Data Science in Finance and Economics

http://www.aimspress.com/journal/dsfe

DSFE, 2(4): 437–463.
DOI:10.3934/DSFE.2022022
Received: 25 October 2022
Revised: 01 December 2022
Accepted: 03 December 2022
Published: 12 December 2022

Research article

High-Frequency Trading with Machine Learning Algorithms and Limit Order Book Data

Manveer Kaur Mangat[1], Erhard Reschenhofer[1], Thomas Stark[1] and Christian Zwatz[2]

[1] Department of Statistics and Operations Research, University of Vienna, Oskar-Morgensternplatz 1, 1090 Vienna, Austria

[2] Department of Economics, University of Klagenfurt, Universitätsstrasse 65-67, 9020 Klagenfurt am Wörthersee, Austria

* **Correspondence:** Email: manveer.mangat@univie.ac.at; Tel: +431-4277-38622.

Abstract: In this paper, we examine the usefulness of machine learning methods such as support vector machines, random forests and bagging for the extraction of information from the limit order book that can be used for intraday trading. For our empirical analysis, we first get 50 raw features from the LOBSTER message file and order book file of the iShares Core S&P 500 ETF for the time period from 27.06.2007 to 30.04.2019 and then construct 18 higher-level features (aggregated to 5 minutes frequency) which serve as predictors. Using straightforward specifications for the machine learning procedures and thereby avoiding excessive data snooping, we find that these procedures are unable to find high dimensional patterns in the order book that could be used for trading purposes. The observed significant predictability is mainly due to the inclusion of only one variable, namely the last price change, and is probably too small to ensure profitability once transaction costs are taken into account.

Keywords: directional forecasting; trading strategies; support vector machines; random forests; bagging

JEL Codes: C53, C55, C58

1. Introduction

A proven method for the directional forecasting of financial time series would clearly be of great practical value. Unfortunately, empirical evidence of predictability is often impaired by the nonstationarity of these time series. Their statistical characteristics as well as their relationships with economic and political variables change over time. For example, the Monday-effect (see, e.g., Krämer 1998), the turn-of-the-month effect (see, e.g., Lakonishok and Smidt, 1988; Reschenhofer, 2010), and the positive first-order autocorrelation vanished over time (Reschenhofer et al., 2020b). In this regard,

high-frequency data have a certain advantage over daily or monthly data because the prediction of the former is usually not based on stable patterns or relationships. Machine learning methods are used instead. Changes over time can easily be taken into account by rebuilding the model periodically, e.g., every year. The sorry practice of dividing the whole observation period in a learning period and a testing period and thereby building the model only once is completely incomprehensible. A dubious advantage of this practice is that it makes it easier to obscure data snooping. While an automatic and transparent approach is required if the model is rebuilt repeatedly, the one-time model building allows the unnoticed fiddling about with the model design and the tuning parameters. The fact that empirical studies of high-frequency data typically use extremely short observation periods is clearly not helpful in this regard. For example, Cont (2011) used thirty minutes, Zheng et al. (2012) and Kercheval and Zhang (2015) four hours and one day, respectively, Ntakaris et al. (2018) and Nousi et al. (2019) ten days, Gao et al. (2021) one month, and Kearns and Nevmyvaka (2013) two years. Kearns and Nevmyvaka (2013) examined the application of machine learning to the problem of predicting near-term price movements (as measured by the bid-ask midpoint) from limit order book data. They found profitable predictive signals but warned that the midpoint is a fictitious, idealized price and profitability is more elusive once trading costs (spread-crossing) are taken into account. Learning was performed for each of nineteen equities using all 2008 data and testing was performed using all 2009 data. Using prices and volumes at different levels of the limit order book and employing support vector machines, Kercheval and Zhang (2015) also found indications of short-term predictability of price movements. Their dataset covers a complete trading day for five stocks listed at the Nasdaq. The features extracted from this dataset included the prices and volumes for ten levels of the ask and bid side of the order book as well as bid-ask spreads. They used support vector machines with various feature sets and evaluated their models with cross validation. However, they did not test their forecasting procedure in a realistic trading experiment but used only a period of four hours to carry out some sort of plausibility check. Nousi et al. (2019) and Han et al. (2015) also found indications for the predictability of mid-price movements using similar sets of features but different observation periods (ten days and thirty minutes, respectively) and other machine learning algorithms (neural networks and random forests, respectively) in addition to support vector machines. Finally, indications of directional predictability were also found in the case of high-frequency exchange rates by Frömmel and Lampaert (2016) and, more directly, by Cai and Zhang (2016), who observed that trading strategies based on their directional forecasts were no longer profitable once transaction costs were accounted for. In this paper we propose a general methodological framework for the construction of machine learning based trading strategies for high frequency limit order data, which also includes a detailed account of the data cleaning procedure. We subsequently put it to effect using the support vector machine algorithm, which we apply to a fairly large limit order book dataset covering slightly more than eleven years. Thereby we consciously avoid tailoring the design of the forecasting procedures for the dataset at hand. Instead, we rather give an overview of the possible results that can be obtained with straightforward specifications. The support vector machine algorithm and the data set are described in Sections 2 and 3, respectively. Section 4 gives a thorough step by step description of the methodological framework. Section 5 presents the empirical results and briefly discusses the results obtained with other machine learning algorithms. Section 6 concludes.

2. Machine learning algorithm

Before providing a description of the binary support vector machine learning algorithm, we first recall basic terminology used in the machine learning context. The aim of binary classifiers is to categorize an observation $x_t = (x_{t1}, \ldots, x_{tp}) \in \mathbb{R}^p$, $p \geq 1$, into one of two classes after being *trained* using a *training set*. The *training set* consists of observations that have already been classified.

Indeed, suppose we have a set of n observations $x_1, \ldots, x_n \in \mathbb{R}^p$ falling into one of two classes $y_t \in \{-1, 1\}$, where $t = 1, \ldots, n$, then the training set T is defined by

$$T = \{(x_t, y_t) | 1 \leq t \leq n\}, \tag{1}$$

where y_t is the *label* of x_t, $\mathbb{R}^p$ is the *feature space*, the components x_{ti}, $i \in \{1, \ldots, p\}$, of $x_t \in \mathbb{R}^p$ are the *features* and x_t is the *feature vector*. Utilizing this training set T, we aim to construct prediction models, that will enable us to - as accurately as possible - predict the class of the next k observations contained in a specified *evaluation set* E

$$E = \{(x_t, y_t) | n + 1 \leq t \leq n + k\}. \tag{2}$$

In order to obtain these prediction models, we use the support vector machine learning algorithm which we briefly describe in the following subsection.

2.1. Support vector machine

When a linear separation between the class of observations x_t with $y_t = -1$ and the class of observations x_t with $y_t = 1$ is possible, we want to compute the optimal separating (affine) hyperplane

$$g(x) = x^T\beta + \beta_0 = 0, \quad \beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R} \tag{3}$$

that maximizes the margin M between both classes, see Figure 1.

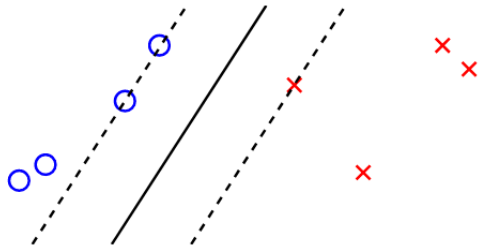

Figure 1. Support vector classification for the separable case. The hyperplane is the solid line and the distance between the two dashed lines is the margin M. Points on the dashed line are the support vectors.

In particular we want to find a hyperplane $g(x) = x^T\beta + \beta_0 = 0$, such that

$$g(x) = x^T\beta + \beta_0 \geq 1 \text{ for } x_i \text{ with } y_i = 1, \tag{4}$$

$$g(x) = x^T\beta + \beta_0 \leq -1 \text{ for } x_i \text{ with } y_i = -1. \tag{5}$$

Then the corresponding *decision* function is given by

$$f(x) = sign(x^T\beta + \beta_0). \tag{6}$$

In Figure 1, the blue points x_b on the dashed line satisfy $g(x) = x^T\beta + \beta_0 = -1$ and the red points x_r on the dashed line satisfy $g(x) = x^T\beta + \beta_0 = 1$. So the margin, i.e. the distance between the two dashed lines, is given by

$$M = \frac{|\beta x_r + \beta_0|}{||\beta||} + \frac{|\beta x_b + \beta_0|}{||\beta||} = \frac{2}{||\beta||}, \tag{7}$$

where $||\beta|| = \sqrt{\sum_{i=1}^{p} \beta_i^2}$ is the Euclidean norm.Hence finding the optimal, i.e. largest, margin is equivalent to solving the following optimization problem

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 \tag{8}$$

$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1, \quad i = 1, \ldots, n. \tag{9}$$

Since we have a constrained minimization problem, we use the Lagrange multiplier method to solve it. In particular we minimize the Lagrangian (primal) function

$$L_P = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{n} \alpha_i(y_i(x_i^T\beta + \beta_0) - 1) \tag{10}$$

with respect to β and β_0. Setting the respective derivatives to zero results in the following criteria

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad \beta = \sum_{i=1}^{n} \alpha_i y_i x_i. \tag{11}$$

Substituting the criteria (11) into L_P, we obtain the dual optimization problem

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{12}$$

$$\text{subject to } \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1, \ldots, n. \tag{13}$$

Using $\beta = \sum_{i=1}^{n} \alpha y_i x_i$, the decision function is therefore given by

$$f(x) = sign(\sum_{i=1}^{n} \alpha_i y_i x^T x_i + \beta_0). \tag{14}$$

When the two classes are not linearly separable, training observations $x_i \in \mathbb{R}^p$ will be missclassified, i.e $sign(y_i) \neq sign(x_i^T\beta + \beta_0)$. Therefore in this case, we define slack variables $\xi_1, \ldots, \xi_n \in \mathbb{R}^+$ and allow some data points to be on the wrong side of the margin, i.e. $\xi_i \leq 1$, or on the wrong side of the hyperplane, i.e. $\xi_i > 1$, by using the constraints

$$g(x) = x^T\beta + \beta_0 \geq 1 - \xi_i \text{ for } x_i \text{ with } y_i = 1, \tag{15}$$

$$g(x) = x^T\beta + \beta_0 \leq -1 + \xi_i \text{ for } x_i \text{ with } y_i = -1, \tag{16}$$

(see Figure 2).

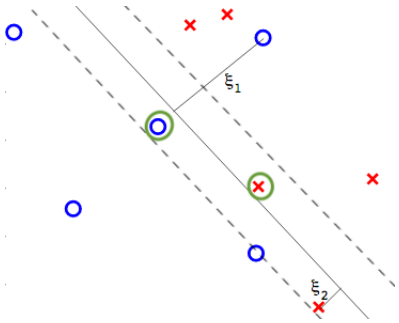

Figure 2. Support vector classification for the non-separable case. Points corresponding to ξ_1 and ξ_2 are on the wrong side of the hyperplane. Points circled in green are on the wrong side of the margin.

