Journal of Computational Social Science
https://doi.org/10.1007/s42001-019-00057-5

RESEARCH ARTICLE

Comparison of machine learning methods for financial time series forecasting at the examples of over 10 years of daily and hourly data of DAX 30 and S&P 500

Deniz Ersan[1] · **Chifumi Nishioka**[2] · **Ansgar Scherp**[3]

Received: 17 June 2019 / Accepted: 24 October 2019

Abstract
This article conducts a systematic comparison of three methods for predicting the direction (+/−) of financial time series using over ten years of DAX 30 and the S&P 500 datasets at daily and hourly frames. We choose the methods from representative machine learning families, particularly supervised versus unsupervised. The methods are support vector machines (SVM), artificial neural networks, and *k*-nearest neighbor (*k*-NN). We explore the influence of different training window lengths and numbers of out-of-sample predictions. Furthermore, we investigate whether kernel principle component analysis (KPCA) improves prediction, through reducing data dimensionality. Additionally, we verify whether combining machine learning methods by bootstrap aggregating outperforms single methods. Key insights from the experiment are: All machine learning methods are in principle useful to predict the direction of (+/−) financial time series. But to our surprise, increasing the window size only helps to a certain extent for hourly data, before it actually reduces the performance. The number of out-of-sample predictions had a small impact, while KPCA made a strong difference for SVM and *k*-NN. Finally, backtesting selected machines with a trading system on daily data revealed that the lazy learner *k*-NN outperforms the supervised approaches.

Keywords Financial time series forecasting · Prediction · Machine learning · Temporal analysis

✉ Chifumi Nishioka
nishioka.chifumi.2c@kyoto-u.ac.jp

Deniz Ersan
deniz.ersan@ism.de

Ansgar Scherp
as18255@essex.ac.uk

[1] Kiel University, Christian-Albrechts-Platz 4, 24118 Kiel, Germany

[2] Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501, Japan

[3] University of Essex, Wivenhoe Park, Colchester CO4 3SQ, England, UK

Published online: 05 November 2019

Introduction

Financial time series (e.g., stock price, index) forecasting is an important task for the financial industry. Even small improvements in predictive performance can be profitable. Although a lot of works have dealt with forecasting financial time series, it is still regarded as one of the most challenging tasks [33]. According to the prominent Efficient Market Hypothesis (EMH), market prices (in efficient markets) reflect all available information of a stock at any time. Prices only adapt on the arrival of new information. Hence, forecasting future stock price movements based on historical information should be impossible in the long run [26]. Despite the prominence of the EMH, numerous researches question its validity [16, 23, 36, 38].

Also, Timmermann et al. [36] consider a possible existence of a "file drawer" bias in published studies, because publishing empirical results that are barely or just statistically insignificant may be out of interest for researchers. Meanwhile, the continuous advance of machine learning techniques and computational power gave birth to machine-learning approaches applied to financial forecasting problems. Encouraging results have been obtained by artificial neural networks (ANNs) [25], support vector machines (SVMs) [33], and k-nearest neighbor method (k-NN) [34]. Krollner et al. [22] provide a survey of recent literature of machine learning approaches applied to financial time series forecasting.

Existing studies usually do not use data with higher than daily sampling rates, with some notable exemptions such as Qu and Zhang [27] and Sirignano and Cont [31]. Finally, we apply further techniques such as dimensionality reduction or ensemble methods, to improve the predicting power of the learning machines. Dimensionality reduction has been used, e.g., by Cao and Tay [7] but only on SVMs. In summary, the contributions given by this paper are

- We analyze and compare the three machine learning methods in terms of their predictive performance. As representative methods, we select k-NN, ANN, and SVM.
- We investigate what are the optimal training window lengths and numbers of out-of-sample (OOS) forecast per training. This is done on both the DAX 30 and the S&P 500 datasets at daily and hourly time frames.
- We analyze the impact of kernel principle component analysis (KPCA), when conducted on the training data, before training our algorithms. We intend to find out if the KPCA can improve the forecasting performance of our algorithms through reducing the dimensionality of the input data.
- We combine our algorithms with the bootstrap aggregating (also known as Bagging) algorithm. More specifically, we investigate, if this ensemble method can outperform the single machine learning method.
- We close our experiments by providing a backtesting and benchmarking of the experimental results for a selection of trained machines. To this end, we apply a simple trading decision method based on our machines' predictions on daily data and compare their performances with a buy-and-hold strategy in the respective index.

Based on our experimental results, we conclude that in principle all three considered machine learning methods are useful for the forecasting financial time series. For all methods, we usually observe that more than 50% of all predictions made by the machines have the same sign as the actual value. Regarding the optimal window length, we can state that compared to daily data, a larger window helps only to certain extent for hourly data. The rationale is simply that too many smaller changes observed on hourly data have no impact on the final index of a trading day, before the prediction quality drops. Regarding the number of OOS predictions that are conducted before a machine is retrained, we can state that it only has a marginal impact on the results. However, we observe that dimensionality reduction with KPCA has a strong impact on the prediction performance when using the SVM and *k*-NN. Given the observation that the smaller changes on hourly data have no impact on the index at the end of the trading day, we run a backtesting of selected machines in a simple trading system on daily returns. To our surprise, we obtained the best prediction performance by the unsupervised lazy learner *k*-NN compared to the supervised methods.

The remainder of this paper is organized as follows: Subsequently, we provide an overview of related work on financial time series forecasting and machine learning. "Selected methods for financial time series forecasting" section briefly introduces the machine learning algorithms we compare in this work, namely SVMs, ANNs, and *k*-NN. Furthermore, we provide an introduction to KPCA and the Bootstrap Aggregating algorithm. "Experiment design" secton describes our experimental approach and research questions. The results of our comparison of the three machine learning methods are documented in "Results of the experiments" section. "Backtesting of the results for selected machines using a simple trading systems" section presents the results of our backtesting of selected trained machines. In "Discussions and limitations" section, we discuss our empirical results to draw inferences regarding our research questions, before we conclude.

Related work

In this section, we review related works in the field of machine learning for financial time series forecasting. While there have been a lot of machine learning methods developed and experimented for financial time series forecasting, these studies usually differ regarding the applied machine learning methods, the datasets, the forecasting time frame, the input variables, and the evaluation method [22]. There is a broad variety of promising applications of artificial neural networks (ANNs) for forecasting financial time series. Among these applications, the accuracy of ANNs varies significantly due to different input variables, data preparation methods and network architecture [17]. Pacelli et al. [25] utilized a genetic algorithm to find the optimal architecture for a multilayer perceptron ANN (MLP-ANN). They used the MLP-ANN for forecasting daily EUR/USD price movements based on fundamentals and historical price data with promising results. Guresen et al. [14] evaluated the predictive performance of an MLP-ANN, a dynamic artificial neural network and a hybrid neural network for forecasting daily closing levels of NASDAQ. Among the

three compared methods, the MLP-ANN achieved the best results in terms of mean square error (MSE) and mean absolute deviation.

Tay and Cao [33] evaluated the performance of a Support Vector Machine (SVM) for forecasting five real future contracts. The authors used lagged historical price returns as well as a technical indicator as input variables. In their experiment, the SVM outperformed the ANN in terms of the normalized MSE (NMSE). Furthermore, in a later work, Cao and Tay [6] validated the dominance of SVM over MLP-ANN, while additionally comparing it to a radial basis function neural networks, which produced comparable results. Additionally, they showed that an SVM with adaptive parameters can achieve better generalization performance as well as needed fewer support vectors than the standard SVM. The results of Cao and Tay are in line with Kim [21] who compared an SVM, a neural network, and the *k*-NN algorithm for predicting the daily directional change of the Korean stock index.

Teixeira et al. [34] proposed an automatic stock trading system combining technical analysis and nearest neighbor classification based on daily stock prices and volumes. Their model was able to outperform a buy-and-hold strategy for most companies in terms of profitability. Furthermore, Brasileiro et al. [4] also elaborated a trading system where buy and sell signals are produced by *k*-NN. Their system was able to outperform 13 out of 15 stocks compared to a buy-and-hold and another strategy.

Cao et al. [7] apply PCA, KPCA, and Independent Component Analysis (ICA) to an SVM for feature reduction. They examined the sunspot data, the Santa Fe dataset A, and five real future contracts and found out that their SVM can perform better with than without feature reduction, while the best performance was accomplished by KPCA followed by ICA. However, they did not apply the feature reduction method to machine learning approaches other than the SVM. Furthermore, performance was only measured by NMSE.

Furthermore, most studies on financial time series data analyze daily changes of the financial data. Hsu et al. use daily and hourly simulations to predict the return of financial indices [16] and determine which factors (such as market maturity, model simulation methodology, covariate composition, and forecast horizon based on daily vs hourly return) explain the disagreement between current machine learning approaches and the EMH. Qu and Zhang [27] used minute-based data to predict the Chinese CSI 300 index. A notable exception is Sirignano and Cont [31], who applied a 3-layer LSTM on high-frequency trading data from 500 stocks. Specific goal of this research was to demonstrate that the combination of 500 stocks yields to good generalization results. However, the computational costs for training the LSTM with the high-frequency trading data are extremely high, too. Thus, in contrast to the works of Qu and Zhang [27] and Hsu et al. [16], we compare the performance of machine learning methods between daily and hourly data. Furthermore, we are investigating which window length is optimal for learning the prediction model, while Sirignano and Cont [31] focused on identifying the universal features in financial time series prediction.

