Research in International Business and Finance 65 (2023) 101958

Contents lists available at ScienceDirect

Research in International Business and Finance

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

Full length article

Entropic approximate learning for financial decision-making in the small data regime

Edoardo Vecchi [a], Gabriele Berra [a], Steffen Albrecht [b], Patrick Gagliardini [c], Illia Horenko [a,d,*]

[a] Università della Svizzera Italiana (USI), Faculty of Informatics, Institute of Computing, Via la Santa 1, 6962 Lugano, Switzerland
[b] University Medical Center of the Johannes Gutenberg-Universität, Institute of Physiology, 55128 Mainz, Germany
[c] Università della Svizzera Italiana (USI), Faculty of Economics, Institute of Finance, Via G. Buffi 13, 6900 Lugano, Switzerland
[d] Technical University of Kaiserslautern, Faculty of Mathematics, Chair for Mathematics of AI, 67663 Kaiserslautern, Germany

ARTICLE INFO

Dataset link: https://finance.yahoo.com/

Keywords:
Approximate learning
Entropic classification
Decision-making

ABSTRACT

Financial decision-making problems based on relatively few observations and several explanatory variables can be problematic for the common machine learning (ML) tools, since they cannot efficiently discriminate the relevant information. To investigate the challenges of this "small data" regime, we employ several state-of-the-art ML methods for predicting whether three selected stocks from the Swiss Market Index will outperform the market, by using, as classification features, a set of commonly used technical indicators. We show that the recently introduced entropic Scalable Probabilistic Approximation (eSPA) algorithm significantly surpasses its competitors in both prediction accuracy and computational cost. We then discuss the interpretability of the employed ML methods and suggest some statistically derived heuristics to select the most appropriate and parsimonious financial decision-making candidate model.

1. Introduction

Financial decision-making is a challenging task, in which both the behaviour of the economic agents and their expectations about an inevitably stochastic future play a major role (Ingersoll, 1987; De Bondt and Thaler, 1995). In particular, investment decisions are hindered by the volatility of the stock prices and by the fact that their future trends are hard to predict, thus forcing investors to rely solely on their experience, financial models and intuitions when searching for the optimal investment strategy. While the traditional financial literature mainly assumes the rationality of economic agents and the efficiency of financial markets (Markowitz, 1952; Fama et al., 1969; Fama, 1970), the research branch of behavioural finance (Goodell et al., 2023) analyses financial decision-making from the perspective of cognitive psychology, by highlighting the non-trivial role played by emotions and unconscious mental processes (Taffler, 2018). In this scenario, a partial automation of the decision-making process with a machine operating according to a set of rules – either pre-determined or dynamically derived from the input data – could definitely be beneficial, since it would eliminate the emotional component and improve the quality of the results by leveraging the machine ability to learn from past outcomes and mistakes. For this reason, towards the end of last century, made their appearance the first softwares aimed at monitoring the market and executing transactions, through pre-determined investment strategies that could be modified on the fly via some sort of learning mechanism (Hawley et al., 1990). Nowadays, the mathematical and statistical models deployed for any type of forecasting in the financial markets are becoming significantly more complex and computationally

* Corresponding author.
E-mail address: illia.horenko@usi.ch (I. Horenko).

https://doi.org/10.1016/j.ribaf.2023.101958
Received 29 January 2022; Received in revised form 17 March 2023; Accepted 5 April 2023
Available online 11 April 2023

intensive. Indeed, with the advent of internet and social networks, the number of variables that could potentially influence the stock market has increased exponentially, apparently leading investment decision-making problems to the "big data" regime (Agarwal et al., 2019). As a matter of fact, investors have to deal with a massive amount of structured and unstructured data, and the possibility of analysing them as fast as possible in order to extract valuable information has become a sought-after perk. The latter is particularly true for practitioners: while they still rely on traditional tools like fundamental analysis and technical analysis, they also start to become increasingly interested in more automated procedures, capable of handling efficiently the large volume of data at their disposal (Rasekhschaffe and Jones, 2019). From this perspective, machine learning (ML) algorithms – i.e., methods aimed at leveraging the available information to improve their performance in a given task like, e.g., financial decision-making – are becoming important tools in several branches of economics and finance research (Goodell et al., 2021). In particular, among the most recent potential application domains, we can mention cryptocurrency return forecasting (Liu et al., 2023), insurance fraud detection (Aslam et al., 2022), company default prediction (Ben Jabeur and Serret, 2023), and sentiment analysis of news and assessment of their impact on financial markets (Costola et al., 2023). And, in all these different scenarios, the aforementioned eventual increase in the amount of available information can definitely foster the development of ML methods with robust performances and *ad hoc* structures, capable of addressing appropriately the several new challenges arising in a "big data" setting (Subrahmanyam, 2019; Hasan et al., 2020).

In this paper, we want to focus on the application of ML algorithms to financial decision-making problems. Specifically, we want to assess whether – on the basis of the information provided by a set of technical indicators commonly used by practitioners – ML models are able to forecast if selected stocks from the Swiss Market Index (SMI) will outperform the market in a given time frame. Predicting whether a stock outperforms the market is an extremely challenging task, which can however provide valuable information to support, e.g., efficient positional portfolio strategies (Gagliardini et al., 2021). In fact, such positional allocation strategies can build on momentum- or reversal-type of behaviour documented for various market sectors and investment horizons (Avramov et al., 2006; Chan et al., 1996; Jegadeesh and Titman, 1993). According to the traditional financial literature and the efficient market hypothesis (Fama, 1970), technical analysis cannot generate excess returns, due to the fact that the latest available information coming from historical stock data is already priced in the market and, thus, obsolete and unusable. Despite the widespread success of technical indicators among practitioners, potentially due to their ease of use and interpretability, the financial literature has yet to reach a consensus concerning their relevance. For example, in Lukac et al. (1988) the authors tested twelve trading systems based on technical analysis, and observed that seven of them were able to achieve a positive gross return and four of them a positive net return. According to Shynkevich (2012), on the other hand, a trading strategy based solely on technical analysis is not good enough to outperform the classic buy-and-hold strategy. However, other studies stress the importance of technical indicators, by showing that they display statistically significant predictive ability, while providing complementary information over the business cycle with respect to, e.g., macroeconomic variables (Neely et al., 2014). Even the application of ML methods to financial decision-making problems based on technical analysis tends to results in contrasting findings on their effective performance and relevance. For example, in Allen and Karjalainen (1999), technical indicators are used in combination with a genetic programming model on historical data from the S&P 500 index, but they are found out to be incapable of generating positive returns in out-of-sample predictions. On the other hand, another study considers using genetic algorithms to compute *ad hoc* parameters for some classic technical indicators – including, e.g., the Relative Strength Index and Moving Average Convergence/Divergence – and shows that a buy-and-hold strategy generates lower returns (in both bear and bull markets) when compared to a trading strategy based on the signals received from the improved indicators obtained through the ML algorithms (Fu et al., 2013).

The main contribution of this paper consists in analysing critically the application of ML methods to stock forecasting, with a particular focus on the limitations and challenges imposed by this application domain. Over the last few years, the financial literature has displayed an increasing interest towards the use of ML models trained on technical indicators for stock forecasting (Borovkova and Tsiamas, 2019; Li et al., 2020; Nabipour et al., 2020), with long short-term memory (Hochreiter and Schmidhuber, 1997) – a particular type of artificial neural network with feedback connections – representing the key player in the majority of these studies (Li and Bastos, 2020). In general, the training phase of the ML algorithms requires a sufficiently large amount of data, since otherwise the whole learning process is heavily compromised and the models tend to quickly overfit the training set, consequently achieving a poor performance on the test set (Rasekhschaffe and Jones, 2019). In particular, depending on the chosen methodology, the amount T of available data instances (i.e., the realisations of the dependent variable) need to be significantly larger than the number D of features (i.e., the explanatory variables) in order to obtain models which are better than a random guess (Horenko, 2020; Vecchi et al., 2022). This limitation is clearly observable in the aforementioned studies, which result either in the inclusion of a limited number of technical indicators as model features (Nabipour et al., 2020) or in a statistically significant but modest model performance (Borovkova and Tsiamas, 2019). In general, we could argue that the presence of an insufficient number of data samples T with respect to the number of features D – indicated in this paper as "small data learning regime" – can be found in many problems dealing with financial and investment decisions. For example, financial asset management is hindered by a significant discrepancy between the length of the time series of asset prices and the potentially massive number of regressors that could be used to model the price variation, as well as by the low values of the signal-to-noise ratio (Israel et al., 2020). While long time series of daily returns are available for many assets, the use of shorter time spans may be more appropriate to deal with the potential non-stationarity of these series due to, e.g., structural breaks. Since this combination of small data and low signal-to-noise ratios can prove rather challenging for the standard ML algorithms, several tools have been developed to address this systematic scarcity of training data, with transfer learning (Weiss et al., 2016) and data augmentation (Shorten and Khoshgoftaar, 2019) representing two of the most widely used approaches. However, neither of them can be straightforwardly applied to financial data, due to the partial incompatibility of their underlying assumptions. Indeed, transfer learning relies heavily on the structural similarity between the source domain where the

model has been trained and the target domain where the previously acquired knowledge has to be applied. Such a strong assumption, however, goes far beyond the eventual positive or negative correlation that can be shown to exist between different financial assets, and transfer learning can then potentially deteriorate the model performance, by propagating noise or features specific to the source domain. On the other hand, data augmentation is based on the structural invariance properties of the input data, but financial time series are not images, and the fact that they cannot be neither rotated nor translated renders this alternative solution impractical. Given the highlighted shortcomings of the standard ML approaches, investigating potential alternatives is essential.