Hence the optimal hyperplane can be obtained by solving the following optimization problem

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{n}\xi_i \tag{17}$$

$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i, \tag{18}$$

$$\xi_i \geq 0, \quad i = 1, \ldots, n, \tag{19}$$

or equivalently by solving its corresponding dual problem obtained by applying the Lagrange multiplier method used above

$$\max \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j \tag{20}$$

$$\text{subject to } \sum_{i=1}^{n}\alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n,$$

where $C > 0$ is a tuning parameter specifying the *cost* of missclassification on the learning set. If C is small, the SVM will have a high tolerance for missclassified observations, resulting in wider margins (high bias, low variance). However when C is large, the SVM is heavily penalized for misclassified observations, hence resulting in narrower margins to avoid as many missclassifications as possible (low bias, high variance). Therefore as C increases the risk of overfitting increases.

The decision function in this case is also given by

$$f(x) = sign(\sum_{i=1}^{n}\alpha_i y_i x^T x_i + \beta_0). \tag{21}$$

A more sophisticated approach to attaining a separation of the two classes is to map the feature space $\mathbb{R}^p$ into a higher-dimensional (or even infinite-dimensional) space, i.e. $\phi(x) \in \mathbb{R}^q$, $x \in \mathbb{R}^p$, where $p << q$ and then to apply the aforementioned optimal margin hyperplane algorithm (20) in the higher dimension. The Lagrange dual function is then given by

$$L_D = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle\phi(x_i), \phi(x_j)\rangle, \tag{22}$$

where $\langle \phi(x_i), \phi(x_j) \rangle = \phi(x_i)^T \phi(x_j)$ and the corresponding decision function by

$$f(x) = sign(\sum_{i=1}^{n} \alpha_i y_i \langle \phi(x), \phi(x_i) \rangle + \beta_0). \tag{23}$$

Note that the optimization problem (22) and its solution (23) only contain the observations x_i via dot products. In particular, it is not necessary to specify ϕ and therefore the dimension q as long as a suitable kernel function K

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle, \quad i, j = 1, \ldots, n, \tag{24}$$

is used. In particular, admissible kernel functions $K(x_i, x_j)$ should be symmetric and positive (semi)definite.

Commonly used kernels are the *linear* kernel

$$K(x_j, x_{j'}) = \sum_{i=1}^{p} x_{ji} x_{j'i}, \tag{25}$$

the *polynomial* kernel

$$K(x_j, x_{j'}) = \sum_{i=1}^{p} (1 + x_{ji} x_{j'i})^d, \tag{26}$$

where $d > 0$ is the degree of the polynomial, and the *radial* kernel

$$K(x_j, x_{j'}) = exp(-\gamma \sum_{i=1}^{p} (x_{ji} - x_{j'i})^2), \tag{27}$$

where $\gamma > 0$. Just like for the cost parameter C, the SVM tries to increasingly avoid any missclassifications as the value of γ increases. Hence a too large value of γ runs the risk of overfitting the data.

3. High frequency limit order book data

In this section we describe the high frequency limit order book data sets used to construct features for the machine learning algorithms presented above. Using the online limit order book data tool LOBSTER, we download the data corresponding to the stock of interest after defining the relevant time period. Thereby LOBSTER provides two files: the *message book* and the *order book* file. The message book contains information about each trading event including its time of occurrence (in seconds after midnight with decimal precision of at least milliseconds), its type (submission of a new limit order, cancelation (partial deletion of a limit order), deletion (total deletion of a limit order), execution of a visible limit order, execution of a hidden limit order), its unique order ID, its size (number of shares), its price (dollar price times 10000) and finally whether a buy (1) or sell (-1) limit order is initiated (see Table 1). The corresponding order book is a list which contains unexecuted limit orders for both ask and bid for a specified level l. Thereby each level provides four values (ask price, ask volume, bid price, bid volume) with level 1 corresponding to the limit order with the best ask price, ask volume, bid price, bid volume and level 2 corresponding to the limit order with the second best ask price, ask volume, bid price, bid volume etc. (see Table 2). As to be expected, the evolution of the message book file results in

a change in the order book file. Indeed, the k-th row in the message book facilitates the change in the order book from line $k-1$ to line k. For instance, as a result of the limit order deletion (event type 3) in the second line of the message book (Table 1), 100 shares from the ask side at price 118600 are omitted. This removal corresponds to the change in the order book (Table 2) from line 1 to 2. Accordingly, the best ask volume drops 9484 to 9384 shares.

Table 1. Sample message book file from LOB (2021): the last column indicates whether a sell (-1) or buy (+1) order is initiated and the event types: submission of a new limit order, cancellation (partial deletion of a limit order), deletion (total deletion of a limit order), execution of a visible limit order, execution of a hidden limit order, cross trade, trading halt, correspond to 1,2,...,7, respectively.

Time (sec)	Event Type	Order Id	Size	Price	Direction
⋮	⋮	⋮	⋮	⋮	⋮
34713.685155243	1	206833312	100	118600	-1
34714.133632201	3	206833312	100	118600	1
⋮	⋮	⋮	⋮	⋮	⋮

Table 2. Sample order book file file from LOB (2021) containing the first 2 levels.

Ask Price 1	Ask Size 1	Bid Price 1	Bid Size 1	Ask Price 2	Ask Size 2	Bid Price 2	Bid Size 2	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1186600	9484	118500	8800	118700	22700	118400	14930	⋮
1186600	9384	118500	8800	118700	22700	118400	14930	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

4. Methodology

The flowchart in Figure 3 gives an overview of our proposed methodology. Utilizing high-frequency limit order book data, the aim is to construct new trading strategies based on the support vector machine algorithm. To evaluate the performance of a trading strategy, it is imperative to use a suitable benchmark for comparison. Indeed, in case the regarded stock exhibits a long-term upward trend, the benchmark strategy is always long (buy and hold strategy) and when the regarded stock exhibits a downward trend, the benchmark is constantly short in the stock. Thereby our objective is to predict one step ahead: using the information at time t, we want to forecast whether to buy (mid-quote return positive or constant) or sell (mid-quote return negative) our stock at time $t+1$.

The first step entails downloading the data from LOBSTER:

4.1. Data

For each of the n trading days of the regarded stock, we download a message file and an orderbook file (up to level 5) from LOBSTER. For each 1-second interval, we extract 20 values from the orderbook file (the best ask price, the best ask volume, the best bid price, the best bid volume, ..., the 5-th best ask

price, the 5-th best ask volume, the 5-th best bid price, the 5-th best bid volume) and 30 values from the message file (see variables 21–50 in Table 3), resulting in 50 raw features (Table 3). To obtain the values for the 1-second intervals, we extract the value of the regarded features at the beginning of each second contained within the trading hours (9:00 am – 4:30 pm), which amounts to $m = 6.5$ hours $\times$ 60 minutes $\times$ 60 seconds = 23,400 seconds, hence 23,400 values for each feature per trading day.

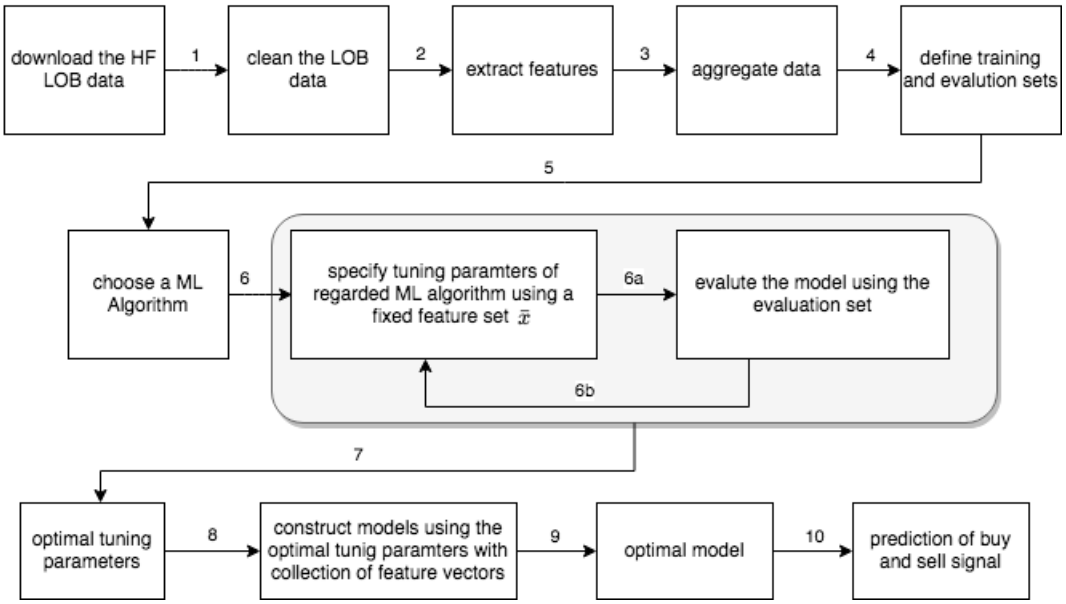

Figure 3. Architectural design of the proposed methodology for obtaining the optimal trading strategy, where HF stands for High Frequency and LOB stands for Limit Order Book

Table 3. 1-second features extracted from the limit order book data downloaded from LOBSTER (for $1 \leq j \leq 5$: $P^{ask}_{j,t}$ level j ask price at time t; $V^{ask}_{j,t}$ level j ask volume at time t; $P^{bid}_{j,t}$ level j bid price at time t; $V^{bid}_{j,t}$ level j bid volume at time t; BO buy order; SO sell order; n number; ap average price; min minimum; max maximum; pd partial deletion order; td total deletion order; ev executed visible order; eh executed hidden order).

1) $P^{ask}_{1,t}$	2) $V^{ask}_{1,t}$	3) $P^{bid}_{1,t}$	4) $V^{bid}_{1,t}$	5) $P^{ask}_{2,t}$
6) $V^{ask}_{2,t}$	7) $P^{bid}_{2,t}$	8) $V^{bid}_{2,t}$	9) $P^{ask}_{3,t}$	10) $V^{ask}_{3,t}$
11) $P^{bid}_{3,t}$	12) $V^{bid}_{3,t}$	13) $P^{ask}_{4,t}$	14) $V^{ask}_{4,t}$	15) $P^{bid}_{4,t}$
16) $V^{bid}_{4,t}$	17) $P^{ask}_{5,t}$	18) $V^{ask}_{5,t}$	19) $P^{bid}_{5,t}$	20) $V^{bid}_{5,t}$
21) BOn	22) BOap	23) BOmax	24) SOn	25) SOap
26) SOmin	27) pdBOn	28) pdBOap	29) pdBOmax	30) pdSOn
31) pdSOap	32) pdSOmin	33) tdBOn	34) tdBOap	35) tdBOmax
36) tdSOn	37) tdSOap	38) tdSOmin	39) evBOn	40) evBOap
41) evBOmax	42) evSOn	43) evSOap	44) evSOmin	45) ehBOn
46) ehBOap	47) ehBOmax	48) ehSOn	49) ehSOap	50) ehSOmin

The next step entails cleaning the data:

4.2. Data cleaning

Before the data can be used, it is subject to thorough processing. Clearly, every algorithm will perform poorly if it is fed with raw unfiltered data. It is crucial to analyze the data at hand before conducting any kind of data cleaning procedure, as applying a fixed set of filters could potentially lead to the loss of vital information.