In summary, to the best of our knowledge, there is no empirical comparison of SVMs, neural networks, and *k*-NN applied to hourly stock data. Consequently, the datasets in previous works are relatively small, with the exception of Qu and Zhang

[27]. Not many previous experiments utilized a walk-forward training routine, where the machine is retrained after a predetermined number of OOS predictions is done.

Selected methods for financial time series forecasting

We briefly introduce the applied machine learning methods and parameters that we selected for our experiments to compare forecasts of financial time series.

Support vector machine A support vector machine (SVM) is a supervised machine learning method originally for pattern recognition and binary classification [9]. It learns nonlinear separation, which maximizes the margin between given two classes from training data, by employing kernels. It can be applied to classification as well as regression. In this paper, we specifically employ a support vector regression machine [10], which outputs a real value $y \in \mathbb{R}$.

Artificial neural network An Artificial Neural Network (ANN) is a machine learning method inspired by the functioning of the human brain. Basically the aim of an ANN is to learn a nonlinear mapping $\mathcal{F}_{NN} : X \mapsto Y$, producing a real-valued, discrete-valued or vector-valued function, where X represents a d-dimensional input matrix and Y denotes the output variable [11, 24]. For more details of ANNs, we refer to references [11, 24]. In our experiments, we use 3-layer MLP-ANN with middle layer having 8 hidden nodes.

k-nearest neighbor Both SVMs and ANNs learn a function that infers a real value $y \in \mathbb{R}$ during training process from training data. Thus, they are called supervised machine learning methods. In contrast, the k-Nearest Neighbor (k-NN) simply stores all training data and does not generalize them. Thus, it requires no training process (unsupervised or so-called lazy learning). For a given data which we would like to make regression, the k-NN searches the k closest training data and outputs a real value $y \in \mathbb{R}$ using the closest data. In this paper, we use the Euclidean distance to compute distance between data to find the k closest training data and calculate mean of them as an output value.

Kernel principle component analysis The aim of principle component analysis (PCA) is to transform the original input data, such that the dimension of the data is reduced to a smaller set that still contains most of the information [19]. The Kernel principle component analysis (KPCA) [30] can perform a nonlinear form of PCA. In contrast to the previous mentioned machine learning methods, KPCA involves unlabeled data. To enable nonlinearity, KPCA employs the kernel trick which is used for the nonlinear SVM as well. Using the kernel trick, KPCA can reduce the dimensionality of the input data nonlinearly.

Bootstrap aggregating Bootstrap aggregating, often referred to as "bagging", belongs to the class of ensemble methods. It is the most prominent independent method of ensemble methods [29]. As the name of the method suggests, bootstrap aggregating consists of bootstrapping and aggregating [5]. Bootstrapping is resampling a training set T containing n paired observations of input and output combinations, into subsamples with t observations, where $t \leq n$. This can be done by randomly drawing n observations with replacement from T and repeating this procedure for a predetermined number of times. After bootstrapping, the idea is to use

Fig. 1 DAX 30 index-level chart

the subsamples for training supervised machine learning methods for multiple times and then use the trained machines to predict the outputs, given new input data. Subsequently, aggregation is performed. Via aggregation, the predictions coming from multiple trained machines will be combined. There are numerous ways to aggregate the predictions. A simple and unbiased way to aggregate is to calculate the mean average over all predictions for a particular point in time. Thus, the predictions of all the trained machines are equally weighted. In the end, the predictions are supposed to be less erroneous, since we average over multiple single trained machines.

Experiment design

We first describe the selected datasets of daily and hourly DAX 30 and S&P 500 indices. Subsequently, we describe the features that are used for the financial time series forecasting. Furthermore, we provide information about the data preprocessing, training method, evaluation metrics, as well as our choice of hyper-parameters for the machine learning methods.

Datasets

The datasets used in our experiment cover the period of over 10 years from 02/01/2004 08:00 GMT–06/03/2015 20:00 GMT at an hourly sampling rate. We use hourly as well as daily sampling rates. It contains open-high-low-close index levels of the German stock index (DAX 30) and the American Standard & Poor's 500 index (S&P 500). The datasets are provided by courtesy of OANDA Corporation. In Figs. 1 and 2, we see the index-level evolution of both data for the whole length of our experiment. We observe that over long periods both indexes show similar index level development. Both indexes reveal up and down trends, while the up trends tend to be more consistent in terms of their length.

The output variable of the machine learning methods is the logarithmic rate of change based on the close level. It is common to transform absolute price changes

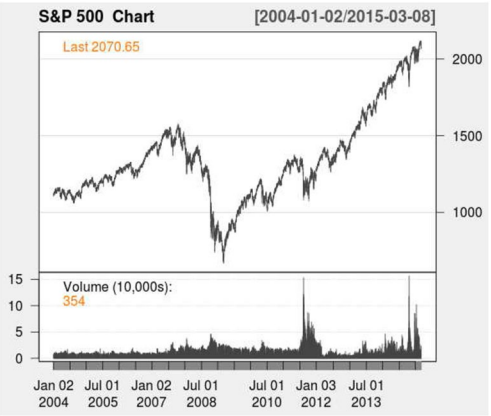


Fig. 2 S&P 500 index-level chart

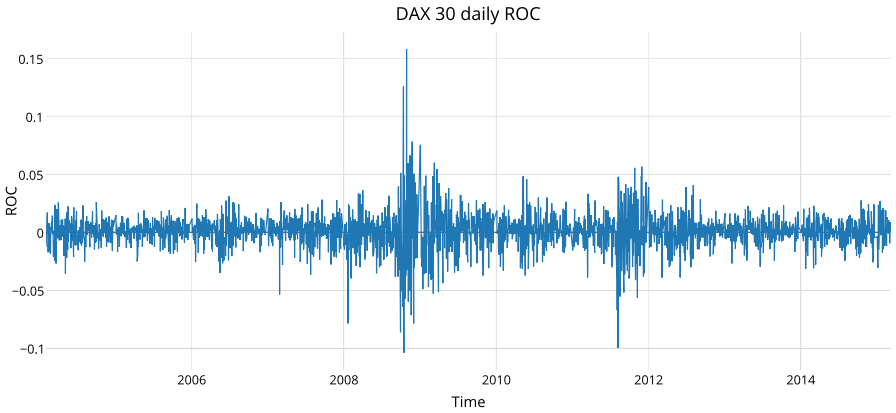


Fig. 3 The rate of changes (ROC) of the DAX 30 daily data, of which we aim at predicting the time series

into relative price changes in order to deal with more symmetric distributed data which will follow a normal distribution more closely [35]. Using logarithmic returns instead of arithmetic returns, we may increase performance in training process, since multiplicative relationships turn into additive, which are also simpler to handle for later calculations [20].

$$r_t = \ln(p_{t+1}) - \ln(p_t). \tag{1}$$

Equation (1) shows the logarithmic price change with p_t denoting the close index level at a point in time t. The index-level DAX 30 data, which is transformed to logarithmic rate of changes by Eq. (1), can be found in Fig. 3.

Table 1 provides the descriptive statistics for the transformed data. Please note that due to hours of low trading activity, hourly data contain hours with zero returns regularly which have been removed from the datasets. The reason is that according to Fig. 3 the ROCs at vast majority of points in time in the data are

Table 1 Descriptive statistics for the return series (a) hourly and (b) daily

	Hourly sampled data		Daily sampled data	
	DAX	S&P	DAX	S&P
Number of observations	37,160	61,085	2836	3384
Mean	2.84e−05	1.01e−05	0.00037	0.00018
Std. dev.	0.00380	0.0026	0.0146	0.0113
Skewness	−0.1243	0.6780	0.0744	−0.2072
Kurtosis	17.04	50.98	11.89	14.46
Minimum	− 0.0551	− 0.0479	− 0.1041	0.1051
Maximum	0.0472	0.0668	0.1578	0.1121
AD test p value	< 0.001	< 0.001	< 0.001	< 0.001
KS test p value	< 0.001	< 0.001	< 0.001	< 0.001

closed to zero. In this paper, we rather want the machine learning methods to learn and train for outputs with a non-zero value.

The difference in the number of observations for the DAX and the S&P datasets is a consequence of different trading hours and days, since both cover the same time span. All time series have means of a value close to zero paired with tails, which are heavier compared to the normal distribution, quantified by the kurtosis measure being greater than 3.00. Our tests for normality, namely the Anderson–Darling (AD) test and the Kolmogorov–Smirnov (KS) test clearly neglect normality with very low p values. These are prominent features of financial time series, which are well documented in the literature [12, 13].

Feature selection

The choice of input variables plays an important role in forecasting and has a large impact on the prediction performance. Since we aim at comparing different machine learning methods rather than finding a combination of input variables that maximizes the prediction performance, we select similar input variables to those of previous works.