The results of this paper can be essentially divided in two parts. In the first part, we benchmark several state-of-the-art ML algorithms on the aforementioned prediction of stock outperformance over the market by using a set of technical indicators. The data set used in the simulations consists of $T = 186$ daily observations of three different binary response variables for each stock (corresponding to the presence/absence of stock outperformance after 1, 5 and 20 days respectively) and of $D = 63$ technical indicators used as explanatory variables, thus clearly configuring a small data learning problem as defined above. The results of the experiments show that the recently introduced entropic Scalable Probabilistic Approximation (eSPA) algorithm for entropic approximate learning (Horenko, 2020; Vecchi et al., 2022) dramatically outclasses its competitors in the considered classification problem, both in terms of prediction accuracy and computational time, thus providing valuable insights to support the financial decision-making process. In particular, the identification of a robust algorithm with an extremely low computational cost is a result of crucial importance for financial trading, since nowadays the main actors operating on the market are computers and algorithms and, consequently, the ability to process the up-to-date information faster than the competitors is a sought-after perk (Ammar et al., 2020). Furthermore, we carefully analyse the challenges imposed by the learning of small financial data, and show that the presence of several time-dependent groups of classification features cannot yet be efficiently addressed by the ML methods considered. In other words, the fact that the technical indicators identified as relevant for the stock outperformance prediction tend to change over time represents a major hindrance, which prevents us from tackling the shortage of observations in the training set by simply providing the ML models with longer time series.

In the second part of this paper, we focus on the interpretability of the resulting ML models and directly address all recent concerns about relying on learning tools which are, *de facto*, black boxes (Rudin, 2019). In particular, for each of the ML algorithms considered, we provide either a direct assessment of the importance of each feature in the stock outperformance prediction, or a series of proxies that allow us to better understand the model results and, *ceteris paribus*, the eventual repercussions of changes in the input data (e.g., Shapley additive explanation, partial dependency plots). The results show that only the entropic weights computed by the eSPA algorithm provide a clear interpretation of the importance of each technical indicator. Furthermore, since the latter method is able to discriminate between the relevant and irrelevant features, we are able to explore hidden patterns in the indicators that go beyond what is reasonable, directly interpretable, used by practitioners or commonly discussed in the literature. In particular, we found out that, among others, the Relative Strength Index, the Williams Percent Range and the Momentum do not play an active role in the stock outperformance prediction. In contrast, the Bollinger Bands and the Moving Average Convergence/Divergence computed on all available daily prices (i.e., Open, High, Low, Close and Adjusted Close) have been identified by the model as relevant explainers. Furthermore, we discovered that some unusual technical indicators without a direct financial interpretation were, indeed, discarded in the majority of the simulations performed by the eSPA algorithm. This analysis is then completed by a series of statistically derived heuristics – based on the Akaike Information Criterion and Bayesian Information Criterion – aimed at choosing the most explicative and parsimonious ML model among the different potential candidates obtained from multiple cross-validations.

The remainder of this paper is organised as follows. In Section 2, we outline the prediction problem and the simulation framework, while Section 3 describes the ML algorithms used in the experiments. In Section 4, we discuss the study results about stock outperformance prediction and model interpretability. Section 5 concludes the paper, by highlighting limitations and future challenges.

2. Problem description and simulation framework set-up

In this section, we provide details on the construction of the experimental data set and on the formulation of stock outperformance prediction as a binary classification problem. We also describe the response variables used in the simulations and the technical indicators exploited as predictive features.

2.1. Stock outperformance prediction as a binary classification problem

In order to investigate the potential applicability of ML algorithms to financial decision-making problems, we try to use them to forecast if the return of three selected stocks from the Swiss Market Index (SMI) will outperform the return of the market. The prediction is carried out on the basis of the information provided by a set of technical indicators – defined in the next subsection – which are used as explanatory features. Specifically, in the simulations we consider the following three stocks, which are chosen according to their market capitalisation at the time of writing: Nestlé S.A. (NESN.SW, highest capitalisation), Givaudan SA (GIVN.SW, median capitalisation) and Logitech International S.A. (LOGN.SW, lowest capitalisation). As anticipated, instead of predicting directly the return of the stock, we formulate the problem as a binary classification task, in which the return of the stock is compared with the return of the index in the same day to check if it is either higher (class 1) or lower (class 0). This type of analysis is consistent with the one performed in other recent studies, in which the response variable of interest is not the return of the stock itself, but rather its deviation from a given benchmark, like, e.g., the risk-free rate or the market index (Tan et al., 2019;

Table 1
List of the technical indicators used in the simulations.

Category	Technical indicators
Oscillators	Accumulation/distribution oscillator, Chaikin oscillator, Moving Average Convergence/Divergence, stochastic oscillator, acceleration, momentum
Stochastics	Chaikin volatility, Williams %R
Indexes	Negative volume index, positive volume index, relative strength index
Indicators	Accumulation/distribution line, Bollinger band, highest high, lowest low, median price, on balance volume, price-volume trend, typical price, volume rate of change, weighted close, Williams accumulation/distribution

Zhao and Cheng, 2022). We deem this problem formulation quite interesting, since the technical indicators – used as classification features – provide buy/sell signals derived from the historical stock data, while the relationship between the stock and the market is contained directly in the response variable. In this way, we can then truly test if the considered ML methods are able to extract information and hidden synergies from the data to tackle at best the classification task.

To provide a more formal definition of the problem, given a certain time instant t, we indicate the return of the SMI over the next n days as R^{SMI}_{t+n}, while the corresponding return of the stock can be expressed as R^{stock}_{t+n} (with *stock* replaced by the actual stock ticker when a distinction is necessary).[1] In the experimental results, n indicates the number of days over which the returns are computed and assumes the values 1, 5 and 20, with R_{t+1}, R_{t+5} and R_{t+20} thus indicating the return after 1 day, 5 days and 20 days, respectively. The decision of including different time horizons stems from the will of providing a better assessment of the ML model capabilities also with respect to response variables with different characteristics, especially concerning serial persistence (see Section 4.1). The prediction of the stock outperformance is then modelled as a binary classification problem, where the response variable $y^{(n)}_t$ – for time $t = 1, \dots, T$ and time horizon $n = 1, 5, 20$ – can only assume the values:

$$y^{(n)}_t = \begin{cases} 1, & \text{if } R^{\text{stock}}_{t+n} \geq R^{\text{SMI}}_{t+n}, \\ 0, & \text{if } R^{\text{stock}}_{t+n} < R^{\text{SMI}}_{t+n}. \end{cases} \tag{1}$$

In order to predict whether a stock will outperform the market, we use a set of technical indicators computed from the historical Open-High-Low-Close-Volume values of each stock and whose details are provided in Section 2.2.

2.2. Feature space and technical indicators

In order to tackle the binary classification task and determine whether a stock outperforms the market, all the ML methods described in Section 3 need, as input, a set of explanatory numerical features for each realisation of the response variable. In this contribution, the features consist in a group of technical indicators (Lo et al., 2000) computed on the historical Open-High-Low-Close-Volume values of each of the three stocks. While technical indicators are frequently used by practitioners – together with other complementing techniques, like fundamental analysis – to formulate educated predictions on the stock market,[2] they have been, for several years, partially neglected by the academic literature (Neely et al., 2014). However, some recent contributions have highlighted that the use of technical indicators can provide a valid addition to financial forecasting models (Gao and Chai, 2018; Wang et al., 2020), also in those scenarios characterised by high volatility, like the prediction of bitcoin returns (Huang et al., 2019). This evidence prompted us to consider, in the construction of the feature space, 22 commonly used technical indicators – including leading indicators, lagging indicators, indexes, stochastics, centred oscillators and banded oscillators – whose complete list is reported in Table 1.

Starting from these technical indicators, we then computed, for each of the response binary variables, the 63 explanatory features used in the experiments. The apparent discrepancy between the number of technical indicators and the final number of features can be motivated as follows. First of all, some of the technical indicators are actually associated with more than one variable: for example, the computation of the Bollinger bands leads to 3 values, corresponding – respectively – to the upper band, the middle band and the lower band. Likewise, the stochastic oscillator is expressed through 4 different quantities: the two fast stochastics (fast%K and fast%D) and the two slow stochastics (slow%K and slow%D). Furthermore, since the aim of this paper is conducting an experimental study on the potential applicability of ML methods to stock outperformance forecasting, we introduced some alternative variants of commonly used technical indicators. For example, the Bollinger bands have not been computed only on the closing prices, but on each of the input historical variables (Open-High-Low-Close-Volume). These newly introduced indicators serve a double purpose: on

[1] The return is then calculated, both for the stock and for the index, simply as:

$$R_{t+n} = \frac{P_{t+n} - P_t}{P_t},$$

where P_t and P_{t+n} indicate the price today and after n days, respectively.

[2] For several success stories of practitioners using technical analysis see, e.g.: A. W. Lo, J. Hasanhodzic, *The heretics of finance: Conversations with leading practitioners of technical analysis*, Vol. 16, John Wiley and Sons, 2010.

the one hand, they give the possibility to the ML methods to find alternative or hidden patterns in the input data to solve at best the classification task, while – on the other hand – they serve as control variables to test the ability of the algorithms to discriminate the most relevant information. Indeed, as we will further discuss in Section 4, it is crucial that the ML methods can correctly identify the explanatory features which are relevant for predicting the presence or absence of stock outperformance over the market.