With that in mind, we present two different data cleaning procedures, where the first cleansing process is specifically designed for the price and volume variables (variables 1–20 from Table 3), while the second cleaning procedure is defined for the remaining 30 variables from Table 3.

The data cleansing process for the level price and volume variables can be divided into four steps. First, the daily percentage of missing values for each of the variables are computed. If the magnitude of the missing values is extensive, the respective day is discarded.

The next step entails capturing extreme values of the price and volume variables and explicitly identifying these as non-available data points. Which measures can be used to detect extreme values? Given the nature of our regarded variables (volume and price levels), an obvious choice is the following collection of relative spreads:

$$\begin{aligned} P^{spr}_{1,t} &= (P^{ask}_{1,t} - P^{bid}_{1,t})/P^{ask}_{1,t} \\ P^{spr}_{j,t} &= (P^{ask}_{j,t} - P^{ask}_{1,t})/P^{ask}_{j,t}, \quad 2 \le j \le 5 \\ P^{spr}_{j+4,t} &= (P^{bid}_{1,t} - P^{bid}_{j,t})/P^{bid}_{1,t} \quad 2 \le j \le 5. \end{aligned} \tag{28}$$

Indeed, using 0.015 as a benchmark, we identify the entry of a variable at time t as non-available, when the corresponding spread exhibits a value greater than 0.015, i.e.

$$\begin{aligned} P^{spr}_{1,t} > 0.015 &\implies P^{ask}_{j,t}, P^{bid}_{j,t} \leftarrow \text{NA, for } j = 1, \dots, 5 \\ P^{spr}_{j,t} > 0.015 &\implies P^{ask}_{j,t} \dots, P^{ask}_{5,t}, V^{ask}_{j,t}, \dots, V^{ask}_{5,t} \leftarrow \text{NA}, 2 \le j \le 5, \\ P^{spr}_{j,t} > 0.015 &\implies P^{bid}_{j-4,t} \dots, P^{bid}_{5,t}, V^{bid}_{j-4,t} \dots, V^{bid}_{5,t} \leftarrow \text{NA}, 6 \le j \le 9, \end{aligned} \tag{29}$$

where *NA* indicates that the value at time t is classified as non-available.

Furthermore we only retain the values of $V^{ask}_{j,t}$ ($V^{bid}_{j,t}$) whose corresponding level price $P^{ask}_{j,t}$ ($P^{bid}_{j,t}$) does not deviate more than 4 cents from the best ask price $P^{ask}_{1,t}$ (bid price $P^{bid}_{1,t}$), i.e.

$$\begin{aligned} P^{ask}_{j,t} - P^{ask}_{1,t} > 0.04 &\implies V^{ask}_{j,t} \leftarrow \text{NA, for } j = 1, \dots, 5 \\ P^{bid}_{1,t} - P^{bid}_{j,t} > 0.04 &\implies V^{bid}_{j,t} \leftarrow \text{NA, for } j = 1, \dots, 5 \end{aligned} \tag{30}$$

In the last step, we additionally compute the log returns of $P^{ask}_{1,t}$, $P^{bid}_{1,t}$ and the mid-quote prices $(P^{bid}_{1,t}+P^{ask}_{1,t})/2$ to identify whole days opposed to single values of the variables at a time t, that need to be removed due to extreme values. Indeed, as soon as a return greater than 0.015 is detected, the whole respective day is completely removed.

This concludes the data cleaning process for the level ask and volume prices. The cleansing procedure for the last 30 variables from Table 3 consists of only two steps. Once again we compute the daily percentage of missing values for each of the variables and discard the respective day in case of a large number of missing values. In contrast to the data cleaning procedure applied to the volume and price variables, we conduct a visual inspection of the last 30 variables from Table 3, by plotting the variables against time to identify any potential outliers. Once these are identified, their corresponding values are set to missing concluding the cleansing procedure.

Having cleaned the data set, the next step is to select and define relevant features:

4.3. Feature extraction

In addition to the cleaned features from the previous section, we define additional features, using the cleaned price and level features, that can be categorized into three groups: 1) spreads 2) volumes and 3) returns (see Table 4).

Table 4. Collection of Features X_i $i = 1, \ldots, 18$.

Feature Category	Feature
Spreads	$X_{1,t} = (P^{ask}_{1,t} - P^{bid}_{1,t})/P^{ask}_{1,t}$
	$X_{j,t} = (P^{ask}_{j,t} - P^{ask}_{1,t})/P^{ask}_{j,t}, \quad 2 \leq j \leq 5$
	$X_{j+4,t} = (P^{bid}_{1,t} - P^{bid}_{j,t})/P^{bid}_{1,t} \quad 2 \leq j \leq 5$
Volumes	$X_{10,t} = \log(V^{ask}_{1,t})$
	$X_{11,t} = \log(V^{bid}_{1,t})$
	$X_{12,t} = \log\Big(\sum_{j=1}^{5} V^{ask}_{j,t} \mathbb{1}_{\{P^{ask}_{j,t} - P^{ask}_{1,t} < 0.04\}}\Big)$
	$X_{13,t} = \log\Big(\sum_{j=1}^{5} V^{bid}_{j,t} \mathbb{1}_{\{P^{bid}_{1,t} - P^{bid}_{j,t} < 0.04\}}\Big)$
	$X_{14,t} = X_{12,t} + X_{13,t}$
	$X_{15,t} = X_{12,t}/X_{13,t}$
Returns	$X_{16,t} = \log(P^{ask}_{1,t}) - \log(P^{ask}_{1,t-1})$
	$X_{17,t} = \log(P^{bid}_{1,t}) - \log(P^{bid}_{1,t-1})$
	$X_{18,t} = \log((P^{ask}_{1,t} + P^{bid}_{1,t})/2) - \log((P^{ask}_{1,t-1} + P^{bid}_{1,t-1})/2)$

For uniformity reasons, we henceforth denote the cleaned features from Table 3 by $X^{basic}_{i,t}$ $i = 1, \ldots, 50$ (e.g. $X_{1,t}$ corresponds to the variable 1 in Table 3, $X_{2,t}$ to the variable 2 etc.) and define $\mathbf{X}$ as the set of all features, i.e.

$$\mathbf{X} = \left\{X^{basic}_{1,t}, \ldots, X^{basic}_{50,t}, X_{1,t}, \ldots, X_{18,t}\right\} \tag{31}$$

containing all the $p = 68$ considered features.

4.4. Data aggregation

Due to the sampling frequency of one second, the features in Table 3 and 4 are inevitably polluted by microstructure noise which is caused by the bid-ask spread, the discreteness of price changes, asynchronous trading, etc.. A common way to reduce the impact of this problem is to perform data aggregation. In particular, we conduct a 5-minute data aggregation procedure reducing the number of data points from 23400 to 78 per day for each variable, adjusting the time stamps accordingly (see Figure 4).

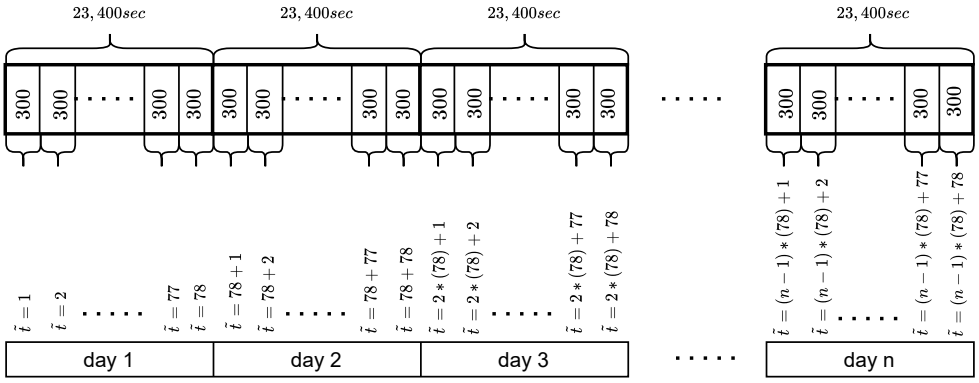

Figure 4. Each day is partitioned into 78 intervals of 300 seconds (5 minutes). A trimmed mean is applied to each of these intervals to obtain the value best representing the data within the regarded 5-min frame.

Clearly in order to aggregate the variables, one has to carefully choose a measure, that gives an accurate representation of the variable values within the considered aggregation interval. To aggregate the return variables $(X_{16,t}, X_{17,t}, X_{18,t})$, we simply compute the telescoping sums: we add 300 one second returns to obtain 5 min returns

$$X_{j,\tilde{t}} = \sum_{t=1+(\tilde{t}-1)300}^{300\tilde{t}} X_{j,t+1} - X_{j,t}, \quad \tilde{t} = 1, 2, 3, \ldots, (nm/300), \tag{32}$$

for $j \in \{16, 17, 18\}$, where n is the total number of regarded trading days and $m = 23400$ (amount of seconds contained in each trading day).

For the basic features $(X^{basic}_{1,t} - X^{basic}_{50,t})$, spread features $(X_{1,t} - X_{9,t})$ and volume features $(X_{10,t}\text{-}X_{15,t})$, we apply a trimmed mean (discard 10% on either ends) to each of the 78 300-second intervals contained in a trading day, to obtain the aggregated variables. We illustrate its application with an example: consider the following sequence of values $(X^h_{j,t})_{t=1}^{300}$, which consists of the first 300 entries of the feature $X^h_{j,t}$, corresponding to the first 5 minute interval on the first trading day, where $(j, h) \in I = \{(j, basic)|j = 1, \ldots, 50\} \cup \{(j, .)|j = 1, \ldots, 18\}$. The first step consists of computing the order statistics, i.e. arranging the values in ascending order $X^h_{j,(1)} \leq X^h_{j,(2)}, \ldots, \leq X^h_{j,(300)}$. Subsequently, 10% of the observations are removed from either sides, which amounts to 30 values being removed. Hence more generally we have

$$X^h_{j,\tilde{t}} = \frac{1}{300 - 2l} \sum_{t=1+(\tilde{t}-1)300+l}^{300\tilde{t}-l} X^h_{j,(t)} \quad \tilde{t} = 1, 2, 3 \ldots, (nm/300), \tag{33}$$