The input variables for our models are lagged technical indicators, which can be extracted from our output time series. All indicators are at least lagged for one time step, as we want to avoid look-ahead bias. For our experiment, we can only use data that would have been observable or known during training and testing in order to obtain the true forecasting capability. Table 2 gives an overview of the calculations of the selected input variables and lags, as well as the parameters for some of the input variables. We have four types of input variables. The first one is the lagged logarithmic rates of change (log ROC). It allows the machines to search for patterns in the history of the output variable. In addition, we added lagged exponential moving average rates of change (EMA ROC) to obtain smoothed ROCs. The third type of the input variables is the n-period ROC, which computes the rate of change for a specified length n. For the last one, we computed differences between the current price and the current simple moving average (MA difference), again

Table 2 List of the input variables, their calculations, and parameters

Feature name	Parameter	Selected lags	Calculation
Log ROC		1, 2, 3, 4, 5	$\ln(p_{t+1}) - \ln(p_t)$
EMA ROC	$n = 3$	1, 2, 3, 4, 5	$\frac{\text{EMA}_{t+1}}{\text{EMA}_t}$
n-period ROC		2, 3, 4, 5	$\frac{p_{t+1}}{p_{t-n}} - 1$
MA difference	$n = 20$	1, 2, 3, 4, 5	$p_{t+1} - \text{MA}_{t+1}$

for multiple lags. We expect the MA difference to capture some of potential inter-temporal index-level trends, as it will take larger (or smaller) values when the current index-level departs upwards (or downwards) from the current 20-period moving average index-level. In total, we have 19 input variables for training machine learning methods to predict the log ROC of the next point in time.

Data preprocessing and walk-forward routine

Important for a fair comparison of the machine learning methods is to create a systematic approach for data preprocessing, training, and prediction. We use a walk-forward routine for all the machine learning methods. In the walk-forward routine, we divide the whole data into smaller and equally sized subsets, which are used for training and test. A subsample contains $n + i$ observations, with n denoting the window length for training and i the number of desired out-of-sample (OOS) predictions. Once the respective machine learning method is trained with the window length n, it will produce i OOS predictions. Afterwards, the training window moves i points in time forward. Then, the machine learning methods run training with the new training window and generate new predictions. Basically, the data for training the machine learning methods are simply "walking-forward", while adding new observations and dropping old observations. The walk-forward routine continues to move forward until the entire data are consumed. More specifically, until there are not enough observations to produce the desired number of OOS predictions. Since the machine learning methods are retrained using the "walking-forward" method after each step, we expect them to well adapt to different market regimes.

Furthermore, for comparing different machine learning methods, it is mandatory to ensure that all methods use exactly the same data as well as avoid look-ahead bias. Otherwise, the results would lack validity. For example, the look-ahead bias can be introduced during normalizing or scaling the data (see, e.g., [6, 7, 33]). If one seeks to use scaled or normalized data in a walk-forward environment, the scaling of the data has to strictly occur for each subsample separately. If one scales the whole data prior to subsampling and applies a walk-forward routine after scaling, each subset will contain information which originally have not been available at that point in time. The reason is that scaling and normalizing functions contain elements of order theory, e .g., global minima or global maxima.

Thus, it is important to note that in our experiments, we do not apply scaling. This keeps the level of comparability among the machine learning methods as high as possible. However, it also introduces a restriction regarding the activation

function for the ANN, which is described in "Selected hyper-parameters" section. Moreover, as mentioned earlier, we remove all zero-valued ROC before computing subsamples and inputs. It can hedge the risk of different treating of zero values in the different machine learning methods, which could influence or bias the comparability. However, this will only affect our hourly data, as they contain considerably more zero-valued ROCs.

Performance measures

The forecasting performance of the methods is evaluated by three measures. The first one is the root-mean-square error (RMSE), which measures the magnitude of the deviation between a predicted value and the true observable value. A smaller RMSE indicates a better prediction performance. Formally, RMSE is defines as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{2}$$

with y_t denoting the actual value and $\hat{y}_t$ the predicted value.

The second measure focuses on the directions of predictions. The directional symmetry (DS) measure calculates the proportion of correctly predicted changes of ROC in direction from t_{i-1} to t_i from the total number of predictions as defined in Eq. (3).

$$\text{DS} = \frac{100}{n}\sum_{t=1}^{n} d_t \text{ with } d_t = \begin{cases} 1 \text{ if } (y_t - y_{t-1})(\hat{y}_t - \hat{y}_{t-1}) \geq 0 \\ 0 \text{ otherwise} \end{cases}. \tag{3}$$

The higher the values of DS, the better is the method in predicting directional changes. We multiply 100 in Eq. (3), in order to represent the DS measure in percentage.

The third measure is the sign symmetry, which simply measures the percentage of total predictions that have the same sign as the actual value. It is defined as Eq. (4).

$$\text{SS} = \frac{100}{n}\sum_{t=1}^{n} s_t \text{ with } s_t = \begin{cases} 1 \text{ if } sign(y_t) = sign(\hat{y}_t) \\ 0 \text{ otherwise} \end{cases}. \tag{4}$$

The higher SS measures suggest a better prediction performance. Again, we multiply 100 in Eq. (4) to represent the DS measure in percentage.

Both the DS measure and SS measure are rather important for traders, since one of their goals is to anticipate the correct direction of the indices at next point in time. Please note that even a very high value of the DS measure and SS measure cannot guarantee sustainable profits or alternatively a higher value of the DS measure and SS measure does not necessarily result in a higher trading profit. Nevertheless, a higher DS measure and SS measure should be preferred from an investors' point of view, ceteris paribus.

Selected hyper-parameters

It is important to select the optimal set of parameters for machine learning methods. Improper parameters can lead to overfitting or underfitting [6], which in return negatively influences the generalization performance. The methodologies for finding the optimal parameters have been investigated, but yet not successfully answered. Typical suggested methodologies are grid search [2], random search [2], Bayesian optimization [18, 32], and gradient-based optimization [8]. In addition, optimizing hyper-parameters with large datasets, i.e., financial time series at hourly sampling rates, combined with a walk-forward routine, can easily become unfeasible for standard desktop multi-core central processing units. The same holds true for the required memory, which can easily exceed the standard 8 or 16 GB. As the focus of this work is to compare various machine learning methods, we do not cover the optimization of parameters. Instead, we use similar hyper-parameters to those from experiments in related works (see "Related work" section). The selected parameters with respect to each machine learning method are in terms of SVM, we employ the radial basis function kernel (i.e., Gaussian kernel). We set the cost $C = 100$, $\gamma = 0.1$, $\epsilon = 0.001$, and choose ϵ-regression as regression type. Regarding ANN, standard backpropagation is employed as learning function. We set the learning parameter $\eta = 0.1$ and ANN has eight hidden nodes.At the initial iteration, we randomly set weights. We use a linear function as activation function instead of a more commonly used sigmoid function. The reason is that the sigmoid function strictly results in the range of [0, 1] and our data cannot be fit in the range of [0, 1]. For k-NN, we set the number of neighbors $k = 5$. In KPCA, we employ the Radial basis function kernel (i.e., Gaussian kernel). We set the number of features 100 and $\sigma = 0.025$

Results of the experiments

This section presents the results of our experiments. It is organized in subsections according to the research questions (i) to (iv) described in "Introduction". We begin with the (i) performance comparison of the three machine learning methods, on both daily and hourly data. Subsequently, we determine the (ii) optimal window lengths for training the machines as well as the (iii) optimal number of OOS predictions. Finally, we investigate the (iv) influence of Kernel principle component analysis on the prediction performance. Please note that the research question (v) on the backtesting of selected trained machines is documented in "Backtesting of the results for selected machines using a simple trading systems" section.

Performance comparison of SVM, ANN, and *k*-NN

Daily Data Tables 3 and 4 provide the performance of the three machine learning methods (i .e., SVM, ANN, k-NN) with respect to each of the daily basis DAX 30 and S&P 500 data. The RMSEs of the three machine learning methods range from

Table 3 Results of the three machine learning methods on the daily DAX 30 data for varying windows lengths and different numbers of OOS predictions per training

Window	# OOS	RMSE			DS			SS		
		SVM	ANN	KNN	SVM	ANN	KNN	SVM	ANN	KNN
100	10 (10%)	0.021	0.021	0.016	50.262	59.288	53.109	49.213	51.536	49.700
100	25 (25%)	0.021	0.019	0.016	49.944	58.879	55.402	48.710	50.729	50.953
100	50 (50%)	0.021	0.020	0.016	50.604	61.321	58.906	49.434	49.509	48.642
200	20 (10%)	0.021	0.018	0.016	50.625	59.570	52.188	49.492	48.945	49.922
200	50 (25%)	0.021	0.019	0.017	51.686	61.804	54.000	49.412	51.412	50.784
200	100 (50%)	0.021	0.018	0.016	50.760	59.240	58.680	49.560	49.680	51.400
400	40 (10%)	0.021	0.017	0.017	51.653	61.949	50.847	**51.653**	**52.712**	49.492
400	100 (25%)	0.021	0.018	0.017	51.609	60.478	51.043	51.087	48.783	49.826
400	200 (50%)	0.021	0.017	0.017	**52.727**	63.682	59.364	51.136	50.045	50.773
800	80 (10%)	0.022	0.018	0.018	50.781	62.031	51.771	50.990	50.885	51.250
800	200 (25%)	0.022	0.019	0.019	50.167	**65.278**	52.500	49.111	50.722	49.389
800	400 (50%)	0.024	0.020	0.019	49.563	61.438	**59.875**	48.625	50.063	49.625
1600	160 (10%)	**0.019**	0.017	**0.015**	51.250	64.732	47.500	50.357	50.625	52.143
1600	400 (25%)	**0.019**	**0.016**	0.016	50.250	61.000	52.625	50.125	49.000	**53.750**
1600	800 (50%)	**0.019**	0.019	0.016	49.625	63.875	55.875	50.250	50.875	49.250
	Avg.	**0.021**	**0.018**	**0.017**	**50.767**	**61.638**	**54.246**	**49.944**	**50.368**	**50.460**
	Min	**0.019**	**0.016**	**0.015**	**49.563**	**58.879**	**47.500**	**48.625**	**48.783**	**48.642**
	Max	**0.024**	**0.021**	**0.019**	**52.727**	**65.278**	**59.875**	**51.653**	**52.712**	**53.750**

Bold values indicate the best performance

0.015 to 0.024 in the DAX 30 data and slightly lower from 0.010 to 0.017 in the S&P 500 data. All machine learning methods achieve on average lower RMSEs for the S&P 500 data compared to the DAX 30 data. In terms of the minimum RMSE, the *k*-NN achieved the lowest minimum RMSE in the DAX 30 data, followed by the ANN with a slightly higher minimum RMSE of 0.016, while the SVM ranks last with a higher minimum RMSE of 0.019. The same order pertains for the maximum RMSE. In the S&P 500 data, the ANN outperforms on average the *k*-NN by a 0.001 margin, while the SVM generates the highest RMSE.