All technical indicators have been computed by using the functions provided in the *Financial Toolbox* of MATLAB, which represents – as indicated in Sections 3 and 4 – the programming language used for implementing the benchmark ML algorithms and the simulation framework. It is important to stress that, since all features are computed exclusively on the available historical information of each stock, none of them introduces any kind of hindsight bias in the solution of the binary classification problem. As a general remark, several of the technical indicators are computed over time intervals, whose length is usually arbitrary, and more tied to practice and experience than to a solid theoretical background. While we are aware that the time interval width could potentially have an impact on the predictive power of the indicator (Shynkevich et al., 2017), considering several different time horizons for each one of them would go beyond the scope of our analysis and would dramatically increase the number of features, thus making intractable the classification problem for the majority of the ML methods considered. Indeed, as we will more thoroughly explain in the next Section 2.3, the dimensionality of the data set used in the simulations already falls in the small data regime, and increasing even further the number of explanatory features would only contribute to worsen the overfitting and to reduce the predictive capability of the ML algorithms. For this reason – whenever a choice on the time interval was possible – we tried to create a balanced set of features by adopting either the most commonly used or recommended values (e.g., the 10-day moving average for the Chaikin volatility suggested by Chaikin, or the 14-day Relative Strength Index recommended by Wilder) or the standard values provided by MATLAB (usually directly derived from the theoretical examples presented in Achelis (2013)). In Appendix A, we report the complete list of the 63 features included in the experiments, as well as the formulas and parameters used for computing the technical indicators, together with a brief description.

2.3. Description and dimensionality of the data set

In order to construct the data set, we started by downloading[3] the daily prices of the Swiss Market Index (SMI) and the Open-High-Low-Close-Volume data of three of its underlying stocks – namely, Nestlé S.A. (NESN.SW), Givaudan SA (GIVN.SW) and Logitech International S.A. (LOGN.SW) – for the period between the 4th of January 2016 and the 30th of December 2020. As discussed in Section 2.1, the stocks have been selected according to their market capitalisation, thus allowing us to assess the eventual impact of this variable on the classification task. Starting from this historical information, we computed both the response variables according to Eq. (1) and the set of technical indicators acting as explanatory features, described in Section 2.2. In order to systematically assess whether the prediction horizon has an impact on the identification of the stock outperformance, we consider three different response variables for each stock. Specifically, by adopting the generalised notation introduced in Eq. (1), the three response variables considered are $y_t^{(1)}$, $y_t^{(5)}$ and $y_t^{(20)}$, which correspond to a stock outperformance over the market after 1 day, 5 days and 20 days, respectively. For example, the variable $y_t^{(5)}$ assumes the label 1 if the return of the stock computed over the next 5 days is higher than the return of the SMI over the same period, and 0 otherwise. As we will discuss in Section 4.1, computing the returns over a longer time frame contributes to smoothing out the high volatility of the daily returns and provides response variables which are more stable with respect to the set of features. Since the computation of the stock outperformance requires price information from the next days and the technical indicators are based on the historical information of the stocks, some time instances need to be excluded and the post-processing data set goes from the 19th of February 2016 to the 30th of November 2020, thus resulting in a total of $T = 1201$ observations, each with $D = 63$ features.

Given the current data set dimensionality, it seems that the classification problem does not belong to the "small data" learning regime, since we have at our disposal a sufficient number of observations of the response variable with respect to the number of features. Indeed, according to the empirical definition of "small data" regime mentioned in the introduction and provided in Horenko (2020), given the 63 classification features, approximately 880 instances of the response variable (i.e., 14 times the number of features) are sufficient to be able to obtain good classification results with the ML algorithms considered. Unfortunately, some preliminary tests confirmed that also financial data, as other kinds of data streams, are subject to concept drift (Webb et al., 2016), i.e., a phenomenon which occurs when some changes in non-observable factors have an impact also on the response variable. In financial time series analysis, this situation might be related to time-varying data generating processes and to the "model instability" resulting from, e.g., a structural break, which causes perturbations in the response variable and, as a consequence, in the predictive model itself (Pesaran et al., 2013). Structural breaks can be caused, e.g., by changes in the monetary policy, or by an abrupt leakage of sensible information, and are accompanied by a sudden change in the relative importance of the explanatory variables, which hinders the capability of performing accurate post-break predictions for the majority of the ML methods. As discussed in Section 4.1, in the simulations performed the effect of concept drift manifests itself as time-dependent set of features with varying importance in the classification process. In other words, some features, which were relevant – in a certain time frame – for the prediction of the class of the response variable, can become irrelevant or even hinder the classification task in a different time interval. Concept drift has a negative impact on ML models, due to the fact that they can end up incorporating past dynamics that are no longer present in the input data, and a solution to this problem is thus necessary to improve their performance. In our exploratory analysis, after a preliminary evaluation of the ML algorithms on the full data set, we decided to tackle this issue by reducing the size of the

[3] The data are publicly available on Yahoo Finance: https://finance.yahoo.com/.

training data set. In particular, empirical testing on samples with varying size has shown that a data set containing roughly 9 months of observations represents a good compromise, which allows to obtain meaningful classification results while not being excessively biased by the presence of groups of time-dependent features. After testing the ML models on the different 9-month intervals obtainable from the full data set and obtaining similar results, we decided to select – as reference data set to be used in the simulations – the period from the 1st of January to the 30th of September 2019, which contains $T = 186$ observations of the response variable. Given the dimensionality of the reference data set, which presents only 3 times more data instances than feature dimensions, we are then clearly operating in the "small data" learning regime, with all the challenges it entails. This will be particularly evident in the results presented in Section 4.1, where all the methods considered except eSPA tend to display an unsatisfactory performance.

3. Methodology and implementation details

Before presenting the experimental results, we briefly introduce the state-of-the-art ML algorithms used to solve the binary classification problem introduced in Section 2, while specific details on their MATLAB implementation and on the hyperparameter optimisation can be found in Appendix B. Specifically, we consider the following ML models: eSPA, random forest, generalised linear model with lasso regularisation, linear discriminant analysis, support vector machine, shallow neural networks and deep learning with long short-term memory.

3.1. The reference method: eSPA

As shown in the experimental results in Section 4, the eSPA method (Horenko, 2020; Vecchi et al., 2022) markedly outperforms all its competitors in the prediction of stock outperformance and can be, thus, considered our reference model. Specifically, it is an entropic supervised learning algorithm, which allows a straightforward interpretation of the classification results and a clear assessment of the importance of each feature (i.e., the technical indicators in our case) in the label prediction. The algorithm output can then be easily combined with information criteria to select the best model among the candidates obtained through cross-validation, as explained in Section 4.3. The MATLAB implementation considered in the experiments is based on the improved algorithm version introduced in Vecchi et al. (2022), which displays higher stability and lower computational cost (see Appendix B for additional details).

3.2. The other state-of-the-art ML algorithms

Random forest (RF). An ensemble learning algorithm which, by combining the results of several decision trees, tries to mitigate the overfitting of the training set while boosting the predictive performance (Breiman et al., 2017). In our analysis, the decision tree branches contain the values of the technical indicators for each response variable, while the leaves contain the class assignment (1 if the stock outperforms the market, 0 otherwise).

Generalised linear model with lasso regularisation (Lasso GLM). A generalisation of the standard linear regression, in which the response variables are assumed to be generated by a distribution belonging to the exponential family (James et al., 2013). Since the response variables $y_t^{(n)}$ can take only the values 0 or 1, we assume an underlying Binomial distribution and apply logistic regression to compute the model parameters, while improving the optimisation procedure through lasso regularisation (Tibshirani, 1996).

Linear discriminant analysis (LDA). A method which solves the classification problem by deriving a linear classifier from a linear combination of the features (Park and Park, 2008). In our analysis, we assume that the covariance matrix is the same across all classes and we invert it numerically by using the Moore–Penrose inverse.

Support vector machine (SVM). A supervised learning method capable of performing both linear and non-linear classification through an embedding of the input feature space into a higher dimensional space (Noble, 2006).

Shallow neural network (shallow NN). This term usually refers to a neural network characterised by a small number of hidden layers (Bianchini and Scarselli, 2014). In this context, we consider a shallow neural network with a single hidden layer and we include this methodology to provide an alternative to long short-term memory, which represents an example of deep learning.

Deep learning with long short-term memory (DL with LSTM). An artificial recurrent neural network which presents, unlike the aforementioned feedforward network, several feedback connections. LSTM (Hochreiter and Schmidhuber, 1997) represents probably the most widely known and utilised example of deep learning, with a long series of successful applications in several domains.

3.3. Practical details on the experimental setting

Before presenting the results, we briefly discuss some practical aspects of the experiments performed. First of all, unless otherwise specified, all the simulations have been performed on the reference data set described in Section 2.3. In each experiment, we divided the data set (containing $T = 186$ observations) in two parts: 80% of the data (i.e., 149 observations) was dedicated to the model training, while the remaining 20% (i.e., 37 observations) was used for testing the performance of the ML methods. In order to include statistically significant confidence intervals in each figure, we performed at least 30 cross-validations with random train/test splits,

by using, for each of them, the same proportion described above (80%/20%) to divide the data between training and test set. We evaluate the performance of the ML algorithms in terms of their area under the receiver operating characteristic curve (AUC). The latter metric assesses the ability of the methods to discriminate between the two classes (James et al., 2013): an AUC equal to 1 indicates that the presence or absence of stock outperformance can be identified with absolute certainty; an AUC of 0.5 indicates that the two classes are not distinguishable and that the model prediction consists in a random guess; finally, an AUC below 0.5 indicates that the algorithm is reciprocating, thus incorrectly identifying as stock outperformance the situations in which the return of the stock is lower than the return of the market, and vice versa. Finally, since the ML models rely on several hyperparameters that can be tuned in order to improve the classification performance and avoid overfitting, we performed – for each method – an extensive grid search, and in Appendix B we report all the values considered.