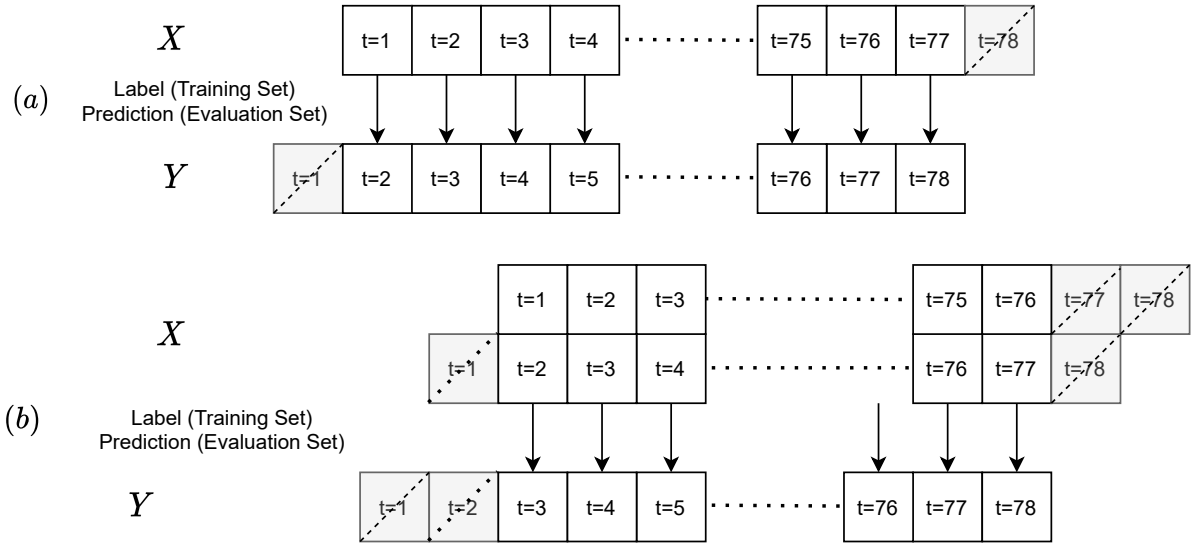

Figure 5. Concept of one step ahead forecast using *a*) lagged and *b*) additionally doubly lagged features applied to the first training day: Each day d consists of 78 data points. For the training (evaluation) set, the feature vector *a*) $\mathbf{x}_{j,\tilde{t}} \in \mathbb{R}^{\mathbb{P}}$ at time $\tilde{t} \in \{1, \ldots, 77\}$, *b*) $(\mathbf{x}_{j,\tilde{t}}, \mathbf{L}\mathbf{x}_{j,\tilde{t}}) \in \mathbb{R}^{2\mathbb{P}}$ at time $\tilde{t} \in \{2, \ldots, 77\}$ is labeled with (predicts) *a*) $Y_{\tilde{t}+1}$, $\tilde{t}+1 \in \{2, \ldots, 78\}$ *b*) $Y_{\tilde{t}+2}$, $\tilde{t}+2 \in \{3, \ldots, 78\}$.

for $(j,h) \in I$, where $l = 300 * \alpha$ with $\alpha = 0.10$. Henceforth we denote $\mathbf{X}$ as the set of all $p = 68$ aggregated features:

$$\mathbf{X} = \left\{X_1^{basic}, \ldots, X_{50}^{basic}, X_1, \ldots, X_{18}\right\}, \tag{34}$$

where $X_j^h = (X_{j,1}^h, \ldots, X_{j,(nm/300)}^h) \in \mathbf{X}$ with $(j,h) \in I$. Note, as we are interested in the one step ahead forcast (i.e we want to predict the trading signal for time $t+1$ using the information at time t), our regarded features are lagged by nature. Evidently, the number p of features can be increased significantly by additionally including lagged features. In particular, we additionally consider features that are doubly lagged, hence increasing the number of features from p to $2p$. We denote the corresponding set as $\mathbf{LX}$. For illustration: the lagged features on the first day would be given by $(X_{j,2}^h, X_{j,3}^h, \ldots, X_{j,77}^h)$ and the corresponding doubly lagged features as $(X_{j,1}^h, X_{j,2}^h, \ldots, X_{j,76}^h)$, $(j,h) \in I$. Lastly, due to their varying scales (e.g. volume vs. spread), we standardize all features. Once the features are established, we can proceed to construct training and evaluation sets:

4.5. Training and evaluation sets

In order to define training and evaluation sets, the feature vectors have to be provided with appropriate labels. Using the feature vector at time $\tilde{t}$, we want to predict the directional movement of the mid-quote stock returns at time $\tilde{t}+1$, to determine whether to buy or sell our stock at time $\tilde{t}+1$. Hence with the

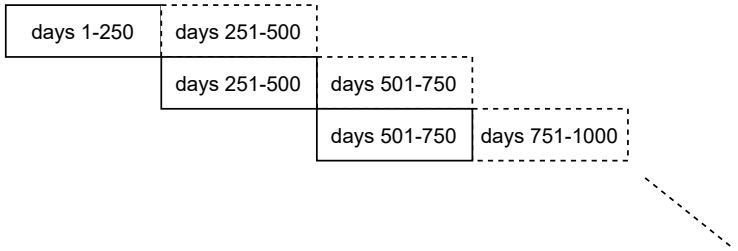

Figure 6. 250 shifting day approach: days in the boxes with solid lines are used as training sets, the subsequent boxes with dashed lines are used as evaluation sets.

dependant variable given by

$$Y_{\tilde{t}} = \begin{cases} 1 & \text{if } X_{18,\tilde{t}} \geq 0 \\ -1 & \text{if } X_{18,\tilde{t}} < 0 \,, \end{cases} \tag{35}$$

we define the training set as $L = \{(x_{\tilde{t}}, y_{\tilde{t}})\}$, where either $x_{\tilde{t}} \in \mathbb{R}^p$ with $p \in \{1, \ldots, 68\}$ or $x_{\tilde{t}} \in \mathbb{R}^{2p}$ with $2p \in \{2, 4, \ldots, 2 * 68\}$ (depending on the inclusion of doubly lagged features) is the feature vector at time $\tilde{t}$ and $y_{\tilde{t}} \in \{-1, 1\}$ is the corresponding true label indicating whether a buy signal ($Y_{\tilde{t+1}} = 1$) or sell signal ($Y_{\tilde{t+1}} = -1$) is generated at time $\tilde{t} + 1$.

When regarding the feature set $\mathbf{X}$, which by definition consists of lagged variables, the features at time $\tilde{t}$ are used to forecast the value of the dependent variable at time $\tilde{t} + 1$. Hence to construct the training and evaluation sets, the 78-th (last) entry has to be omitted from each day of the feature vectors, while the first entry has to be omitted from the dependant variable Y every day to obtain the appropriate corresponding labels (see Figure 5.*a*).

However, when regarding the feature set $\mathbf{LX}$, which consists of lagged X_j^h and additionally doubly lagged features LX_j^h, the number of possible labels and predictions diminish from 77 to 76 per day (see Figure 5.*b*).

Having established the appropriate labels for the features, we next have to determine the size of the training and evaluation sets. In contrast to the studies by Kercheval and Zhang (2015), Nousi et al. (2019) and Kearns and Nevmyvaka (2013), we use much larger limit order book data sets that cover slightly more than eleven years. We use a whole trading year, consisting of 250 days, to form our training set to see whether we can capture annual trends and characteristics of the regarded data. Since depending on the inclusion of additionally lagged features, each day provides us with $u \in \{76, 77\}$ pairs of data points, our training set T consists of $250u$ pairs.

The training set T is succeeded by the evaluation set E, which also covers a whole trading year and is used to determine how well the machine learning model has learned from the training set T. As the statistical characteristics of financial time series change over time, we continuously update our training and evaluation set, in order to keep up with changing trends and patterns in the limit-order book data. We employ a 250-days shifting window approach (see Figure 6): using the first 250 days ($250u$ data points) (training set T_1), the intraday buy and sell signals of the subsequent 250 days ($250u$ values) (evaluation set E_1) are predicted, then E_1 is used as the training set T_2 to once again predict the intraday buy and sell signals for the subsequent 250 days (evaluation set E_2) etc. until the last training and evaluation set.

Once the training and evaluation sets are established, the next step entails defining meaningful *tuning parameters* of the regarded machine learning algorithm.

4.6. Tuning machine learning models

Depending on the algorithm to be implemented, one has to first specify single or multiple tuning parameters. How should the tuning parameters be chosen? Clearly, before the tuning procedure can be conducted, a feature set $\bar{\mathbf{x}} \in \mathbb{R}^{\mathbb{P}}$ has to be selected first, where depending on the inclusion of doubly lagged features $p \in \{1, \ldots, 68\}$ or $p \in \{2, 4, \ldots, 2 * 68\}$. While the task of selecting features is based on the trial and error approach and hence varies significantly across different cases, a general guideline for selecting features for tuning procedures is to use a modest amount of features that cover the whole spectrum of available information as well as possible. Once the feature set $\bar{\mathbf{x}}$ is established, there are two ways to find possibly useful tuning parameters.

The common way is to regard various models that adopt a wide spectrum of tuning parameter values or tuning parameter combinations, when more than one tuning parameter is involved, and to compute their *predictive performance* (Accuracy) to establish which tuning parameter or tuning parameter combination yields the best result, i.e. exhibit the highest number of correct classifications. When using the 250-day shifting window approach, this translates into computing

$$PP = \Big(\frac{n}{250} - 1\Big)\Big(\frac{1}{250u} \sum_{k=(l-1)250u+1}^{(l-1)250u+250u} \mathbb{1}_{y_k = \tilde{y}_k}\Big), \quad l = 1, \ldots, (n/250) - 1, u \in \{76, 77\},$$

where y_k is the actual label (trading signal) and $\tilde{y}_k$ is the predicted label. Recall, $u = 77$ when the lagged features are used and $u = 76$, when the double lagged features are additionally included.

The second way to find possibly useful tuning parameters entails plots of cumulative returns. For various values of a tuning parameter or for various combinations of tuning parameters (in case of more than one tuning parameter), we can depict the cumulative returns of the corresponding trading strategies generated by the machine learning methods. This is a highly effective method to analyze how the performance of competing trading strategies change over time and to identify which strategies deliver good results in different time periods. Using solely the total sum of returns as a measure for the trading performance can be severely misleading hence it is imperative to consider the evolution of the cumulative sums over time. Indeed, suppose we have a trading strategy that outperforms the benchmark in the distant past (e.g. 2010-2015) but then underperforms it in the recent past (e.g. 2016-2017). Then the total sum corresponding to this strategy might still be a lot higher than that of the benchmark although its performance was inferior in the recent past - which is clearly of relevance - therefore delivering a deceptive result.

Thereby, for fixed values of a tuning parameter or parameters, the first training set T_1, corresponding to the first 250 days, is used to train the model, after which the model provides a trading signal, i.e. $y_{\bar{t}}$ for each of the feature vectors $x_{\bar{t}}$ contained in the evaluation set E_1. Due to the 250 days shifting window approach, this process is repeated with E_1 being used as the training set T_2 to train the model, so that the model can predict the trading signals for the feature vectors $x_{\bar{t}}$ in E_2. This is repeated until the last training and evaluation sets are reached. Then the cumulative returns are obtained by multiplying the

predicted trading signals $y_{\tilde{t}}$, with the corresponding returns $X_{18,\tilde{t}}$, i.e.

$$S_\tau = \sum_{\tilde{t}=250*u+1}^{\tau} X_{18,\tilde{t}+1} y_{\tilde{t}}, \quad \tau = 250u + 2, 250u + 3, \ldots, nu \text{ and } u = 77 \tag{36}$$

$$S_\tau = \sum_{\tilde{t}=250*u+1}^{\tau} X_{18,\tilde{t}+2} y_{\tilde{t}}, \quad \tau = 250u + 2, 250u + 3, \ldots, nu \text{ and } u = 76 \tag{37}$$

with $S_{250u+1} = X_{18,250u+2} * y_{250u+1}$ for $u = 77$ and $S_{250u+1} = X_{18,250*u+3} * y_{250u+1}$ for $u = 76$, depending on the inclusion of additionally lagged features (see Figure 5).