The direction performance measures show a different ranking. Referring to the DS measure, the ANN achieved on average the highest DS measure of 61.64 in the DAX 30 data. Whereas the *k*-NN holds the lead on the S&P 500 data, but only by a small margin. In contrast, the SVM shows the worst average performance among the three machine learning methods. However, in terms of predicting directional changes of the daily ROC, all the three machine learning methods are able to produce on average more correct predictions than wrong ones in both data.

In addition, all methods—with one exceptional case—are able to produce more correct predictions than incorrect predictions for the sign of the next day's index level change. The SVM has the highest average SS overall in the S&P 500 data. At the same time, the SVM is the one exception with an SS smaller than 50, namely 49.43 for the DAX 30. The performance of the other two machine learning methods

Table 4 Results of the three machine learning methods on the daily S&P 500 data for varying windows lengths and different numbers of OOS predictions per training

Window	# OOS	RMSE			DS			SS		
		SVM	ANN	KNN	SVM	ANN	KNN	SVM	ANN	KNN
100	10 (10%)	0.016	0.016	0.013	51.672	51.115	53.313	49.721	49.164	49.319
100	25 (25%)	0.016	0.016	0.013	52.093	53.891	55.659	49.860	50.419	50.140
100	50 (50%)	0.016	0.015	0.013	51.969	52.281	58.844	51.125	50.031	50.938
200	20 (10%)	0.016	0.014	0.013	52.276	52.756	52.083	50.833	50.385	50.513
200	50 (25%)	0.016	0.013	0.013	51.677	52.774	54.194	51.290	50.290	51.000
200	100 (50%)	0.016	0.013	0.013	52.548	52.871	**59.355**	50.742	49.161	50.452
400	40 (10%)	0.016	0.013	0.013	51.130	52.123	50.171	49.829	51.438	50.068
400	100 (25%)	0.016	0.013	0.013	51.690	53.207	52.414	50.655	49.690	50.000
400	200 (50%)	0.016	0.014	0.013	50.857	52.929	57.643	50.107	50.250	49.893
800	80 (10%)	0.017	0.014	0.014	50.806	53.347	50.685	49.919	50.524	48.750
800	200 (25%)	0.016	0.014	0.014	50.833	51.042	53.500	49.875	51.292	50.375
800	400 (50%)	0.016	0.014	0.014	50.833	56.250	58.625	50.958	47.083	50.167
1600	160 (10%)	**0.013**	0.011	**0.011**	**53.125**	53.500	51.313	51.438	48.938	**51.188**
1600	400 (25%)	**0.013**	**0.010**	**0.011**	52.188	59.188	50.875	51.563	49.688	49.063
1600	800 (50%)	**0.013**	0.011	**0.011**	51.813	**59.750**	57.188	**52.750**	**53.688**	49.500
	Avg.	**0.016**	**0.013**	**0.013**	**51.701**	**53.802**	**54.391**	**50.711**	**50,136**	**50.091**
	Min	**0.013**	**0.010**	**0.011**	**50.806**	**51.042**	**50.171**	**49.721**	**47.083**	**48.750**
	Max	**0.017**	**0.016**	**0.014**	**53.125**	**59.750**	**59.355**	**52.750**	**53.688**	**51.188**

Bold values indicate the best performance

is very similar with the SS measure of 50.40 for the DAX 30 and the SS measure of 50.10 for the S&P 500. Furthermore, the *k*-NN has the highest maximum SS measure with 53.75 in the DAX 30 data, whereas the ANN obtains the peak maximum SS measure of 53.69 in the S&P 500 data. The *k*-NN produces the lowest maximum SS measure in the S&P 500 data.

Considering the average scores of all three performance measures for the three machine learning methods, the ANN and the *k*-NN outperform the SVM under the vast majority of cases in both data.

Hourly data The results of the machine learning methods for hourly data are depicted in Tables 5 and 6. Again, we observe smaller RMSEs values for the S&P 500 data compared to the DAX 30 data. Overall, the RMSEs are smaller than in the daily dataset, which is due to the shorter time frame. The mean and standard deviation of the hourly data are also lower than those of the daily data according to Table 1.

Furthermore, we observe that the *k*-NN classifier achieves the best RMSE results for both datasets, considering the average, minimum and maximum RMSEs. The second best method is ANN, while the SVM produces the highest RMSEs again. In contrast to the daily data, we have a clear cut order regarding the RMSEs of our three learning machines.

Table 5 Results of the three machine learning methods on the hourly DAX 30 data for varying windows lengths and different numbers of OOS predictions per training

Window	# OOS	RMSE			DS			SS		
		SVM	ANN	KNN	SVM	ANN	KNN	SVM	ANN	KNN
800	80 (10%)	0.0054	0.0044	**0.0042**	50.6167	52.1090	50.3524	50.4075	50.3442	50.0909
800	200 (25%)	0.0054	0.0045	**0.0042**	50.4945	52.2238	52.1492	50.3923	50.1713	50.1188
800	400 (50%)	0.0053	0.0044	**0.0042**	50.5611	51.7861	57.3389	50.6194	49.7694	49.7333
1600	160 (10%)	0.0051	0.0043	**0.0042**	51.0586	52.2889	50.4730	50.8080	50.3266	49.9352
1600	400 (25%)	0.0051	0.0043	**0.0042**	51.0795	53.0597	51.8438	50.8608	50.5710	50.1875
1600	800 (50%)	0.0051	0.0043	**0.0042**	51.1818	52.3551	**57.3949**	50.7301	50.3352	50.1648
3200	320 (10%)	0.0050	0.0043	**0.0042**	51.4239	51.8190	50.2889	**50.9876**	50.4127	50.0943
3200	800 (25%)	0.0049	0.0043	**0.0042**	51.5387	51.8839	51.2560	50.7649	50.1161	50.1964
3200	1600 (50%)	**0.0048**	**0.0042**	**0.0042**	51.2589	53.8780	57.0268	50.5893	50.5744	50.1042
6400	640 (10%)	0.0049	0.0045	0.0044	51.4193	51.8620	49.9772	50.3418	50.3320	50.0260
6400	1600 (25%)	0.0049	0.0044	0.0043	51.7072	54.1875	51.8553	50.4178	**50.7961**	49.9605
6400	3200 (50%)	0.0050	0.0047	0.0044	51.8819	51.9514	56.5868	50.4757	48.9757	50.3542
12800	1280 (10%)	0.0050	0.0048	0.0047	52.6974	50.0699	50.0164	50.8840	49.7738	50.0247
12800	3200 (25%)	0.0051	0.0049	0.0047	52.5357	54.8304	51.7545	50.9598	50.6116	**50.6250**
12800	6400 (50%)	0.0053	0.0050	0.0050	**52.9219**	**54.9323**	56.3177	50.7552	49.4375	49.5885
	Avg.	**0.0051**	**0,0045**	**0.0044**	**51.4918**	**52.6158**	**52.9754**	**50.6663**	**50.1698**	**50.0803**
	Min	**0.0048**	**0.0042**	**0.0042**	**50.4945**	**50.0699**	**49.9772**	**50.3418**	**48.9757**	**49.5885**
	Max	**0.0054**	**0.0050**	**0.0050**	**52.9219**	**54.9323**	**57.3949**	**50.9876**	**50.7961**	**50.6250**

Bold values indicate the best performance

Table 6 Results of the three machine learning methods on the hourly S&P 500 data for varying windows lengths and different numbers of OOS predictions per training