4. Discussion of the experimental results

In this section, we present and discuss the results of the experiments performed. We start by comparing the state-of-the-art ML methods described in Section 3 in the prediction of stock outperformance in the small data learning regime. In these simulations, we examine more closely the structure of the reference data set and provide evidence towards the presence of concept drift in our selected financial application. Specifically, we show empirically that the 1-day-ahead prediction of stock outperformance over the market cannot be improved by considering a longer time series as training input data, due to the presence of groups of time-dependent features. Actually, an increase in the size of the training set results detrimental for the training process, and we stress that this important issue needs to be carefully considered and addressed in potential future studies in this field. After showing that eSPA markedly outperforms all its competitors both in terms of prediction accuracy and computational cost, we delve on how the resulting ML models can be interpreted to provide either further insights into the financial decision-making process or additional information which can be incorporated in more complex trading strategies. Unfortunately, many of the ML methods considered tend to behave like *black boxes*, and do not allow a direct interpretation of the different features in the stock outperformance prediction, thus forcing us to rely on some proxies (like, e.g., Shapley values) to provide an interpretation of the output model. Lastly, since eSPA – the best-performing algorithm – produces clearly interpretable results by providing an entropic filtering of the features based on their importance in the classification task, we discuss some statistically derived heuristics to select the most appropriate model between the different candidates obtained through cross-validation. In particular, we show that a combination of performance metrics and model quality indicators, like the Akaike Information Criterion and Bayesian Information Criterion, can be useful to find a good balance between model quality and number of parameters.

4.1. Comparison of ML methods on stock outperformance prediction

The first part of the experiments consists in a thorough comparison of the ML algorithms in the outperformance prediction of NESN.SW, LOGN.SW and GIVN.SW over the SMI. As explained in Section 2, the data set used in the simulations consists of $T = 186$ observations, covering the period from the 1st of January to the 30th of September 2019, while the stock outperformance prediction is modelled as a binary classification problem, where we consider three different response variables (indicating, respectively, the 1-day-, 5-day- and 20-day-ahead forecast). The following simulations will show that the 1-day-ahead stock outperformance prediction is particularly challenging, mainly due to the lack of persistency in the behaviour of the response variable $y^{(1)}$. To immediately address this issue, as a preliminary analysis, we want to determine if there is any kind of structural difference between the response variables $y^{(1)}$ and $y^{(20)}$. In Fig. 1, we report the results obtained for LOGN.SW in these two cases while omitting the variable $y^{(5)}$, since it represents an intermediate scenario that would not add any information to this discussion.

Specifically, in panels A and B we plot the returns achieved by LOGN.SW and by the SMI over 1 day and 20 days, respectively, while in panels C and D we visualise the corresponding stock outperformance over the market, modelled as a binary variable according to Eq. (1). As we can notice, the values of $y^{(20)}$ tend to remain in the same state (either presence or absence of stock outperformance) for longer time frames, while – on the other hand – the variable $y^{(1)}$ is characterised by a constantly switching process without any kind of persistency. This reflects the overlapping periods behind the computation of two consecutive values of $y^{(20)}$. This kind of behaviour has, of course, major implications both on the training process and on how much the model features can explain the response variable. Indeed, we can reasonably assume that the values of the technical indicators – due to the way in which they are computed from the historical information – will tend to be bounded by certain range of variations, and that the features of data points which are contiguous in time will tend to present rather similar values. Now, if we consider again panels C and D, we can understand that, in the case of $y^{(1)}$, the training set will contain many observations with rather similar set of indicators and opposite values of the response variable. On the other hand, this phenomenon will be far less pronounced for $y^{(20)}$, since, after switching from one state to the other, the variable tends to persist in the new state for several successive observations, thus resulting in a well-posed classification problem. The impact of the training set differences can be observed in Fig. 2, where we report the results obtained for the prediction horizons $y^{(1)}$, $y^{(5)}$ and $y^{(20)}$.

In panel A, B and C we report – grouped by prediction horizon – the AUC of the ML models, obtained for the three stocks through 30 cross-validations, with the random train/test splits described in Section 3.3. As we can notice, the longer is the time horizon on which the stock outperformance is computed, the higher is the corresponding prediction quality, with the best results achieved for the response variable $y^{(20)}$. This different behaviour is consistent with our interpretation of the results in Fig. 1: the non-persistent pattern of the response variable has a major impact on the 1-day-ahead stock outperformance prediction, and the methods struggle to correctly classify this conflicting information. Moreover, many of the features used to predict the response

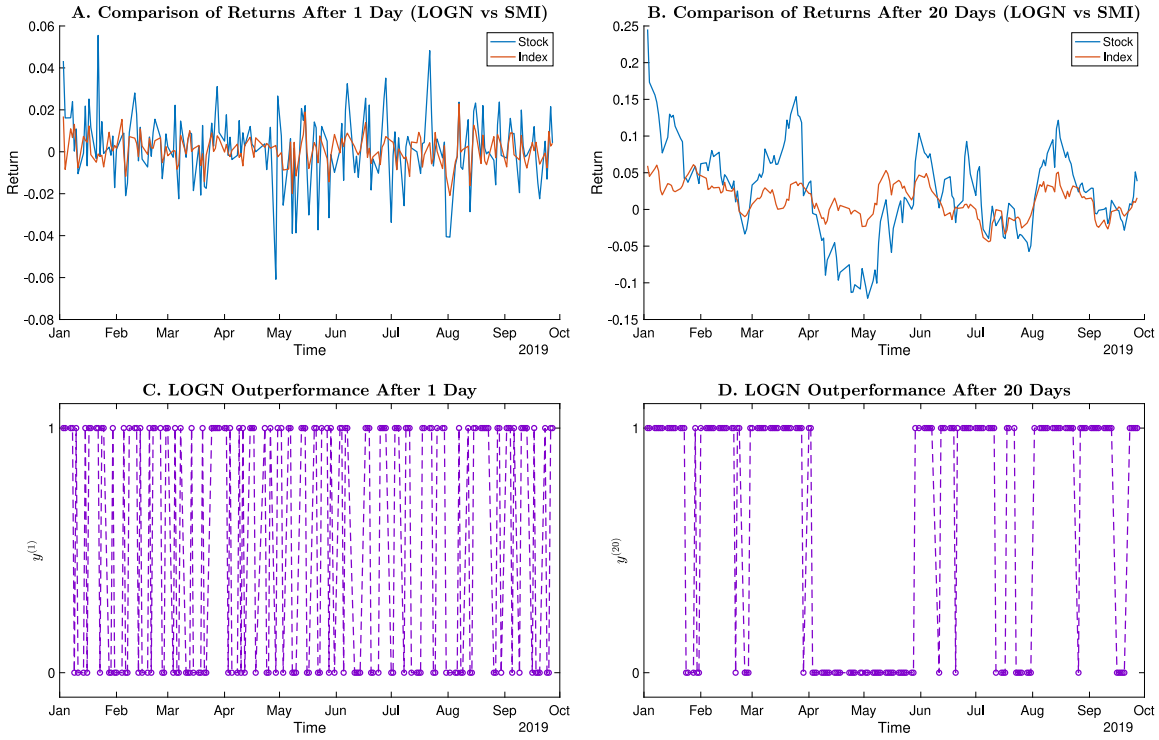

Fig. 1. Returns and outperformance (after 1 and 20 days, respectively) of LOGN.SW and of the SMI, in the period from the 1st of January to the 30th of September 2019.

variables consist in trend indicators based on the stock historical information and, thus, we can reasonably expect them to have a better predictive performance over a longer time horizon, characterised by a lower volatility. Regardless of these difficulties in the classification, we can notice that the eSPA algorithm consistently and substantially outperforms its competitors in all the scenarios considered, while achieving AUC values which are up to 60% higher than the ones of the other methods. Furthermore, the quality of eSPA outputs tends to change significantly less across the cross-validations performed, thus providing more reliable information to support the financial decision-making process. In panel D, we report the computational cost of the best performing models for the three different time horizons. As mentioned in Section 3.3, in order to allow a fair comparison of the different methods, the timing results have been obtained by disabling the parallelisation and by running all algorithms on a single computational thread. While the dimensionality of the data set is the same in all three cases, eventual large differences in the computational time (as in the case of Lasso GLM) are determined by a premature convergence of the algorithm to a suboptimal solution, as confirmed by the corresponding low AUC values. It is important to stress that for every train/test split, the methods have been run several times by considering different combinations of the hyperparameters described in Appendix B. However, the times plotted in panel D do not correspond to the total running times of the algorithms, but only to the time of a single instance – corresponding to the best performing model – for each cross-validation. In this way, we can provide a fair assessment of their computational cost, without including any bias stemming from the size of the grid search performed or the poor convergence of the worst-performing models. As we can observe, the eSPA algorithm markedly outperforms its competitors also in terms of computational cost, and requires roughly 3 order-of-magnitude less time than LSTM, the second-best model quality-wise. Furthermore, the extremely low computational cost of eSPA makes its potential application to financial decision-making and stock forecasting particularly interesting. Indeed, since the method can be re-trained very quickly, it results useful in all those situations in which an update on the fly with new information is beneficial (e.g., high-frequency trading).

