEEE, pp. 278–282.

Huck, N., 2009. Pairs selection and outranking: An application to the S & P 100 index. European J. Oper. Res. 196, 819–825.

Huck, N., 2010. Pairs trading and outranking: The multi-step-ahead forecasting case. European J. Oper. Res. 207, 1702–1716.

Krauss, C., Do, X.A., Huck, N., 2017. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S & P 500. European J. Oper. Res. 259, 689–702.

Lohrmann, C., Luukka, 2019. Classification of intraday S & P500 returns with a random forest. Int. J. Forecast. 35 (1), 390–407.

Mallqui, D., Fernandes, R., 2019. Predicting the direction, maximum, minimum and closing prices of daily bitcoin exchange rate using machine learning techniques. Appl. Soft Comput. 75, 596–606.

Moritz, B., Zimmermann, T., 2014. Deep Conditional Portfolio Sorts: The Relation Between Past and Future Stock Returns. LMU Munich and Harvard University Working Paper.

Papaioannou, P., Dionysopoulos, T., Russo, L., Giannino, F., Janetzko, D., Siettos, C., 2017. S & P500 forecasting and trading using convolution analysis of major asset classes. Procedia Comput. Sci. 113, 484–489.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.

Qiu, J., Wang, B., Zhou, C., 2020. Forecasting stock prices with long-short term memory neural network based on attention mechanism. PLoS One 15 (1), e0227222.

Sadorsky, 2021. A random forests approach to predicting clean energy stock prices. J. Risk Financ. Manag. 14, 48.

Schmidhuber, J., Hochreiter, S., 1997. Long short-term memory. Neural Comput. 9, 1735–1780.

Sezer, O.B., Ozbayoglu, A.M., 2018. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. Appl. Soft Comput. 70, 525–538.

Sharma, A., Tiwari, P., Gupta, A., Garg, 2021. Use of LSTM and ARIMAX algorithms to analyze impact of sentiment analysis in stock market prediction. Intell. Data Commun. Technol. Internet Things: Proc. ICICI 2020, 377–394.

Siami-Namini, S., Namin, A.S., 2018. Forecasting economics and financial time series: ARIMA vs. LSTM. preprint, arXiv:1803.06386.

Singh, K., Tiwari, R., P, Johri., Elngar, A., 2020. Feature selection and hyper-parameter tuning technique using neural network for stock market prediction. J. Inf. Technol. Manag. 12, 89–108.

Takeuchi, L., Lee, Y.-Y.A., 2013. Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks. Technical Report, Stanford University.

Tran, D.T., Iosifidis, A., Kanniainen, J., Gabbouj, M., 2018. Temporal attention-augmented bilinear network for financial time-series data analysis. IEEE Trans. Neural Netw. Learn. Syst. 30, 1407–1418.

Xue, J., Zhou, S., Liu, Q., Liu, X., Yin, J., 2018. Financial time series prediction using ℓ_2,1rf-ELM. Neurocomputing 277, 176–186.