Window	# OOS	RMSE			DS			SS		
		SVM	ANN	KNN	SVM	ANN	KNN	SVM	ANN	KNN
800	80 (10%)	0.0036	0.0031	**0.0028**	51.0276	50.8184	50.3254	50.3005	50.1544	50.2822
800	200 (25%)	0.0036	0.0034	**0.0028**	50.9518	50.4684	52.1927	50.4585	49.8322	50.0797
800	400 (50%)	0.0036	0.0031	**0.0028**	51.0067	50.7533	**57.6533**	50.3483	49.7817	50.2717
1600	16 (10%)0	0.0035	0.0031	0.0029	51.1540	51.0478	50.1567	50.4751	50.0623	50.2375
1600	400 (25%)	0.0035	**0.0029**	0.0029	51.1740	51.0743	51.3970	50.5997	50.0811	49.5777
1600	800 (50%)	0.0034	0.0030	**0.0028**	50.9375	50.1520	57.3699	50.3615	49.8176	50.3209
3200	320 (10%)	0.0034	**0.0029**	0.0029	51.2830	50.1979	50.1389	**50.9479**	50.4896	50.0556
3200	800 (25%)	**0.0033**	**0.0029**	0.0029	51.4531	51.1250	51.8542	50.8785	49.7569	50.0503
3200	1600 (50%)	**0.0033**	0.0030	0.0029	51.4080	49.2865	57.0677	50.7622	50.2257	50.2257
6400	640 (10%)	**0.0033**	**0.0029**	0.0029	51.8070	52.8051	50.0919	50.5846	50.0129	**50.3548**
6400	1600 (25%)	**0.0033**	0.0030	0.0029	51.6011	50.2096	51.2831	50.5790	**50.6305**	49.7794
6400	3200 (50%)	**0.0033**	0.0030	0.0029	51.4265	50.6250	56.6434	50.7574	50.0533	49.7279
12800	1280 (10%)	0.0034	0.0030	0.0031	52.7724	52.2530	49.8628	50.5617	49.7635	50.1098
12800	3200 (25%)	**0.0033**	0.0033	0.0031	52.6875	52.5563	51.8375	50.5104	49.8375	50.2917
12800	6400 (50%)	0.0034	0.0033	0.0031	**53.2679**	**53.0536**	57.1183	50.6473	50.2254	49.9353
	Avg.	**0.0034**	**0.0031**	**0.0029**	**51.5972**	**51.0951**	**52.9995**	**50.5848**	**50.0483**	**50.0867**
	Min.	**0.0033**	**0.0029**	**0.0028**	**50.9375**	**49.2865**	**49.8628**	**50.3005**	**49.7569**	**49.5777**
	Max.	**0.0036**	**0.0034**	**0.0031**	**53.2679**	**53.0536**	**57.6533**	**50.9479**	**50,6305**	**50.3548**

Bold values indicate the best performance

The DS of our algorithms on the hourly data are lower on average than for the daily dataset, except for the SVM, which has slightly higher average DS on hourly data. Even the minimums of the SVM are greater than 50. The *k*-NN produces the best average DS values with 52.9995 (S&P 500) and 53.9754 (DAX 30) while the highest average SS values are obtained by the SVM with 50.6663 (DAX 30) and 50.5848 (S&P 500). Furthermore, all algorithms can correctly predict the sign of the next hour's logarithmic return more often than every second time, meaning that their average SS exceeds 50.

Determining the optimal training window lengths

We increased the window length by factor two, for four times, starting at 100 days for the daily data and 800 hours for the hourly data. The longest window lengths for training are 1600 days for the daily data and 12,800 hours for the hourly data. Tables 3, 4, 5 and 6 show the performance for the different window lengths.

Daily Data All machine learning methods reach their lowest RMSEs with the longest window length consisting of 1600 latest observations in the training data. Therefore, the RMSE improves when a window length is increased. However, the *k*-NN is able to obtain low levels of RMSEs already from the shortest window length of 100 days. In the daily S&P 500 data, the RMSE of the *k*-NN with the window length of 100 days is equal to the minimum RMSE of the SVM. In the daily DAX 30 days data, the RMSE of the *k*-NN with the window length of 100 days is even smaller than the minimum RMSE of the SVM.

However, in terms of the DS measure and SS measure, they fluctuate unsteadily over different window lengths. Thus, they do not show a systematic pattern for the different window lengths.

Hourly Data We observe that increasing the window length improves the RMSE only until window lengths reach 3200 or 6400 hours. Advancing to 12,800 often causes the RMSEs of our learning machines to steepen up to their average RMSEs or higher. This holds particularly true for the *k*-NN algorithm, which validates its capability to produce low levels of RMSEs from short training window lengths as against the SVM and ANN, as well on hourly data. In terms of the DS measure, the SVM output results in higher values when the window length is increased. In both data, the SVM reaches the maximum DS measures when using window length of 12,800 hours. In addition, ANN outputs the maximum DS measures when using the longest window length in both data, as well. When using *k*-NN, the DS measures vary unsteadily over different window lengths. Referring to the SS measure, we observe almost no correlation between the SS measure and the window length.

Compared to the daily data, a larger window helps only to certain extent for hourly data. This is possibly because too many smaller changes observed on hourly data have no impact on final index of a trading day.

Determining the optimal number of out-of-sample predictions

To gain an intuition about how many out-of-sample (OOS) predictions to produce before retraining of the model should be triggered, we choose three different window lengths for testing. These window lengths are always 10%, 25% and 50% of the training window lengths as shown in Tables 3, 4, 5, 6.

Considering all experiments on both the daily data and hourly data, we observe that variations in the number of OOS predictions only have a marginal impact on the RMSE of the three machine learning methods. This can be concluded by keeping the training window length constant and analyzing the variations in RMSEs for the 10%, 25% and 50% OOS proportions of training window length. Furthermore, our experiments cannot highlight a certain proportion, which systematically outperforms other proportions across all data and machine learning methods.

Influence of the kernel principle component analysis on the prediction performance

Finally, we examine whether the Kernel Principle Component Analysis (KPCA) can further enhance the prediction performance of the three machine learning methods. In Table 7, we compare the prediction performance of the machine learning methods shown in "Performance comparison of SVM, ANN, and *k*-NN" section to the machine learning methods when KPCA is applied to the input data prior to each training process. For the sake of brevity, we only show results for selected window lengths in Table 7. The selected window lengths are 1600 days for the daily data and 3200 h for the hourly data.

The RMSEs of the SVM and *k*-NN decrease or remain close to the RMSEs without KPCA in the vast majority of cases. This holds particularly true for the SVM, as its RMSEs decrease in all cases through KPCA. For *k*-NN, we observe that the RMSEs remain constant, slightly decreasing, or slightly increasing occasionally. On the other hand, KPCA has no effect or even negative effect on RMSEs when using the ANN. The RMSEs mostly rise or stay close to the RMSEs without KPCA.

In terms of the DS measure, the results demonstrate that KPCA enhances the prediction performance a lot for the SVM and *k*-NN. Both the SVM and *k*-NN result in DS values higher than 80 or even 90, when trained on the extracted features. Divisively, their SS measures usually decreases, when the DS measures increase by great margins.

Different from the SVM and *k*-NN, we observe a different result for the ANN with KPCA in terms of the DS measure and SS measure. In most cases, the DS measures decrease in utilization of KPCA, while the SS measures fluctuate bidirectionally revealing no systematic pattern.

For RMSE, one can assume that even without KPCA the machine learning algorithms already recognize most of the patterns in the data and the dimension reduction has no significant influence on the reduction of the estimation error. This is one of the reasons why KPCA does not improve RMSE.

In summary, we observe dimensionality reduction done by KPCA has a strong impact on the prediction performance measured by DS when using the SVM and *k*-NN.

Table 7 Results of the three machine learning methods with KPCA compared to those without KPCA

	Window	# OOS	RMSE			DS			SS		
			SVM	ANN	KNN	SVM	ANN	KNN	SVM	ANN	KNN
	DAX 30 daily										
KPCA	1600	160	0.0190	0.0169	0.0152	51.25	64.73	47.50	50.36	50.63	52.14
	1600	160	0.0190	0.0141	0.0148	56.43	54.11	49.82	48.66	49.20	51.52
KPCA	1600	400	0.0191	0.0161	0.0162	50.25	61.00	52.63	50.13	49.00	53.75
	1600	400	0.0171	0.0171	0.0165	52.38	51.50	53.38	47.50	50.38	48.63
KPCA	1600	800	0.0194	0.0188	0.0162	49.63	63.88	55.88	50.25	50.88	49.25
	1600	800	0.0152	0.0268	0.0157	53.13	55.25	59.75	48.63	54.63	54.13
	S&P 500 daily										
KPCA	1600	160	0.0132	0.0107	0.0106	53.13	53.50	51.31	51.44	48.94	51.19
	1600	160	0.0095	0.0114	0.0098	99.56	51.69	73.56	48.38	51.00	49.56
KPCA	1600	400	0.0132	0.0103	0.0109	52.19	59.19	50.88	51.56	49.69	49.06
	1600	400	0.0095	0.0119	0.0099	99.88	52.50	81.69	49.00	52.00	49.75
KPCA	1600	800	0.0128	0.0113	0.0107	51.81	59.75	57.19	52.75	53.69	49.50
	1600	800	0.0095	0.0112	0.0100	99.94	47.31	85.31	45.00	46.13	49.31
	DAX 30 hourly										
KPCA	3200	320	0.0050	0.0043	0.0042	51.42	51.82	50.29	50.99	50.41	50.09
	3200	320	0.0039	0.0042	0.0043	99.83	50.46	53.73	50.93	49.64	49.80
KPCA	3200	800	0.0049	0.0043	0.0042	51.54	51.88	51.26	50.76	50.12	50.20
	3200	800	0.0039	0.0043	0.0043	99.93	50.27	56.39	50.87	50.30	50.27
KPCA	3200	1600	0.0048	0.0042	0.0042	51.26	53.88	57.03	50.59	50.57	50.10
	3200	1600	0.0039	0.0042	0.0043	99.96	50.71	62.77	51.09	50.30	50.08
	S&P 500 hourly										
KPCA	3200	320	0.0034	0.0029	0.0029	51.28	50.20	50.14	50.95	50.49	50.06
	3200	320	0.0027	0.0030	0.0027	99.84	49.11	99.79	50.36	50.13	50.18

Table 7 (continued)

	Window	# OOS	RMSE			DS			SS		
			SVM	ANN	KNN	SVM	ANN	KNN	SVM	ANN	KNN
KPCA	3200	800	0.0033	0.0029	0.0029	51.45	51.13	51.85	50.88	49.76	50.05
	3200	800	0.0027	0.0029	0.0028	99.93	49.22	99.91	50.24	50.49	50.38
KPCA	3200	1600	0.0033	0.0030	0.0029	51.41	49.29	57.07	50.76	50.23	50.23
	3200	1600	0.0027	0.0030	0.0028	99.97	50.40	99.93	50.04	50.47	50.32

Influence of bootstrap aggregating on the prediction performance

Finally, we employ an ensemble method, namely bootstrap aggregating. As for the KPCA, we use the same training window lengths that yielded better prediction performance results relative to the other training window lengths. Table 8 summarizes the prediction performance of the machine learning methods when using the bootstrap aggregating. Through the utilization of the bootstrap aggregating, the RMSEs of the three machine learning methods are only slightly reduced for both daily data and hourly data. Specifically, the bootstrap aggregating algorithm reduces the RMSEs of the three machine learning methods on average by 0.0023 in the daily DAX 30 and S&P 500 data.