Among the different small data problems solved, the 1-day-ahead prediction of stock outperformance over the market represents the most challenging scenario for all three stocks. Indeed, we can notice that the extremely different market capitalisations of NESN.SW and LOGN.SW bear apparently very little impact on the forecast quality. In order to assess if the prediction in this regime can be improved, in Fig. 3 we consider an extended simulation on NESW.SW.

Since, in Fig. 2, the benchmark machine learning algorithms were achieving a median AUC value of roughly 0.5, thus indicating their inability to distinguish whether the stock would outperform the market or not, in Fig. 3 we repeat the same analysis by conducting a comprehensive hyperparameter grid search and 30 cross-validations. With these new settings, we can notice a slight improvement in the performance of all algorithms, while eSPA still dominates its competitors, with a median AUC of 0.86 against the value of 0.63 achieved by LSTM, the second best performing model. Furthermore, we can observe also a reduction in the number of cases in which the methods are reciprocating (i.e., consistently predicting the wrong class). Reciprocation represents a fairly

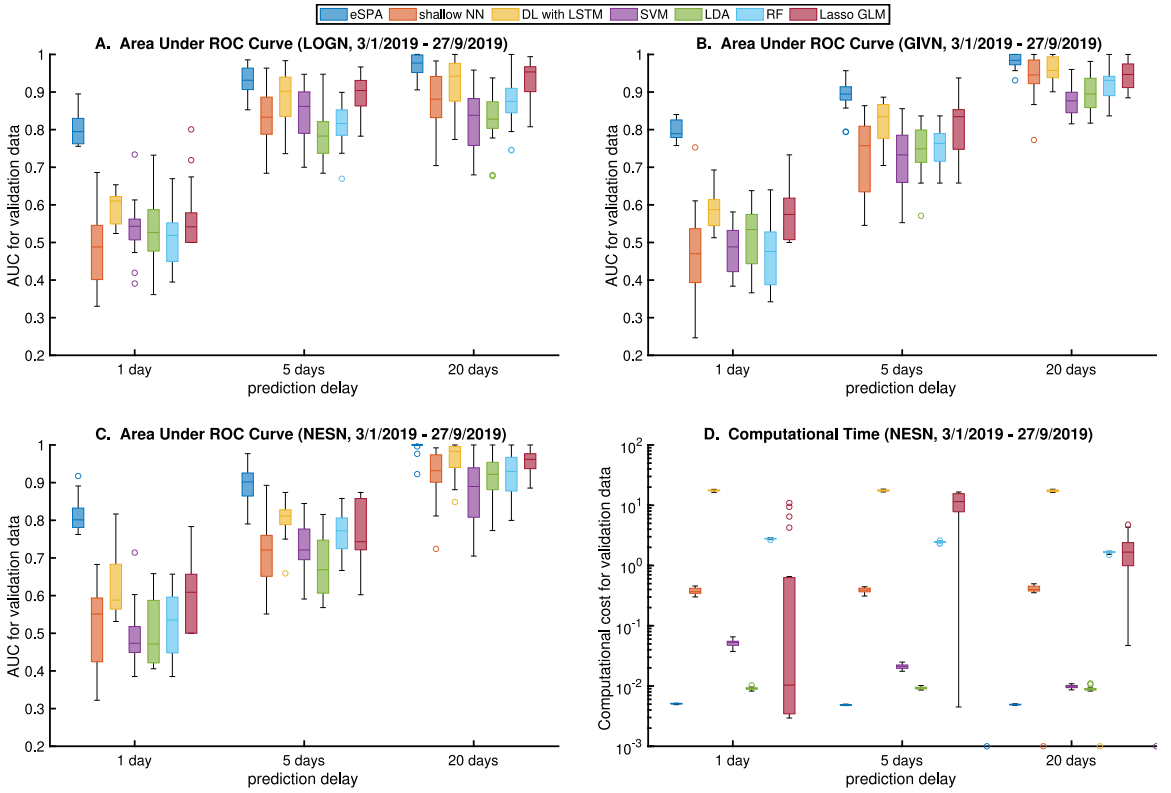

Fig. 2. Prediction of the outperformance of the three reference stocks over the market for three different time horizons in the period from the 1st of January to the 30th of September 2019. Panel A, B and C report the prediction quality, while panel D shows, as a reference, the computational cost for NESN.SW.

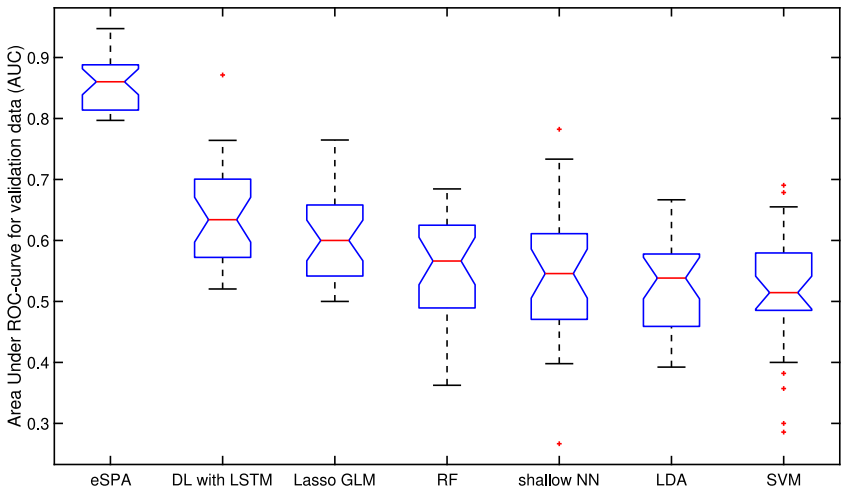

Fig. 3. Quality of the overperformance prediction after 1 day – $y^{(1)}$ – for NESN.SW in the period from the 1st of January to the 30th of September 2019. The box plots are constructed from 30 cross-validations over random train/test splits (80% training set, 20% test set).

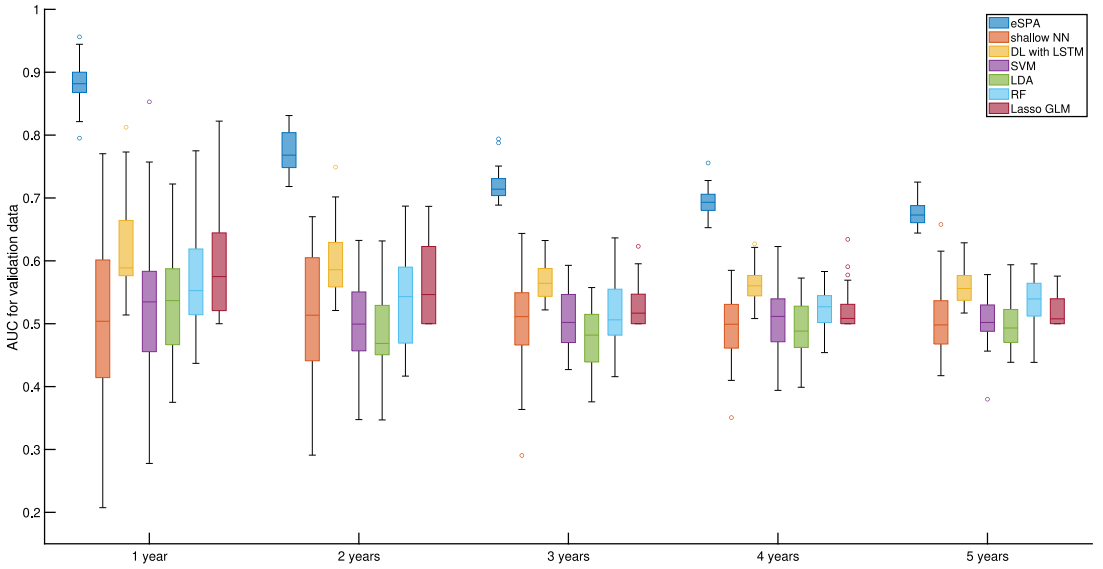

Fig. 4. Quality of the outperformance prediction after 1 day – $y^{(1)}$ – for NESN.SW over increasingly larger time frames (from 1 year to 5 years). The box plots are constructed from 30 cross-validations over random train/test splits (80% training set, 20% test set).

common issue in this kind of small data settings, and even a robust algorithm like eSPA has been sometimes observed to invert the labels in certain unlucky train/split. However the mathematical formulation of the latter method allows to easily and efficiently solve the rare occurrences of this problem.[4]

After assessing the minor impact of a careful hyperparameter tuning on the prediction results, one could still argue that including more points in the training set (thus going beyond the mere $T = 186$ observations in our reference data set) could significantly contribute to improving the performance of the ML algorithms, especially in the case of those methods – like LSTM – which are known to be data-hungry. If this intuition was correct, the classification problem could potentially be applied to a sufficiently large data set, thus eliminating all the issues entailed by the small data learning regime. Unfortunately, this turned out to be unfeasible, due to the presence of concept drift explained in Section 2.3 and confirmed by the experimental results reported in Fig. 4.