This approach has the following advantage over the tuning procedure based on predictive performance: The predictive performance only differentiates between correct and incorrect classifications and does not allow for different *degrees of misclassifications*. For instance, when a buy signal for a significantly small (absolute) negative return is generated, it clearly has a smaller overall effect on the cumulative return than when a buy signal is generated for a larger (absolute) negative return. While this concept can be clearly illustrated by plots of cumulative returns, the predictive performance evaluation method is too limited to capture this concept. Indeed, it does not matter how small or large the (absolute) negative return is, a misclassification in that case is just that: a missclassification.

4.7. Optimizing machine learning models

Once the "optimal" tuning parameters are found, the next step entails finding the "optimal" feature vector. Clearly, considering the set of all features, there is a generous amount of feature vectors that can be constructed. Trying every single combination will inevitably pose significant challenges with regard to computer memory (size) and computing power (speed), hence it is crucial to choose feature vectors systematically. We therefore propose the following procedure, where we only consider the features $X_1 - X_{18}$: First we start off by considering 1-dimensional feature vectors X_i, $i = 1, \ldots, 18$, to establish the feature $X_i \in \mathbf{X}$, which provides the most profitable 1-dimensional trading strategy. Secondly, including X_i, we regard all 2-dimensional feature vectors $x = (X_i, X_j)$, where $j \in \{1, \ldots, 18\} \setminus \{i\}$, to once again determine which 2-dimensional feature vector corresponds to the most profitable strategy. This process can be repeatedly applied. Thereby the resulting k-dimensional feature vector $\bar{\mathbf{x}} \in \mathbb{R}^k$, where $1 \leq k << 18$ consists of $k - 1$ fixed features and a varying $k - th$ feature, which covers all the features with the exception of the $k - 1$ features that are already included as a fixed component. Note, as the mid-quote returns X_{18} are based on best ask and best bid prices, the information content of X_{18} does not significantly vary from that provided by the ask X_{16} and bid X_{17} returns. Hence if one of the features X_j, $j \in \{16, 17, 18\}$ is contained in the feature vector $\bar{\mathbf{x}}$ and X_i with $i \in \{16, 17, 18\} \setminus \{j\}$, then it suffices to either only include X_i or X_j.

In the last step, we repeat the above mentioned process by additionally including the lagged variables of each respective feature, therefore doubling the number of predictors contained in each feature vector and reducing the number of predictions to 76 from 77 per day.

Subsequently we can establish which feature vector $\hat{\mathbf{x}}$ delivers the optimal output. Hence the optimal prediction model is given by the machine learning algorithm equipped with 1) the tuning parameters obtained by following the procedure in Section 4.6 and 2) the feature vector $\hat{\mathbf{x}}$ obtained by following the process described above.

We now proceed to apply our methodology to high-frequency data:

5. Empirical analysis

For our empirical analysis we download the message and order book file data of the iShares Core S&P 500 ETF (IVV) from LOBSTER and extract the features presented in Table 3 as described in Section 4.1 for the time period 27.06.2007 to 30.04.2019, resulting in $n = 2980$ trading days.

Subsequently we proceed with cleaning the data: We first conduct the data cleaning procedure for the price and volume variables (variables 1–20 from Table 3). As a result of the first step, which entails computing the daily percentage of missing values, we remove 27 early-closure days (the day before Independence Day, the day after Thanksgiving, and Christmas Eve, respectively) and, in addition, January 9, 2019 due to their large amount of daily missing values. Secondly, when the relative spreads (28) are greater than 0.015, we set the values of the thereby affected variables to missing as described in (29). Third, we examine the log volumes for outliers. As Figure A1 clearly illustrates, our data does not exhibit any outliers, hence there is no need to set additional values to missing. After that we set level volumes at time $t \in \{1, 2, 3, \ldots, (m2980)\}$ to missing, whose corresponding level prices at time t significantly deviate from the best price (first level price) at time t (see (30)). In the last step, we compute the log mid-quote, ask and bid returns to identify and consequently omit days that exhibit a return greater than 0.015, which leads to the elimination of the following days: September 19, 2008, May 6, 2010 and August 24. Hence in total 2949 days remain. We then conduct the data cleaning procedure of the remaining 30 variables (variables 31–50 from Table 3). As for the price and volume variables, we first compute the daily percentage of missing values for each of these variables. Thereby the last 30 variables exhibit a considerable amount of missing values per day (see Figure A2 - note, since variable j, for $j \in \{21, 24, 27, 30, 33, 36, 39, 42, 45, 48\}$, is used to construct variable $j + 1$ and $j + 2$, it is sufficient to compute the missing values for variable j only). Omitting these days, leads to a significantly minor set of remaining days for each of the last 30 variables. Hence due to their sparse information content we discard these variables. Lastly we compute the features presented in Table 4 before applying the data aggregation procedure from Section 4.4, to obtain $X_{j,\tilde{t}}$ $j = 1, \ldots, 18$ and $X^{basic}_{i,\tilde{t}}$ $i = 1, \ldots, 20$. However, since $X_{j,\tilde{t}}$ $j = 1, \ldots, 18$ are defined using $X^{basic}_{i,\tilde{t}}$ $i = 1, \ldots, 50$, we henceforth only regard the features $X_{j,\tilde{t}}$ $j = 1, \ldots, 18$ to avoid repetition of information content, hence $p = 18$.

Subsequently, we utilize the 250 day shifting window approach to train and evaluate our machine learning models. Note, since our data set covers 2949 days in total, our last evaluation set only consists of 199 opposed to 250 days (see Figure 7).

As described in Section 4.6, a feature set to be used in the course of the tuning procedure of the respective machine learning algorithm has to be defined. As we regard three categories of features: 1) spreads ($X_{1,\tilde{t}} - X_{9,\tilde{t}}$), 2) volumes ($X_{10,\tilde{t}} - X_{15,\tilde{t}}$) and 3) returns variables ($X_{16,\tilde{t}} - X_{18,\tilde{t}}$), we use a representative of each class to cover the range of these categories. In particular, we regard the following feature vector

$$\bar{\mathbf{x}} = (X_1, X_{15}, X_{18}), \tag{38}$$

to conduct the tuning procedure.

Once the optimal tuning parameters are established, the next step entails finding the optimal feature vector such that the trading strategy generated by the corresponding machine learning model is the overall most profitable. Following Section 4.7, we first start off by considering 1-dimensional feature vectors X_j, where $j \in J = \{1, \ldots, 18\}$, to establish which feature $X_j \in \mathbf{X}$ facilitates the generation of the most profitable trading strategy. Depending on the most informative feature X_j $j \in J$, we systematically

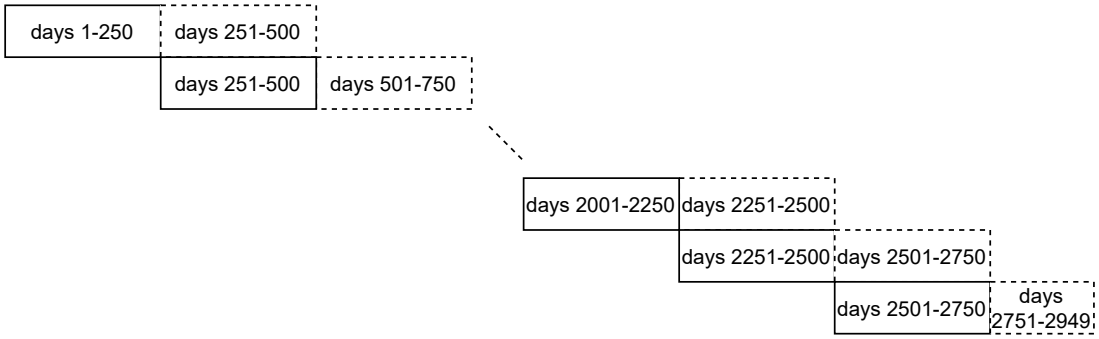

Figure 7. 250 shifting day approach for the data set IVV: days in the boxes with solid lines are used as training sets, the subsequent boxes with dashed lines are used as evaluation sets

define a collection of feature vectors (see Table A1 in the appendix) and subsequently use these for the optimization procedure to establish which feature vector translates into the most profitable strategy.

Having established the features, training and evaluation sets, as well as the feature vector to be used for the tuning procedure in addition to the collection of feature vectors that are to be used for the optimization process, we can finally proceed with applying the machine learning algorithm from Section 2. Thereby, due to the overall increasing trend of the regarded ETF, the buy and hold strategy (always long) is used as a benchmark when using the approach of plots of cumulative returns to compare the performance of competing trading strategies. For all our computations, we use the free statistical software *R* R Core Team (2021) and in particular the *R* package `e1071` to implement the machine learning algorithms:

5.1. SVM

As seen in Section 2.1, the support vector machine algorithm has different numbers of tuning parameters depending on the choice of the kernel. Indeed, in case of a linear kernel there is one parameter (cost C) to be tuned, in case of a polynomial kernel there are two (cost C and degree d of the polynomial) and lastly for the radial kernel there are also two parameters (cost C and γ, accounting for the smoothness of the decision boundary) to be tuned. Thereby there is no general consensus on what kernel and subsequently what tuning values to choose. For instance, Kercheval and Zhang (2015) employ the SVM algorithm using the polynomial kernel of order two and set the cost parameter C to 0.25 when working with LOB data, claiming that different choices for C do not have a significant affect on the results. On the other hand Nousi at al. (2019), also using LOB data, used a linear kernel and specified the cost parameter C by performing a 3-fold cross-validation on the training set, where possible values for C ranged from 0.00001 to 0.1.

For our empirical analysis we use the radial kernel, which requires the specification of two tuning parameters: γ and C. In Section 4.6, two different methods to determine the optimal tuning parameters were presented. Here we consider the method based on the plots of cumulative returns (see Table A2 and Table A3 in the appendix for the predictive performance of different models). Using the feature vector (38), we regard a generous amount of values for the tuning parameters to establish which combination of tuning values results in the most profitable trading strategy (see Figure 8). In particular we optimize

over the following set A,

$$A = \{(\gamma, C)|\gamma \in \Gamma, C \in \bar{C}\}, \tag{39}$$

where $\Gamma = \{0.1, 0.5, 1, 2, 3, 5, 8, 10\}$ and $\bar{C} = \{0.1, 0.5, 0.7, 1, 2, 3, 5, 8, 10, 30, 50, 100\}$. The reason we consider values with minor differences (such as $\gamma = 0.1$ and $\gamma = 0.5$ or $\gamma = 2$ and $\gamma = 3$ or $C = 0.7$ and $C = 1$), is to determine whether this disparity is sufficient to facilitate the production of vastly different results or whether there is barely any difference to be noticed. Note, to capture the range of the values, we regard values at the lower, middle and high end of the spectrum for both γ and the cost parameter C.