In the hourly data, we achieve a difference of 0.00038 on average. The reductions of the RMSEs come along with fluctuations in the DS measure and SS measure upwards as well as downwards varying in magnitude. The magnitudes of the fluctuations in the DS measure and SS measure are the lowest for the SVM in both daily and hourly data with average differences ranging from 0.02 to 0.64. This represents a slight worsening through bootstrap aggregating.

Our ensembles of ANN and *k*-NN produce changes of greater magnitude in the DS measure in both directions. For instance, the DS measures of the ANN decrease a lot in both daily and hourly DAX 30 data, while the DS measures rather increase in the S&P 500 data. The DS measure of the *k*-NN with bagging is significantly reduced in all data, except for one out of twelve cases. In contrast, the SS measures increase slightly in seven cases. Overall, a clear statement on the influence of the bootstrap aggregating algorithm cannot be made.

Backtesting of the results for selected machines using a simple trading systems

To provide an impression on the profitability of the machine learning methods, we embed predictions generated by the machine learning methods into a simple trading system on daily data. In the trading system, we thereafter simulate using the data used in the experiment. Based on the one-step ahead ROC prediction, our system decides to be either fully invested in a long trade or in a short trade for the next point in time. It keeps the position until a change in the sign of the next period prediction is forecasted. In this case, it instantly liquidates the position and opens another according to the new predicted direction. We choose to predict the return on daily data, as from the investigation of the optimal training window length in "Determining the optimal training window lengths" section, we found out that compared to daily data, a larger window helps only to certain extent for hourly data. With other words, our rationale to backtest using daily data is simply that the smaller hourly changes have no impact on the final index of a trading day.

Of course, this is neither a valid backtesting approach nor proper trading system. Both contain multiple shortcomings. However, it is beyond scope of this work to delve deep into algorithmic trading. The results are presented solely for the purpose of providing an impression of the profitability. We compute multiple performance

Table 8 Results of the three machine learning methods with bootstrap aggregating ("bagging") compared to those without bootstrap aggregating

	Window	# OOS	RMSE			DS			SS		
			SVM	ANN	KNN	SVM	ANN	KNN	SVM	ANN	KNN
	DAX 30 daily										
Bagging	1600	160	0.0190	0.0169	0.0152	51.25	64.73	47.50	50.36	50.63	52.14
	1600	160	0.0143	0.0134	0.0133	52.77	55.89	50.27	51.88	50.63	47.77
Bagging	1600	400	0.0191	0.0161	0.0162	50.25	61.00	52.63	50.13	49.00	53.75
	1600	400	0.0159	0.0149	0.0145	49.88	59.13	45.88	49.63	48.13	48.13
Bagging	1600	800	0.0194	0.0188	0.0162	49.63	63.88	55.88	50.25	50.88	49.25
	1600	800	0.0156	0.0146	0.0145	50.50	57.25	50.38	49.25	48.63	51.25
	S&P 500 daily										
Bagging	1600	160	0.0132	0.0107	0.0106	53.13	53.50	51.31	51.44	48.94	51.19
	1600	160	0.0104	0.0097	0.0095	51.63	57.00	51.00	51.06	48.00	52.69
Bagging	1600	400	0.0132	0.0103	0.0109	52.19	59.19	50.88	51.56	49.69	49.06
	1600	400	0.0105	0.0096	0.0096	51.63	58.31	49.69	50.13	49.88	49.25
Bagging	1600	800	0.0128	0.0113	0.0107	51.81	59.75	57.19	52.75	53.69	49.50
	1600	800	0.0104	0.0097	0.0095	51.31	61.81	50.19	50.69	47.50	49.06
	DAX 30 hourly										
Bagging	3200	320	0.0050	0.0043	0.0042	51.42	51.82	50.29	50.99	50.41	50.09
	3200	320	0.0041	0.0039	0.0039	51.59	50.69	50.05	50.86	50.78	50.06
Bagging	3200	800	0.0049	0.0043	0.0042	51.54	51.88	51.26	50.76	50.12	50.20
	3200	800	0.0041	0.0040	0.0039	51.76	47.88	49.93	50.93	50.18	50.77
Bagging	3200	1600	0.0048	0.0042	0.0042	51.26	53.88	57.03	50.59	50.57	50.10
	3200	1600	0.0041	0.0040	0.0039	51.08	50.78	49.84	50.88	49.79	50.73
	S&P 500 hourly										
Bagging	3200	320	0.0034	0.0029	0.0029	51.28	50.20	50.14	50.95	50.49	50.06
	3200	320	0.0028	0.0027	0.0027	51.21	52.31	49.93	50.78	50.38	50.19

Table 8 (continued)

	Window	# OOS	RMSE			DS			SS		
			SVM	ANN	KNN	SVM	ANN	KNN	SVM	ANN	KNN
Bagging	3200	800	0.0033	0.0029	0.0029	51.45	51.13	51.85	50.88	49.76	50.05
	3200	800	0.0028	0.0027	0.0027	51.35	53.21	49.90	50.68	50.74	50.22
Bagging	3200	1600	0.0033	0.0030	0.0029	51.41	49.29	57.07	50.76	50.23	50.23
	3200	1600	0.0028	0.0027	0.0027	51.26	52.37	49.86	50.54	50.38	49.95

Table 9 Backtesting results for selected learning machines (top) compared to a buy-and-hold strategy in the respective index (bottom)

	(1)	(2)	(3)	(4)	(5)	(6)
Dataset	Daily Dax 30	Daily S&P 500	Daily Dax 30	Daily Dax 30	Daily Dax 30	Daily Dax 30
Window/OOS	1600/400	1600/800	1600/800	1600/800	1600/160	1600/160
Algorithm	*k*NN	ANN	*k*-NN	*k*-NN	SVM	SVM
KPCA/bagging	None	None	KPCA	None	Bagging	None
Cumulative return	1.310	0.125	0.931	0.003	− 0.231	− 0.269
Annualized return	0.302	0.019	0.230	0.001	− 0.057	− 0.068
Annualized Sharpe ratio	1.313	0.125	1.001	0.004	− 0.273	− 0.324
Annualized SD	0.538	0.514	0.542	0.493	0.511	0.504
Max drawdown	− 0.206	− 0.285	− 0.210	− 0.319	− 0.510	− 0.400
Max length drawdown	166	896	279	501	990	990
Cumulative return	0.271	1.065	0.271	0.271	0.532	0.532
Annualized return	0.078	0.121	0.078	0.078	0.101	0.101
Annualized Sharpe ratio	0.340	0.802	0.340	0.340	0.480	0.480
Annualized SD	0.546	0.550	0.546	0.546	0.546	0.546
Max drawdown	− 0.334	− 0.215	− 0.334	− 0.334	− 0.334	− 0.334
Max length drawdown	509	265	509	509	509	509

measures along with [1] for the selected machine learning methods. The results of the performance are documented in Table 9.

We selected two machine learning methods, one for the DAX 30 and one for the S&P 500, which performed well in the experiment in terms of performance measures. Especially a high level of the SS measure is important for the trading systems. Along with this principle, we choose one method from the experiment in terms of KPCA and one utilized bootstrap aggregating. In addition, we pick the same two machine learning methods but without the KPCA and bootstrap aggregating, respectively, for the sake of comparison.

Among the selected trading strategies, the best prediction performance was achieved by the *k*-NN as shown in column (1) in Table 9 with the highest annualized risk

adjusted returns, measured by the Sharpe ratio. It clearly outperforms the DAX buy-and-hold strategy in every measure. Comparison of the methods shown at columns (3) and (4) reveals that the application of KPCA made the poor algorithm (4) more profitable algorithm (i.e., the algorithm (3)). It enabled the algorithm to clearly outperform the other over DAX 30. However, the bootstrap aggregating SVM shown in column (5) even worsened the performance of the standard SVM (6). We enclose the performance charts of the methods shown in the column (1) to (6) and the two additional SVMs for the hourly DAX 30 and S&P 500 data in the Appendix.

Discussion and limitations

In our performance comparison for the daily data, the SVM brought out the poorest predictions among the three learning algorithms in terms of average RMSEs. Both the *k*-NN and the ANN result on average in lower RMSEs. Regarding the SS, we found the SVM to produce stable results considering the levels of SS. Partially, our results contradict with previous related experiments, which merely found the SVM to be superior over *k*-NNs and ANNs on daily data by the RMSE, or similar squared error measures [6, 33]. This may be due to the usage of different input variables and hyper-parameters and methodological deviations from our approaches. We believe that the latter plays a greater role, as related experiments often made use of scaling and smoothed output variables. This can easily affect comparability of the machines, if not conducted with high diligence. Scaling has to take place before training to avoid look-ahead bias. If the entire data are scaled or normalized in advance and then subsampled, every decision made by a learning system would be based on data comprising sure information about the future. The reason is that scaling and normalizing functions contain information about the global minima or global maxima of the dataset. This information is only available when it has the access to the full dataset, i.e., in our case one would even have access to financial data of a future point in time. That would, non-controversially, break the rules for financial-time series forecasting. Related papers usually do not provide detailed information on their scaling approaches or scale the entire dataset prior to subsampling, which may be due to unawareness of this pitfall [6, 7, 33]. However, it must be explained in detail at least for the sake of reproducibility.