Here, we analyse the impact of an increase of the amount of data instances on the prediction of the response variable $y^{(1)}$ for NESN.SW. We start by considering all the data of year 2020 (which roughly have the same size of the reference data set, since this year has less observations) and systematically add the data of the previous years, thus considering the whole data set. The results confirm exactly what we were expecting: more information is not beneficial for the classification problem and the performance of the different methods tends to decrease, while getting closer to an AUC value of 0.5, which indicates the impossibility of discriminating whether the stock outperforms the market. Even eSPA, despite its robustness, shows a decrease in performance, by moving from a median AUC value of roughly 0.88 to 0.67, which – even if higher than the value of 0.55 achieved by the second best method (LSTM) – is still far away from the original performance. Thus, we can conclude that the addition of more data instances only contributes to complicating the classification problem, due to the inclusion, in the training set, of observations whose relevant classification features are different from the one of the other instances. In particular, since, in presence of concept drift, the importance of the explanatory variables (i.e., in our case, the technical indicators) in the classification task can change over time, we think that the decrease in performance of the ML algorithms can be traced back to the presence of several groups of time-dependent features, which have an impact on the explanatory variables only in some time intervals but not in others. Thus, including this conflicting information in the model training actually contributes to decrease the classification performance, since none of the methods can reliably identify this hidden trend in feature importance.

Before moving to the interpretation of the ML models and to the benefits of the entropic filtering operated by eSPA, we want to assess whether a small modification of the set of features used in the classification can have a positive impact on the 1-day-ahead prediction of stock outperformance. As specified in Section 2.2, the technical indicators considered are based only on the historical information of the different stocks and thus, being forward-looking, they do not introduce any hindsight bias in the problem solution. However, none of them actually considers explicitly the returns of either the stock or the market. In Fig. 5, we tackle again the prediction of $y^{(1)}$ for the three stocks, by considering two additional cases: on the left, we include among the features the returns at time t of both the stock and the market, while, on the right, we add also their lagged returns (at time $t - 5$ and $t - 10$, respectively).

[4] A systematic mislabelling in the binary classification case can be solved by simply flipping the rows of the stochastic matrix Λ, which contains the conditional probabilities linking the K discretisation boxes and the class label probabilities Π.

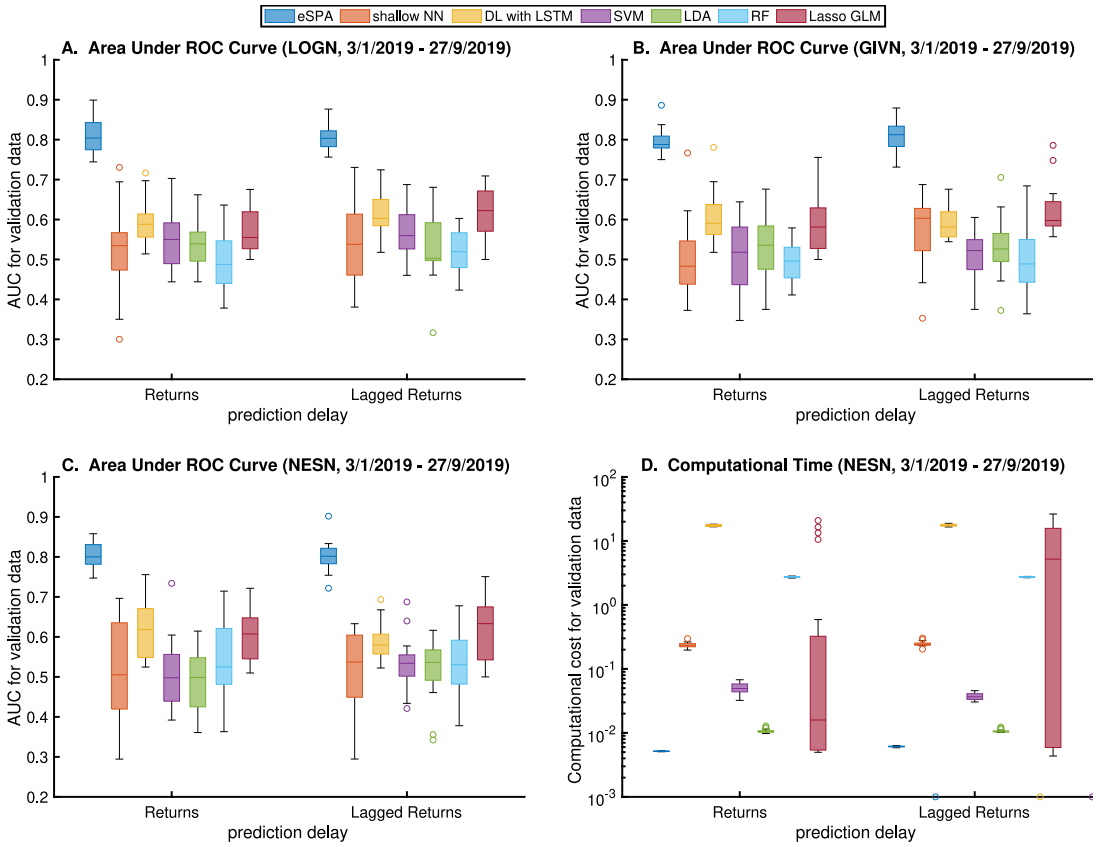

Fig. 5. Quality of the outperformance prediction after 1 day – $y^{(1)}$ – for the three stocks, with an extended predictors' matrix X considering the returns and the lagged returns of the stock and of the SMI in addition to the technical indicators. Panel D reports the computational cost.

The addition of returns and lagged returns among the explanatory variables may capture the possible momentum or reversal behaviour in stock returns (Avramov et al., 2006; Chan et al., 1996; Jegadeesh and Titman, 1993). After their inclusion, the data set contains a total of 65 prediction features in the first case and a total of 69 in the second case (see Appendix A.3 for the complete list). While the resulting small data problem is slightly more challenging to solve due to the increased dimensionality of the feature space, this new information could have had a potential positive impact on the classification results. However, as we can observe in Fig. 5, this is not the case and the results are equal to, or worse than, the ones observed before. Thus, we can conclude that even considering the returns or the lagged returns among the model features does not allow us to achieve a higher prediction quality.

4.2. Model interpretability and feature selection

In this section, we delve on the interpretability of the ML models and on their ability to select the relevant features for predicting the stock outperformance. Due to the challenges encountered in the 1-day-ahead prediction, in the following simulations we consider this specific scenario, and – since all stocks were yielding similar results – arbitrarily focus on NESN.SW for ease of presentation. The possibility of clearly interpreting the model output and identifying the features relevant for forecasting the stock outperformance is extremely beneficial in a financial decision-making scenario. Indeed, the ML models could be used, for example, to rule out some of the features that are never important in the classification process, thus providing the users with a reduced set of indicators to be used in their analysis. Only few of the methods considered allow us to directly check the weight that has been given to each explanatory feature in the final prediction (namely eSPA, RF and LDA), while, unfortunately, the others behave as *"black boxes"*, from which it is challenging to extract useful additional details to aid the decision-making process beyond the predicted value of the response variable. Nevertheless, in order to provide an insight on their interpretability, we consider a series of proxies consisting in partial dependence plots (PDP), individual conditional expectation plots (ICEP) and Shapley additive explanation (SHAP). In particular, PDP (Goldstein et al., 2015) represents the average impact of changes in the value of a feature of interest (i.e., one of the proposed technical indicators) on the classification label output of the model (i.e., stock outperformance presence or absence), while keeping all the other features values constant. ICEP (Goldstein et al., 2015) gives an information analogous to the one provided by PDP, but, while the latter considers the average effect, the former shows how the impact of the chosen feature changes depending on the different observations of the input data. Finally, SHAP (Lundberg and Lee, 2017) tries to assign to each feature a score representing

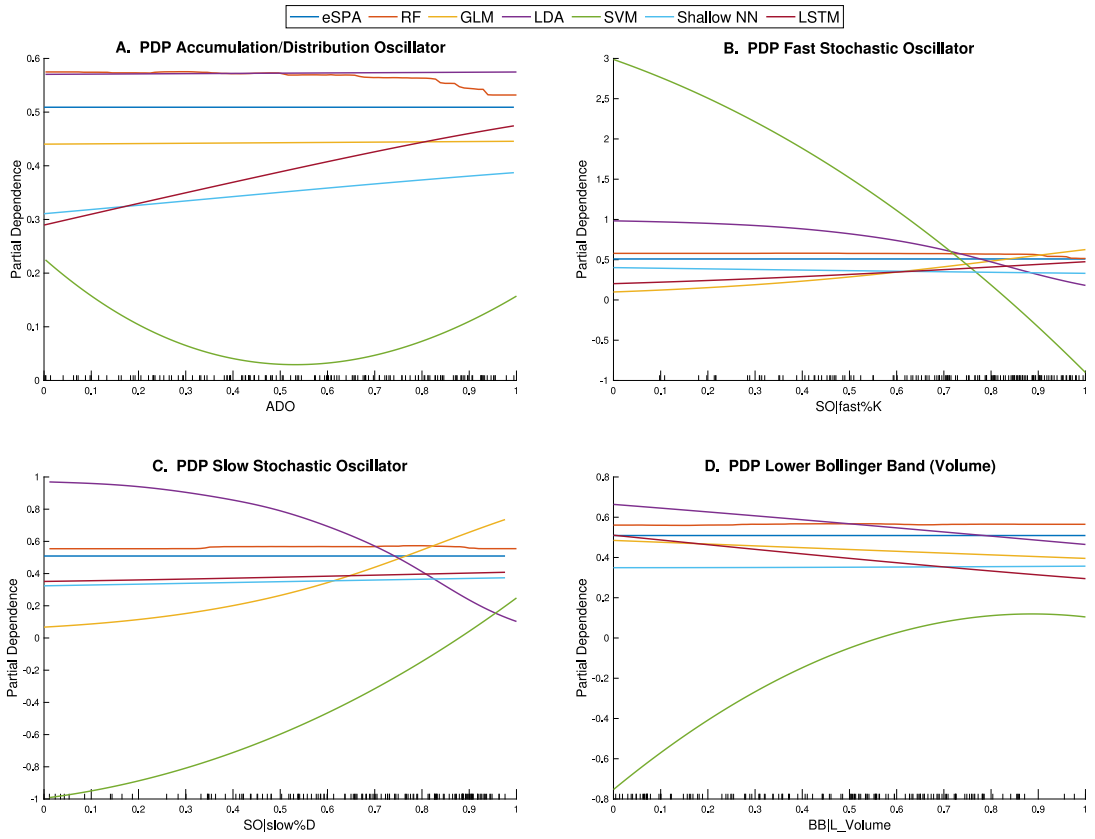

Fig. 6. Comparison of the partial dependence plots of the least important features (according to eSPA) for the 1-day-ahead prediction of NESN.SW outperformance over the SMI.

its importance in the prediction process, while estimating not only the direction of the impact of changes in the feature value, but also their magnitude. For specific details about the computation of these metrics and their implementation, we refer to Appendix C.1.