A visual inception of Figure 8, where each subplot corresponds to a fixed value of γ with varying values of γ, shows that overall there are four optimal tuning pairs that approximately achieve the same result: $A_1 = \{(0.1, C)|C \in \{10, 30\}\}$ and $A_2 = \{(1, C)|C \in \{5, 10\}\}$. Note that the trading strategies corresponding to the tuning pairs contained in A_1 (A_2) either coincide or exhibit an insignificant difference as can be seen in Figure 8.*a* (Figure 8.*b*). This supports the idea, that the choice of the cost parameters C do not significantly influence the outcome of a strategy, as long as they do not differ vastly. Having four tuning pairs to select from, we choose $(\gamma, C) = (0.1, 10)$ to conduct our further analysis. The choice of $C = 10$ is straightforward, as $(0.1, 10) \in A_1$ and $(1, 10) \in A_2$, the selection of $\gamma = 0.1$ over $\gamma = 1$ is discretionary.

Having defined the optimal tuning parameters, we now proceed to systematically define a collection of feature vectors as proposed in Table A1, to establish which of these feature vectors deliver the most profitable trading strategy. Thereby the first step entails regarding all 1-dimensional feature vectors $\mathbf{x} = X_j \in \mathbf{X}$, $j = 1, \ldots, 18$, to determine which feature is the most informative, i.e. results in the most superior trading strategy (see Figure 9.*a*). As Figure 9.*a* clearly illustrates, the return variables are the most informative with the bid returns $X_{i=16}$ (red) taking a slight lead, the lagged ($II - IV$) and doubly lagged ($II^* - IV^*$) feature vectors from Table 5 are of relevance (obtained from Table A1 in the Appendix). Finally, using feature vectors from Table 5, we can define the SVM models with the tuning parameters $(\gamma, C) = (0.1, 10)$ and the 250-day shifting window approach to determine which feature vector $\hat{\mathbf{x}}$ translates into the most profitable trading strategy (see Figure 9.*c-h*.).

Table 5. Collection of lagged and doubly lagged feature vectors considered for the optimization procedure of the svm algorithm with tuning parameters $(\gamma, C) = (0.1, 10)$, when the 1-dimensional feature X_i, $i \in \{16, 17, 18\}$ provides the highest cumulative return, i.e most profitable trading strategy.

Lags	Feature Vectors
1	$II : (X_{18,\tilde{t}}, X_{j,\tilde{t}}) \quad j \in J \setminus \{18\}$
	$III : (X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{j,\tilde{t}}) \quad j \in J \setminus \{1, 18\}$
	$IV : (X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{15,\tilde{t}}, X_{j,\tilde{t}}) \quad j \in J \setminus \{1, 15, 18\}$
2	$II^* : (X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}}) \quad j \in J \setminus \{18\}$
	$III^* : (X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{1,\tilde{t}}, X_{1,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}}) \quad j \in J \setminus \{1, 18\}$
	$IV^* : (X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{1,\tilde{t}}, X_{1,\tilde{t}-1}, X_{15,\tilde{t}}, X_{15,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}}) \quad j \in J \setminus \{1, 15, 18\}$

A visual inspection of Figure 9 shows, that the feature vector corresponding to the optimal strategy, is given by $\hat{\mathbf{x}} = (X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{15,\tilde{t}})$. Generally the feature vectors containing return variables (corresponding to red, darkred and gold in Figure 9) exhibit the best performance. Thereby the inclusion of additional features does not necessarily result in a better outcome (for instance, compare Figure 9.*c* to Figure

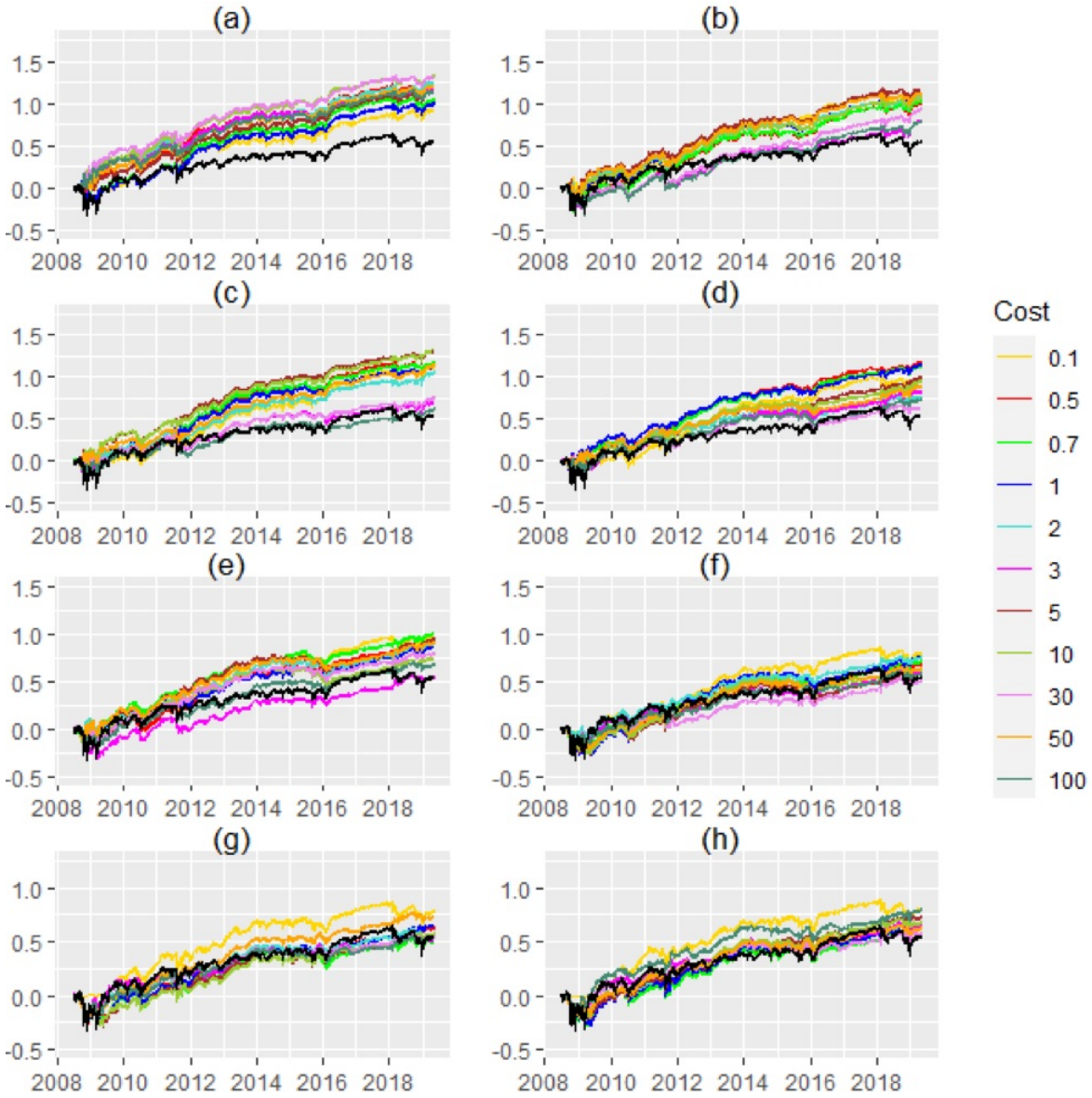

Figure 8. Tuning with 250-day shifting window approach: cumulative returns of the buy & hold trading strategy (black) and of the trading strategies generated by the SVM models using the feature set $\bar{\mathbf{x}}$ extracted from IVV and the radial kernel with $\gamma = 0.1, 0.5, 1, 2, 3, 5, 8, 10$ corresponding to *a*), *b*),..., *g*), *h*) respectively, and cost parameter C.

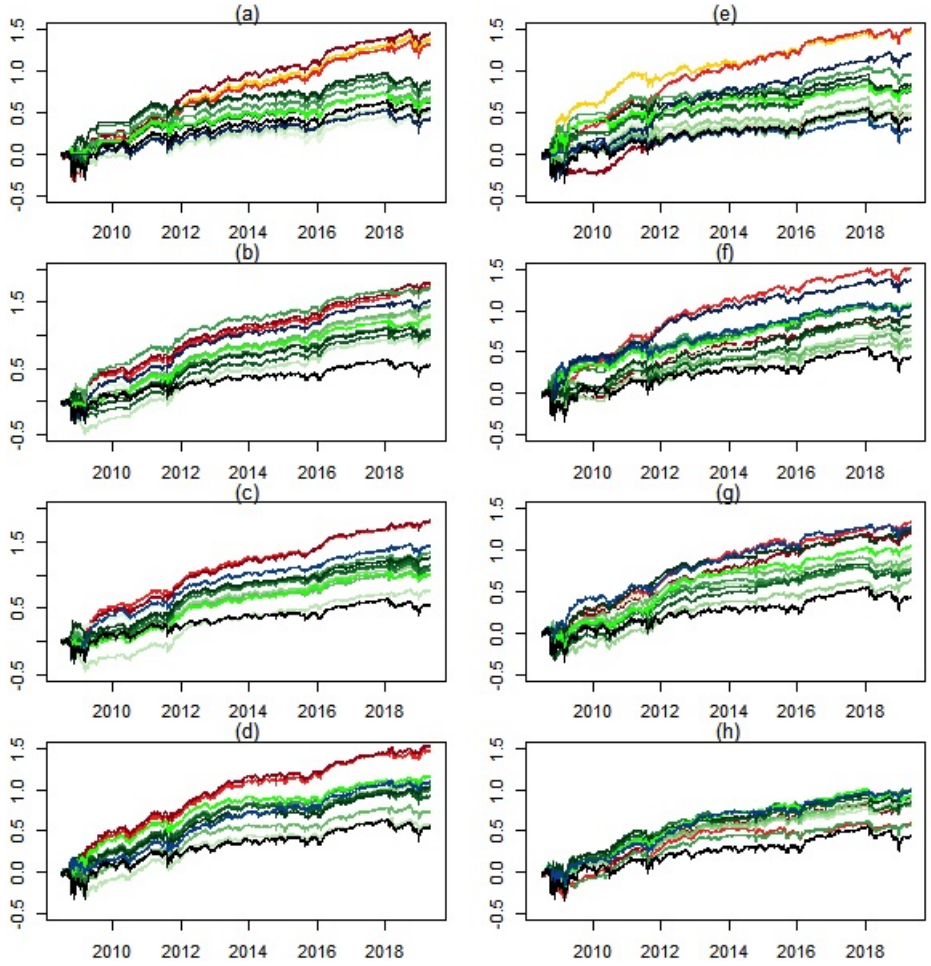

Figure 9. Optimization: cumulative returns of the buy & hold trading strategy (black) and of the trading strategies generated by the SVM models using the radial kernel with $\gamma = 0.1$ and $C = 10$, the 250-day shifting window approach using the lagged (first column) and doubly lagged (second column) feature vectors: (a) $(X_{j,\tilde{t}})$ $j \in J$, (b) $(X_{18,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{18\}$, (c) $(X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 18\}$, (d) $(X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{15,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 15, 18\}$, (e) $(X_{j,\tilde{t}}, X_{j,\tilde{t}-1})$ $j \in J$, (f) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{18\}$, (g) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{1,\tilde{t}}, X_{1,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 18\}$, (h) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{1,\tilde{t}}, X_{1,\tilde{t}-1}, X_{15,\tilde{t}}, X_{15,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 15, 18\}$. The color is determined by the value of j: the spread variables X_j, $j \in \{1, \ldots, 9\}$ correspond to the color green thereby adopting a darker shade of green as j increases; the volume variables X_j, $j \in \{10, \ldots, 15\}$ correspond to the color blue thereby adopting a darker shade of blue as j increases; the return variables X_j, $j \in \{16, 17, 18\}$ correspond to red, darkred and gold, respectively.