Through the analysis of optimal window lengths, we found out that the performance of our learning machines depends to some extent on the training window length. For daily data, the longest training window length under study of 1600 yielded the best results. In the hourly datasets, the best performances were achieved with 3200–6400 training observations. Overall, for the purpose of financial time series forecasting with machine learning algorithms, we recommend to use at least 1000 training observations, as shorter training windows tend to produce inferior performances. We find this particularly relevant with regard to SVMs and ANNs. The *k*-NN method approaches its maximum potential faster, meaning that it needs less observations to come close to its best performances.

The high values of DS achieved through the application of KPCA shed light on a weakness of this performance measure in its calculation. We recall Eq. (3):

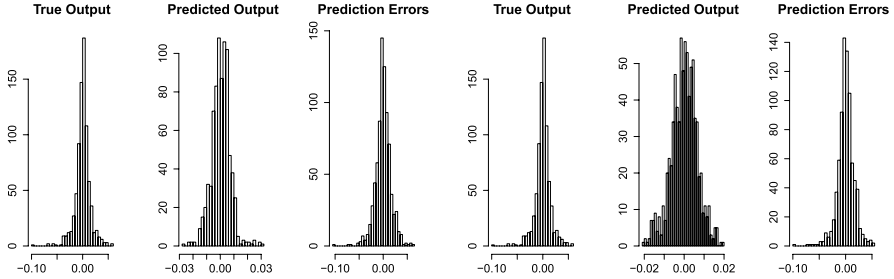
$$\mathrm{DS} = \frac{100}{n} \sum_{t=1}^{n} d_t \text{ with } d_t = \begin{cases} 1 \text{ if } (y_t - y_{t-1})(\hat{y}_t - \hat{y}_{t-1}) \geq 0 \\ 0 \text{ otherwise} \end{cases}. \tag{5}$$

The weakness origins from the term d_t, which will take the value of one if $(y_t - y_{t-1})(\hat{y}_t - \hat{y}_{t-1})$ is greater or equal than zero. This means that the change in direction was correctly predicted. Please note that the term $(y_t - y_{t-1})(\hat{y}_t - \hat{y}_{t-1})$ will be zero if either $(y_t - y_{t-1})$ or $(\hat{y}_t - \hat{y}_{t-1})$ equals zero. Hence, if the actual value y or the predicted value $\hat{y}$ does not change at all in value from $t-1$ to t, then d_t will judge this as correctly predicted directional change, which is wrong in any different case than for: $(y_t - y_{t-1}) = 0$ and $(\hat{y}_t - \hat{y}_{t-1}) = 0$. Thus, the DS will increase in some situations, when it actually should decrease.

Cao and Tay compared different algorithms for financial time series forecasting utilizing the DS and three other measures [6, 33]. Cao et al. [7] analyzed the impact of KPCA for SVMs on financial time series data, among other datasets. However, they evaluated the impact only using NMSE. They summarized that KPCA can improve generalization performance, which we cannot confirm without limitations. Formally, a lower MSE through KPCA endorses this work. However, it turns out that in some cases a lower MSE is accompanied by an undesirable effect, which is the predicting machine producing only two values. In Fig. 4, we show that a lower RMSE through KPCA may not generally be preferable for financial forecasting. In Fig. 4a, b, we observe that the histograms of the predicted outputs remain quite symmetric, containing predictions with positive and negative values. However, in the negative example in Fig. 4c, d we notice that after the application of KPCA the predicted outputs only contain two classes of outputs, while both classes being negative in value.

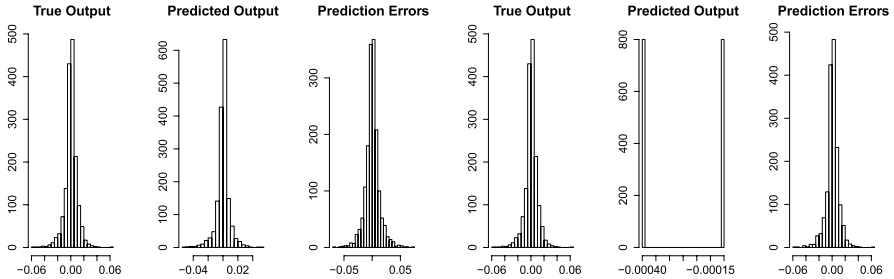
Clearly, this is not a desirable feature for a predicting machine in financial time series forecasting such a particular machine becomes useless as it only contains negative predictions. One could argue that it produced lower RMSEs, but in financial time series forecasting we also care about the correct direction of a forecast. In the negative example, we see that the SVM with KPCA only models the left half of the true output histogram. If one only seeks to minimize the RMSE of a prediction, one can simply use the training sample mean of the output variable as a constant predictor, which is very likely to produce a low RMSE. Additionally, this predictor is certain to obtain a DS of 100. That leads to conclude that—in the domain of financial time series forecasting—the application of KPCA should not only be evaluated on RMSE. In this example, the wrong interpretation of RMSE could have been prevented if the SS measure had been taken into consideration. In the positive example, the SS values increased through the KPCA and vice versa in the negative example.[1]

[1] In the positive example, we used the *k*-NN 1600/800 DAX 30 daily from our experiments. In the negative example, we used the SVM 1600/800 S&P 500 daily.



(a) Positive Example: k-NN before KPCA ($RMSE = 0.0162$)

(b) Positive Example: k-NN with KPCA ($RMSE = 0.0157$)



(c) Negative Example: SVM before KPCA ($RMSE = 0.0128$)

(d) Negative Example: SVM with KPCA ($RMSE = 0.0095$)

Fig. 4 Histograms before and after utilization of KPCA for the true output, the predicted output and the prediction error

Our bootstrap aggregation algorithms could produce lower RMSEs than the single learning machines. In contrast to the KPCA, it did not introduce the aforementioned undesirable features. However, the bootstrap aggregating resulted in both improvements and worsening of the SS values. But the fluctuations are usually small. Considering the reductions in RMSE and the small changes in SS, we found the bootstrap aggregation helpful for forecasting financial time series forecasting. Nevertheless, one has to take into account that bagging introduces new free parameters to the learning problem, which have to be determined by the supervisor in the optimization process. For the objective of improving a learning machine's generalization performance, we prefer bootstrap aggregating over KPCA.

We believe our trading systems to rather represent a lower bound on the general profitability potential. First, we did not optimize our algorithms nor did we try different input variables. Second, our trading strategy is quite naive and omits risk and money management. Further, in a real-world application, one must not require the machine to be invested with every prediction it produces. However, we have not taken bid/ask spreads and transactions costs into account which both would diminish the trading performance in a real-world application to some extent. Nevertheless, we strongly believe that our backtesting of the results for selected machines is of high value since it provides insights into the actual impact of the trained machines on the financial time series forecasting task. However, future work with more advanced trading systems will be clearly required.

Since the results of ANN are between SVM or *k*-NN or slightly above, we have yet not investigated further neural networks. As our experiments show, an optimal window size for hourly data is already reached with 1600 regarding RMSE (depending on the number of OOS). Using other types of neural networks like LSTMs (long short-term memory) [15] does require a lot of training data [31], for which the optimal window sizes we determined may be too small.

Even if trained on the entire history of the financial time series, one may assume that this had not a positive effect on the performance as the network may simply learn to ignore "older" information in the data stream as it does not help further optimizing the prediction output. But, as the ANN outputs the maximum DS measures when using the longest window length, it will be interesting to investigate methods such as LSTMs with attention. Existing research on using LSTMs for predicting the E-mini S&P [37] does not seem to outperform the methods considered in this paper. However, a direct comparison is difficult as the datasets, their time spans, measures applied etc. differ.

Finally, we acknowledge that in recent years a growing number of text-mining approaches have been applied to publicly available text-data like news articles, and tweets on Twitter or social media in general to correlate it with financial data. For instance, Bollen et al. [3] investigated whether the sentiment of large-scale Twitter feeds is correlated to the value of the Dow Jones Industrial Average (DJIA) over time. The authors explored that daily variations in sentiment in the Twitter data have a statistically significant correlation to daily Dow Jones Industrial Average close price movements. However, correlations may improve predictions, but they do not represent a causation between the two. Rao et al. [28] combined Twitter data and search volume index data collected from Google trends. Their results are in line with Bollen et al. [3].

However, the combination of news or sentiment paired with technical indicators as input variables in training predictive machine learning models is beyond the scope of this work. The reason is that so far there are no sufficient text data publicly available that covers the 10 years of our daily and hourly DAX 30 and S&P 500 data from 2004 to 2015 (see description of the datasets in "Datasets" section). To the best of our knowledge, the so far largest publicly available text-data is the New York Times Annotated Corpus (NYTAC). The NYTAC dataset contains 1.8 million new articles from the New York Times published between January 1987 and June 2007. Thus, the overlap with our datasets is rather small.