To compare the models, we ran 50 cross-validations with random train/test splits for the 1-day-ahead prediction of NESN.SW outperformance. In order to allow a fair comparison, we then selected the simulation which gave, on average, the best results for all models, thus dismissing the cases in which the ML algorithms achieved clear overfitting or did not converge to an adequate solution. In general, the results obtained in these experiments confirm the superiority of the eSPA algorithm also from an interpretability and feature selection standpoint. To illustrate this, we start by considering, in Fig. 6, some comparative PDP of the different models. Since including all the 63 technical indicators in this analysis was neither feasible nor more informative, we decided to focus on the 4 features that have been identified by the best-performing model (eSPA) as irrelevant for the prediction of stock outperformance. In particular, they are the accumulation/distribution oscillator, the fast stochastic oscillator, the slow stochastic oscillator and the lower Bollinger band on the volume.

The results in Fig. 6 can be interpreted as follows. First of all, a straight horizontal line indicates that the corresponding feature bears no impact on the classification process and should be dismissed by our financial decision-making model. However, we can notice that not all ML methods agree on the interpretation of the role played by the technical indicators: RF, shallow NN, LSTM and SVM indicate that a variation in the value of one of these 4 features would actually have an impact, and potentially change the final prediction from presence to absence of stock outperformance. In order to better investigate their behaviour and to compare it with the one of a method which deems the same technical indicator not important, in Fig. 7 we consider the ICEP of the least important feature according to eSPA – i.e., the accumulation/distribution oscillator – and assess how the stock outperformance prediction changes according to different observations from the input data.

As explained before, the lines plotted in the PDP represent an average of the values obtained by letting the variable of interest change – while the others are kept fixed – for each observation provided to the model. The closer the ICEP lines are to the average, the less variability we expect to be in the impact of the feature on the final prediction when considering a different data set, and – consequently – the best the model performance. In general, from a preliminary observation of the results, we can notice that there tends to be a lot of dispersion around the average value, thus indicating that both the impact direction and magnitude of the accumulation/distribution oscillator on stock outperformance prediction changes significantly depending on the data considered. This is particularly true for LSTM, which, as we can observe in panel A of Fig. 6, identified a high partial dependence between this

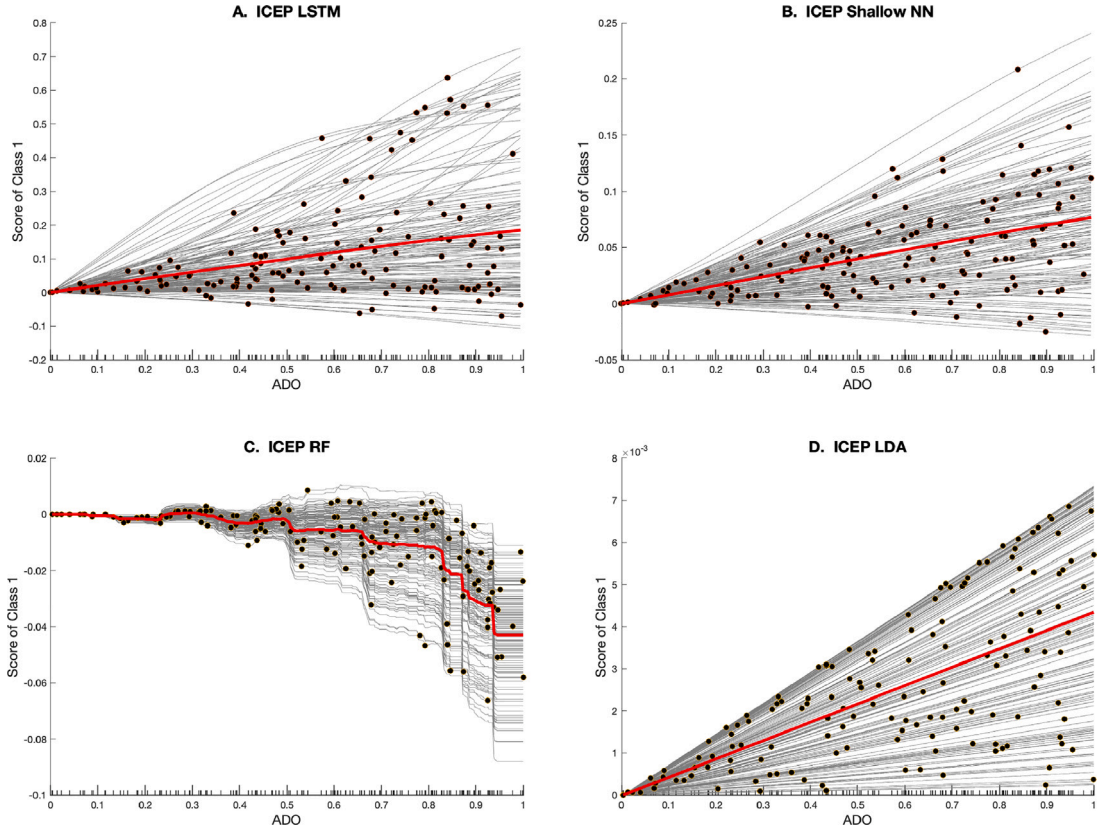

Fig. 7. Comparison of the individual conditional expectation plots relative to the accumulation/distribution oscillator for LSTM, shallow NN, RF and LDA, in case of the 1-day-ahead prediction of NESN.SW outperformance over the SMI.

variable and the final prediction. On the other hand, the apparent dispersion of the values of the ICEP of LDA in panel D of Fig. 7, actually results in minimal changes, as it can be observed by considering the scale of the variations of class 1 assignment (in the order of 10^{-3}). This result is consistent with what we reported in Fig. 6 (panel A) for this method: a straight line indicates that the accumulation/distribution oscillator bears no impact on the stock outperformance prediction, regardless of the data given as input to the model. Finally, we can highlight how the different methods do not always agree with the PDP of eSPA: as an example, in Fig. 6 (panel A), both LDA and GLM agree with our reference model and consider the accumulation/distribution oscillator irrelevant for the stock outperformance prediction, but – on the contrary – in panel B and C, they identify as relevant the fast and slow stochastics. While we cannot expect all ML models to agree on the same set of relevant technical indicators, these discrepancies still partially explain the different prediction qualities they are able to achieve. To provide further evidence towards our comparative assessment of model interpretability, in Fig. 8 we consider the SHAP of eSPA, RF and LDA and compare them against the feature importance, which can be directly computed for these methods. Indeed, among the selected ML models, they are the only three which provide a direct and clear interpretation of the weights given to each technical indicator in the stock outperformance prediction.

As we can observe, SHAP provides a good proxy on the impact of technical indicators on the classification task, and the values obtained are in accordance with the weights assigned by the methods to the different features. However, it is also important to stress the main difference between eSPA and its competitors: the former method operates an entropic filtering of the features, clearly eliminating all those which are not relevant to the problem solution. This can be noticed in both panels A and B of Fig. 8, where only a few technical indicators (in this specific cross-validation split, the upper and middle Bollinger bands on all historical data) are considered relevant for the prediction task. For a more detailed explanation of the entropic filtering and its application to financial decision-making, we refer to Section 4.3. On the other hand, both RF and LDA tend to base their prediction on significantly more features, while assigning marginal weights to the ones identified by eSPA as key players. This different behaviour is potentially at the base of the lower classification performance achieved by these methods. For the sake of completeness, in Fig. 9 we report the SHAP for shallow NN, LSTM, SVM and Lasso GLM.