9.*d* or Figure 9.*g* to Figure.9 *h*). Furthermore note that including the second lags does not enhance the performance of the models. In fact, as Figure 9*h* clearly illustrates, the performance of the trading strategies significantly decline (lower total cumulative sum) in comparison to their single lagged counterpart in Figure 9.*d*.

Hence according to our empirical analysis, the SVM model using the tuning parameters $(\gamma, C) = (0.1, 10)$ and the feature vector $\hat{\mathbf{x}} = (X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{15,\tilde{t}})$ generates the optimal trading strategy.

In a previous study on volatility forecasting (see Reschenhofer et al. 2020a), we also used the iShares Core S&P 500 ETF. However, since large outliers occur more frequently in stock returns than in index returns, we examined additionally also a collection of fifteen DJIA-components. The results obtained with the index ETF were corroborated by the results obtained with the individual stocks. We assume that the same is true for the present study, particularly because outliers are less of a concern in directional forecasting than in volatility forecasting.

5.2. Additonal machine learning algorithms

In this subsection, the results obtained with support vector machines are compared with the results obtained with other machine learning algorithms. Figures 10, 11 and 12 display the results obtained by using the random forest algorithm with 3000 trees, bagging with an interaction depth of 4 and 3000 trees, and the k-nearest neighbors algorithm with 300 neighbors, respectively, using the exact same feature sets that were used to analyze the performance of the trading strategies based on the SVM algorithm in Figure 9. Obviously, there are no major differences in the performance between the different machine learning algorithms. Again, feature vectors containing return variables generally exhibit the best performance and the inclusion of additional features does not necessarily result in a better outcome. Also the inclusion of the second lags does not enhance the performance.

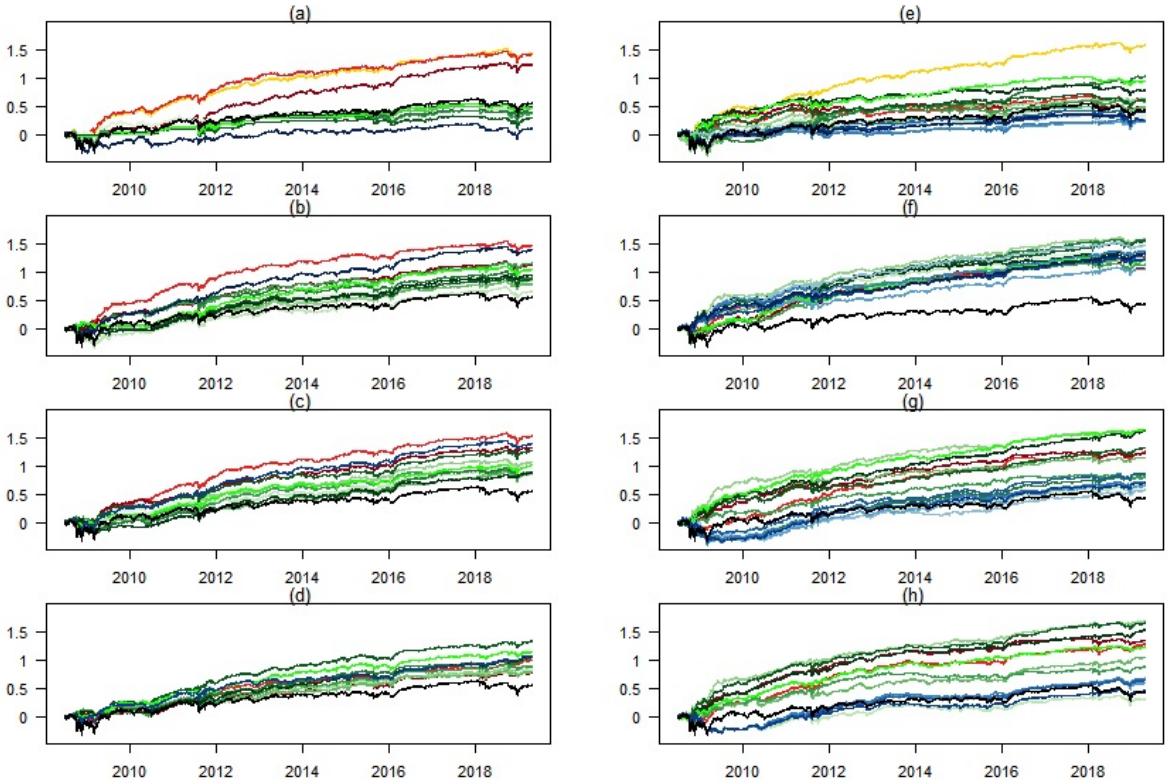

Figure 10. Optimization: cumulative returns of the buy & hold strategy (black) and the trading strategies generated by the random forrest algorithm with 3000 trees and a 250-day shifting window approach. The first column depicts the lagged features and the second column the doubly lagged feature vectors: (a) $(X_{j,\tilde{t}})$ $j \in J$, (b) $(X_{18,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{18\}$, (c) $(X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 18\}$, (d) $(X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{15,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 15, 18\}$, (e) $(X_{j,\tilde{t}}, X_{j,\tilde{t}-1})$ $j \in J$, (f) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{18\}$, (g) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{1,\tilde{t}}, X_{1,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 18\}$, (h) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{1,\tilde{t}}, X_{1,\tilde{t}-1}, X_{15,\tilde{t}}, X_{15,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 15, 18\}$. The color is determined by the value of j: the spread variables X_j, $j \in \{1, \ldots, 9\}$ correspond to the color green thereby adopting a darker shade of green as j increases; the volume variables X_j, $j \in \{10, \ldots, 15\}$ correspond to the color blue thereby adopting a darker shade of blue as j increases; the return variables X_j, $j \in \{16, 17, 18\}$ correspond to red, darkred and gold, respectively.

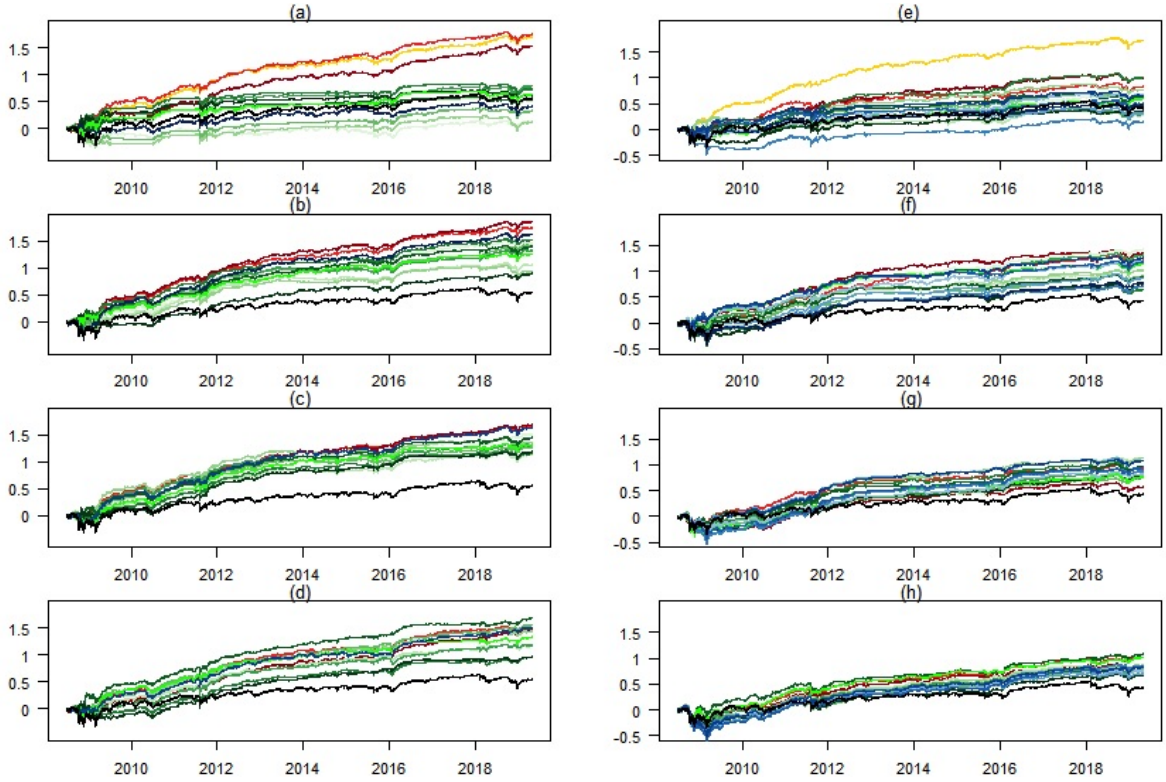

Figure 11. Optimization: cumulative returns of the buy & hold strategy (black) and the trading strategies generated using bagging with an interaction depth of 4 and 3000 trees and a 250-day shifting window approach. The first column depicts the lagged features and the second column the doubly lagged feature vectors: (a) $(X_{j,\tilde{t}})$ $j \in J$, (b) $(X_{18,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{18\}$, (c) $(X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 18\}$, (d) $(X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{15,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 15, 18\}$, (e) $(X_{j,\tilde{t}}, X_{j,\tilde{t}-1})$ $j \in J$, (f) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{18\}$, (g) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{1,\tilde{t}}, X_{1,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 18\}$, (h) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{1,\tilde{t}}, X_{1,\tilde{t}-1}, X_{15,\tilde{t}}, X_{15,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 15, 18\}$. The color is determined by the value of j: the spread variables X_j, $j \in \{1, \ldots, 9\}$ correspond to the color green thereby adopting a darker shade of green as j increases; the volume variables X_j, $j \in \{10, \ldots, 15\}$ correspond to the color blue thereby adopting a darker shade of blue as j increases; the return variables X_j, $j \in \{16, 17, 18\}$ correspond to red, darkred and gold, respectively.