Conclusion

We empirically compared three machine learning methods *k*-NN, SVM, and ANN for their usefulness on forecasting non-linear financial time series. Furthermore, optimal training window lengths and the impact of KPCA and the bootstrap aggregating algorithm were investigated. We measured the performance of the machine learning methods by RMSE, DS, and SS. Our research revealed that the DS measure has a general weakness, which can significantly impact and mislead interpretations of the experiment results. Regarding the RMSE, the *k*-NN and the ANN outperformed the SVM in most cases, whereas the SVM produced the most stable performance in terms of SS. For forecasting daily directions, a training window length of 1600 days yielded in lower RMSE

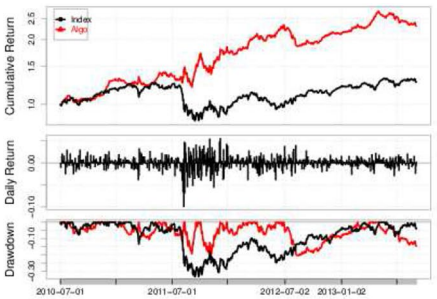
values than for shorter training windows. For hourly predictions, training window lengths of 3200 and 6400 hours were preferable to shorter and longer training window lengths. Overall, we consider all the three machine learning methods useful for the forecasting financial time series, since the levels of SS were usually higher than 50. Additionally, we investigated the influence of KPCA, when applied to the learning algorithms in training. Our results show that it often reduced RMSEs, while the effect on the SS was ambiguous. Additionally, the decrease in RMSE was often accompanied by the introduction of an undesirable side effect, exposed by DS approaching 100. In summary, KPCA can be helpful for financial forecasting, when handled with caution. It is insufficient to evaluate the influence of KPCA only by RMSEs for forecasting financial time.

Compliance with ethical standards

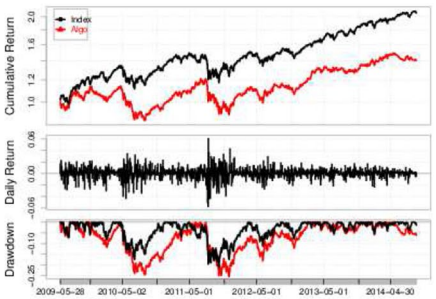
Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Appendix

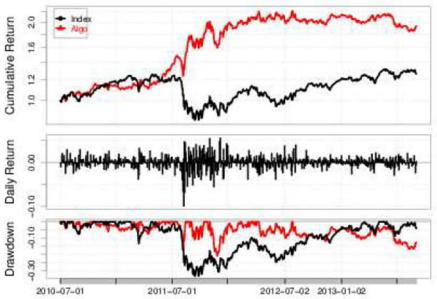
See Fig. 5.



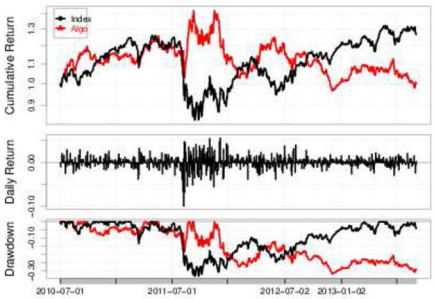
(a) Standard k-NN, 1600 window length, 400 OOS, DAX 30 daily data



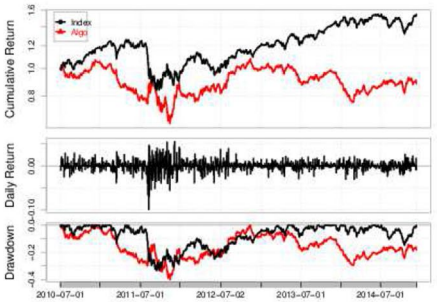
(b) Standard ANN, 1600 window length, 800 OOS, S&P 500 daily data



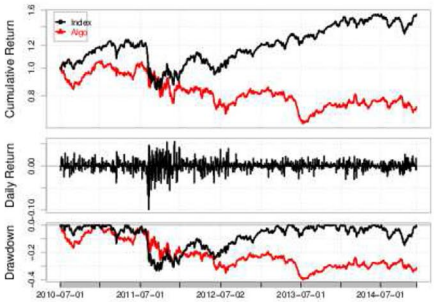
(c) KPCA k-NN, 1600 window length, 800 OOS, DAX 30 daily data



(d) Standard k-NN, 1600 window length, 800 OOS, DAX 30 daily data



(e) Bagging SVM, 1600 window length, 160 OOS, DAX 30 daily data



(f) Standard SVM, 1600 window length, 160 OOS, DAX 30 daily data

Fig. 5 Performance benchmark charts for trading strategy

References

1. Bacon, C. R. (2005). *Practical Portfolio performance measurement and attribution. The wiley finance series*. Oxford: Wiley.
2. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(1), 281–305.
3. Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, *2*(1), 1–8.

4. Brasileiro, R. C., Souza, V. L. F., Fernandes, B. J. T., & Oliveira, A. L. I. (2013). Automatic method for stock trading combining technical analysis and the artificial bee colony algorithm. In *2013 IEEE Congress on Evolutionary Computation (CEC)* (pp 1810–1817).
5. Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.
6. Cao, L. J., & Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, *14*(6), 1506–1518.
7. Cao, L. J., Chua, K. S., Chong, W. K., Lee, H. P., & Gu, Q. M. (2003). A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine. *Neurocomputing*, *55*(1), 321–336.
8. Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, *46*(1–3), 131–159.
9. Cortes, C., & Vapnik, V. (1995). Support vector machine. *Machine Learning*, *20*(3), 273–297.
10. Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. *Advances in Neural Information Processing Systems*, *9*, 155–161.
11. Engelbrecht, A. P. (2007). *Computational intelligence: An introduction* (2nd ed.). Oxford: Wiley.
12. Granger, C. W. J., & Ding, Z. (1995). Some properties of absolute return: An alternative measure of risk. *Annales d'Economie et de Statistique*, *40*, 67–91.
13. Granger, C. W. J., Spear, S., & Ding, Z. (2000). *Stylized facts on the temporal and distributional properties of absolute returns: An update* (pp. 97–120). Singapore: World Scientific. chap 6.
14. Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, *38*(8), 10389–10397.
15. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.
16. Hsu, M. W., Lessmann, S., Sung, M. C., Ma, T., & Johnson, J. E. (2016). Bridging the divide in financial market forecasting: machine learners vs. financial economists. *Expert Systems with Applications*, *61*, 215–234.
17. Huang, W., Lai, K. K., Nakamori, Y., & Wang, S. (2004). Forecasting foreign exchange rates with artificial neural networks: a review. *International Journal of Information Technology & Decision Making*, *3*(1), 145–165.
18. Hutter, F., Hoos, H.H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization (LION)* (pp. 507–523). Springer.
19. Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer Series in Statistics. Springer.
20. Kaastra, I., & Boyd, M. (1996). Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, *10*(3), 215–236.
21. Kim, K. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1–2), 307–319.
22. Krollner, B., Vanstone, B.J., & Finnie, G.R. (2010). Financial time series forecasting with machine learning techniques: a survey. In *Proceedings of the 18th European Symposium on Artificial Neural Networks (ESANN)*.
23. Malkiel, B. G. (2003). The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, *17*(1), 59–82.
24. Mitchell, T. M. (1997). *Machine Learning* (1st ed.). New York: McGraw-Hill.
25. Pacelli, V., Bevilacqua, V., Azzollini, M., et al. (2011). An artificial neural network model to forecast exchange rates. *Journal of Intelligent Learning Systems and Applications*, *3*(2), 57–69.
26. Peters, E. E. (1996). *Chaos and order in the capital markets: A new view of cycles, prices, and market volatility*. Oxford: Wiley.
27. Qu, H., & Zhang, Y. (2016). A new kernel of support vector regression for forecasting high-frequency stock returns. In *Mathematical Problems in Engineering 2016*.
28. Rao, T., & Srivastava, S. (2013). Modeling movements in oil, gold, forex and market indices using search volume index and twitter sentiments. In *Proceedings of the 5th Annual ACM Web Science Conference (WebSci)* (pp. 336–345). ACM.
29. Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, *33*(1–2), 1–39.
30. Schölkopf, B., Smola, A., & Müller, K.R. (1997). Kernel principal component analysis. In *International Conference on Artificial Neural Networks (ICANN)* (pp. 583–588). Springer.
31. Sirignano, J., & Cont, R. (2018). Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, *19*(9), 1449–1459.

32. Snoek, J., Larochelle, H., & Adams, R.P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951–2959). Curran Associates, Inc.
33. Tay, F. E. H., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega*, *29*(4), 309–317.
34. Teixeira, L. A., & De Oliveira, A. L. I. (2010). A method for automatic stock trading combining technical analysis and nearest neighbor classification. *Expert Systems with Applications*, *37*(10), 6885–6890.
35. Thomason, M. (1999). The practitioner methdods and tool. *Journal of Computational Intelligence in Finance*, *7*(3), 35–45.
36. Timmermann, A., & Granger, C. W. J. (2004). Efficient market hypothesis and forecasting. *International Journal of Forecasting*, *20*(1), 15–27.
37. Wang, J., Sun, T., Liu, B., Cao, Y., & Zhu, H. (2019). CLVSA: A convolutional LSTM based variational sequence-to-sequence model with attention for predicting trends of financial markets. In S. Kraus (Ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10–16, 2019, ijcai.org* (pp. 3705–3711). https://doi.org/10.24963/ijcai.2019/514.
38. Yen, G., & Lee, C. (2008). Efficient market hypothesis (EMH): Past, present and future. *Review of Pacific Basin Financial Markets and Policies*, *11*(2), 305–329.