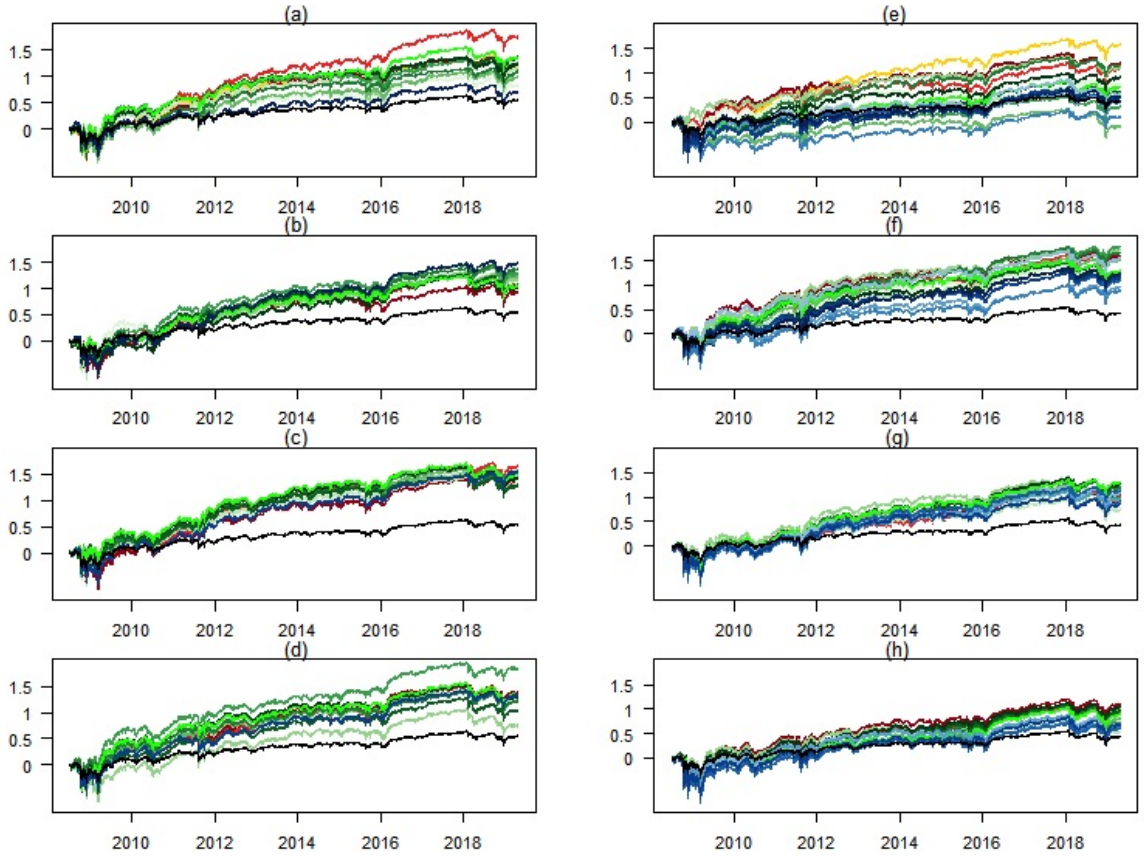

Figure 12. Optimization: cumulative returns of the buy & hold strategy (black) and the trading strategies generated using the k-nearest neighbors algorithm with 300 neighbors and a 250-day shifting window approach. The first column depicts the lagged features and the second column the doubly lagged feature vectors: (a) $(X_{j,\tilde{t}})$ $j \in J$, (b) $(X_{18,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{18\}$, (c) $(X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 18\}$, (d) $(X_{18,\tilde{t}}, X_{1,\tilde{t}}, X_{15,\tilde{t}}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 15, 18\}$, (e) $(X_{j,\tilde{t}}, X_{j,\tilde{t}-1})$ $j \in J$, (f) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{18\}$, (g) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{1,\tilde{t}}, X_{1,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 18\}$, (h) $(X_{18,\tilde{t}}, X_{18,\tilde{t}-1}, X_{1,\tilde{t}}, X_{1,\tilde{t}-1}, X_{15,\tilde{t}}, X_{15,\tilde{t}-1}, X_{j,\tilde{t}-1}, X_{j,\tilde{t}})$ $j \in J \setminus \{1, 15, 18\}$. The color is determined by the value of j: the spread variables X_j, $j \in \{1, \ldots, 9\}$ correspond to the color green thereby adopting a darker shade of green as j increases; the volume variables X_j, $j \in \{10, \ldots, 15\}$ correspond to the color blue thereby adopting a darker shade of blue as j increases; the return variables X_j, $j \in \{16, 17, 18\}$ correspond to red, darkred and gold, respectively.

6. Discussion and conclusions

In their review of recent studies on financial forecasting and trading, Reschenhofer et al. (2020b) identified many weaknesses such as unsuitable benchmarks, too short evaluation periods and nonoperational trading strategies. Their main conclusion was that apparently good forecasting performance does, in general, not translate into profitability once realistic transaction costs and the effect of data snooping are taken into account. While in the case of high-frequency trading, some peculiarities of intraday returns, e.g., autocorrelation, can straightforwardly be employed for directional forecasting (see Reschenhofer et al. 2020a, for the application to volatility forecasting and Hansen and Lunde, 2006, for general issues related to market microstructure noise), the adequate consideration of transaction costs requires the precise knowledge of the order book at any time and is therefore extremely difficult if not impossible (because of the critical dependence on the amount to be invested). However, it is quite clear that the costs of frequent trading are very high and can only be recouped if the prediction accuracy is very high. So the first step is to look for promising trading strategies in the absence of transaction costs. Employing machine learning models and rebuilding these models periodically, we hope to be able to find patterns which are stable over short to medium periods of time and can therefore be used to make profitable trading decisions. Indeed, we achieve significant predictability when we apply support vector machines to high-frequency order book data. However, the performance is only good but not great. It is comparable to the performance that could be achieved with simple methods in past periods of high autocorrelation in daily returns. There appears to be no extra gain due to high frequency, high dimensionality and artificial intelligence (machine learning). Again, we must assume that most of the possible profits will be eaten up by the transaction costs (most notably the bid-ask spread). The steady decrease in transaction costs over time is offset by the increase in trading frequency when we switch from daily trading to intraday trading. The observed predictability is mainly due to the inclusion of only one variable, namely the last price change, whereby it is irrelevant whether we use bid prices, ask prices or mid prices. The inclusion of additional variables from the order book does not result in any significant improvement (see Figure 9). We obtain similar findings when we use random forests (see Figure 10) or bagging (see Figure 11) or a k-nearest neighbors algorithm (see Figure 12) instead of support vector machines. In this paper we proposed trading strategies based on machine learning algorithms with specifically very straightforward specifications as the aim was to provide a overall general framework for the construction of trading strategies by consciously avoiding excessive data snooping. As we were unable to find high dimensional patterns in the order book that could be used for trading purposes, our findings imply that the construction of better performing trading strategies might inevitably rely on being tailored to the regarded dataset at hand (data snooping).

Acknowledgments

The authors are grateful to Nikolaus Hautsch for facilitating the access to LOBSTER. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

Cai C, Zhang Q (2016) High-frequency exchange rate forecasting. *Eur Financ Manage* 22: 120–141. https://doi.org/10.1111/eufm.12052

Cont R (2011) Statistical modeling of high-frequency financial data. *IEEE Signal Proces Mag* 28: 16–25. https://doi.org/10.1109/MSP.2011.941548

Fletcher T, Shawe-Taylor J (2013) Multiple kernel learning with fisher kernels for high frequency currency prediction. *Computat Econ* 42: 217–240. https://doi.org/10.1007/s10614-012-9317-z

Frömmel M, Lampaert K (2016) Does frequency matter for intraday technical trading? *Financ Res Lett* 18: 177–183. https://doi.org/10.1016/j.frl.2016.04.014

Gao K, Luk W, Weston S (2021) High-Frequency Trading and Financial Time-Series Prediction with Spiking Neural Networks. *Wilmott* 18–33. https://doi.org/10.1002/wilm.10927

Han J, Hong J, Sutardja N, et al. (2015) Machine learning techniques for price change forecast using the limit order book data. *Working Paper.*

Hansen P, Lunde A (2006) Realized variance and market microstructure noise. *J Bus Econ Stat* 24: 127–161. https://doi.org/10.1198/073500106000000071

Huang W, Nakamori Y, Wang S (2005) Forecasting stock market movement direction with support vector machine. *Comput opera res* 32: 2513–2522. https://doi.org/10.1016/j.cor.2004.03.016

Kearns M, Nevmyvaka Y (2013) Machine learning for market microstructure and high frequency trading. *High Frequency Trading: New Realities for Traders, Markets, and Regulators* Risk Books London, UK

Kercheval A, Zhang Y (2015) Modelling high-frequency limit order book dynamics with support vector machine. *Quantitat Financ* 15: 1315–1329. https://doi.org/10.1080/14697688.2015.1032546

Krämer W (1998) Note Short-term predictability of German stock returns. *Empir Econ* 23: 635–639. https://doi.org/10.1007/s001810050040

Krollner B, Vanstone Bruce, Finnie G (2010) inancial time series forecasting with machine learning techniques: a survey. *ESANN*,

Lakonishok J, Smidt S (1988) Are seasonal anomalies real? A ninety-year perspective. *Rev financ stud* 1: 403–425. https://doi.org/10.1093/rfs/1.4.403

LOBSTER: Limit Order Book System - The Efficient Reconstructer. LOBSTER Team. Available from: `https://lobsterdata.com/info/DataStructure.php`.

Majhi R, Panda G, Sahoo G (2009) Development and performance evaluation of FLANN based model for forecasting of stock markets. *Expert syst Appl* 36: 6800–6808. https://doi.org/10.1016/j.eswa.2008.08.008

Nousi P, Tsantekidis A, Passalis N, et al. (2019) Machine learning for forecasting mid-price movements using limit order book data. *Ieee Access* 7: 64722–64736. https://doi.org/10.1109/ACCESS.2019.2916793

Ntakaris A, Magris M, Kanniainen J, et al. (2018) Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *J Forecast* 37: 852–866. https://doi.org/10.1002/for.2543

R: A Language and Environment for Statistical Computing R Foundation for Statistical Computing. Available from: `http://www.R-project.org/`.

Reschenhofer E (2010) Further evidence on the turn-of-the-month effect. *Bus Econ J.*

Reschenhofer E, Mangat M, Stark T (2020a) Volatility forecasts, proxies and loss functions *J Empir Financ* 59: 133–153. https://doi.org/10.1016/j.jempfin.2020.09.006

Reschenhofer E, Mangat M, Zwatz C et al. (2020b) Evaluation of current research on stock return predictability. *J Forecast* 39: 334–351. https://doi.org/10.1002/for.2629

Tay F, Cao L (2019) Application of support vector machines in financial time series forecasting. *omega* 29: 309–317. https://doi.org/10.1016/S0305-0483(01)00026-3

Tran D, Kanniainen J, Gabbouj M et al. (2019) Data-driven neural architecture learning for financial time-series forecasting. *arXiv preprint arXiv:1903.06751* https://doi.org/10.48550/arXiv.1903.06751

Zheng B, Moulines E, Abergel F (2012) Price jump prediction in limit order book *arXiv preprint arXiv:1204.1381* https://doi.org/10.48550/arXiv.1204.1381

AIMS Press