First of all, it is important to stress that none of these ML methods allows a direct computation of the feature weights, and so we are forced to consider only the SHAP to assess the importance of the technical indicators in the stock outperformance prediction. Also in this case, the ML methods cannot discriminate between the relevant and irrelevant features, but consider all of them to be of some importance in the classification task. As already mentioned, this is definitely not a desirable behaviour if we want to employ a ML tool in real financial decision-making problems, since the inability to determine the main drivers behind the stock

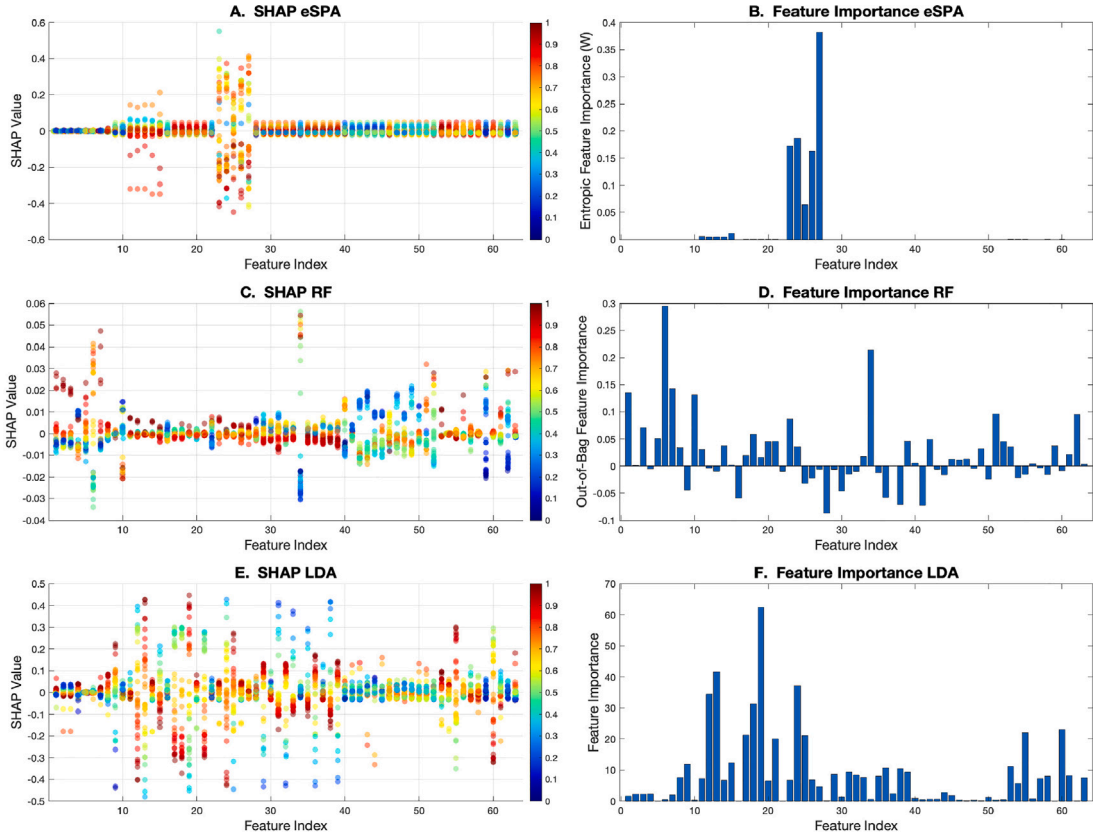

Fig. 8. Shapley additive explanation and feature importance of eSPA, RF and LDA classification models for the 1-day-ahead prediction of NESN.SW outperformance over the SMI.

outperformance does not allow us to filter the relevant information from the one that bears no impact on our prediction. And this problem becomes particularly important in all those situations in which the amount of possible explanatory variables is potentially high, while the time allowed to train the model and output a prediction is short (e.g., high-frequency trading). For these reasons, in the next section we focus only on eSPA and try to understand how the model which has been shown to provide the best performance can actually be tuned even further to develop robust prediction models.

4.3. Financial decision-making through entropic approximate learning

In this section, we want to further focus on the interpretation of the results from the perspective of eSPA, by showing – in particular – how the entropic weights assigned by the model to the various technical indicators provide further insights into the financial decision-making process. This feature of the eSPA algorithm is particularly important, since even those algorithms which provide some information on the importance of the different features in the classification process – like, e.g., RF – still lack a systematic assessment of their impact on the classifier entropy. First of all, we show how the entries of the entropic feature selection vector W – produced as an output by the eSPA algorithm (see Appendix B.1) – allow us to discern the subset of features which are more relevant for the stock outperformance prediction. We then discuss how we can select, across different cross-validations, the most appropriate model, by combining quality metrics with the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC).

4.3.1. Selection of the relevant subset of features through entropic filtering

While achieving a good performance on the validation set is important and desirable, a true insight into the financial decision-making process can only be obtained by identifying and analysing the features included in the output model. However, understanding which technical indicators are actually relevant for the prediction of the stock outperformance over the market and which can be neglected is not a trivial task, which is further complicated by the small number of observations in the training set. For example, even a state-of-the-art method like t-distributed stochastic neighbour embedding (t-SNE) – a nonlinear dimensionality reduction technique aimed at embedding a high-dimensional space into a low-dimensional space (Van der Maaten and Hinton, 2008) – fails completely at identifying the relevant dimensions in the financial data set considered. In Fig. 10, we report the results obtained by

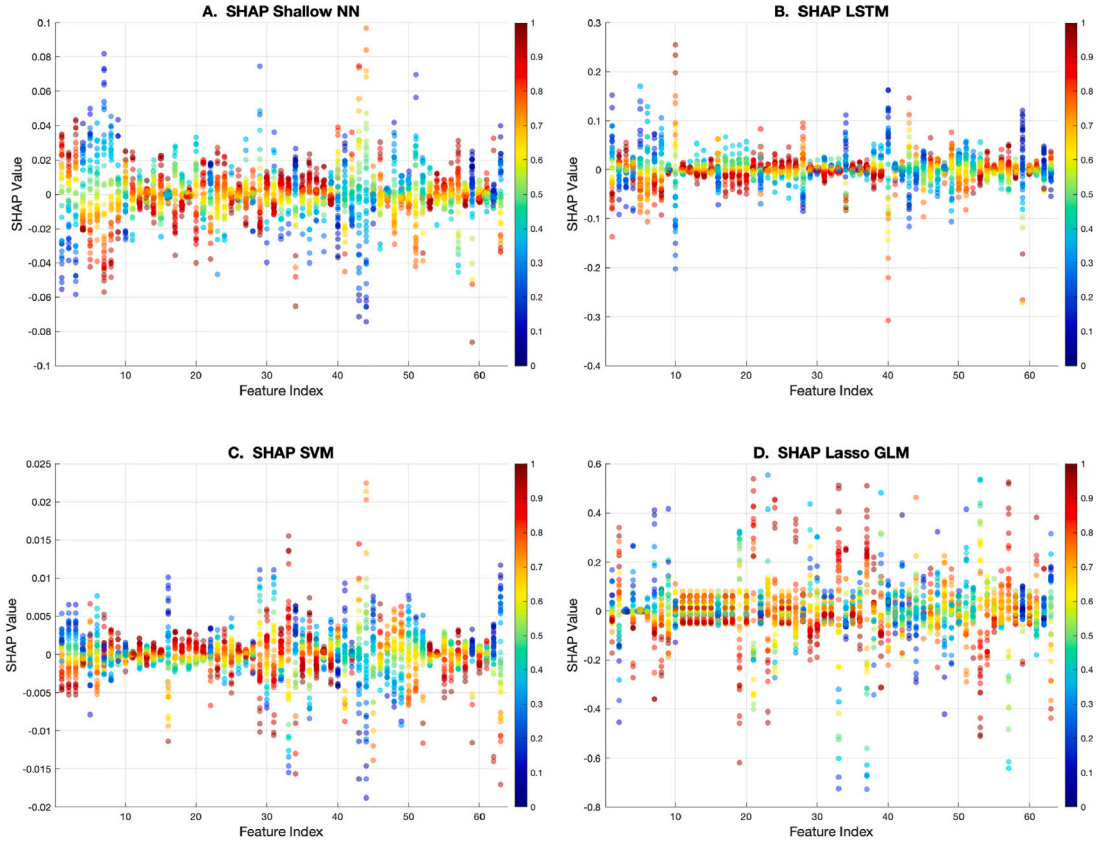

Fig. 9. Shapley additive explanation of shallow NN, LSTM, SVM and Lasso GLM classification models for the 1-day-ahead prediction of NESN.SW outperformance over the SMI.

embedding the original data space for the prediction of variable $y^{(5)}$ (with 63 dimensions, corresponding to the technical indicators) into a 2-dimensional space spanned by the two most relevant features. Panel A and B consider the data relative to NESN.SW in the reference data set and in the complete data set, while panel C and D consider the two same time horizons in the case of LOGN.SW. The application of t-SNE should allow us to distinguish the data instances in which the stock outperforms the market from the others, by plotting the former close to each other and away from the rest of the data instances. However, as we can observe in Fig. 10, this does not happen: the two groups are completely overlapping and indistinguishable from each other. And including more data (as in panel B and D) is not beneficial either, since we obtain the same results observed for the reference data set.

We now want to comment and interpret the models produced by the eSPA algorithm, by providing a direct visualisation of the entries of vector W, which performs an entropy-based selection of the relevant classification features. For more details about how the entropic filtering is performed, we refer to Appendix C.2. Here we just mention that the maximum entropy limit of vector W is reached when every component has a value of $1/D$, with $D = 63$ features in this case (Gerber et al., 2020). This distribution of W leads to a scenario in which the uncertainty is maximised, since the algorithm would be unable to distinguish which features have an active role on the stock outperformance prediction and which are just noise, irrelevant for the classification process. In order to determine the weights assigned to every feature by the eSPA algorithm, in Fig. 11 we show the results of 50 cross-validations performed over random train/test splits (80% training set, 20% validation set) of the reference data set for the 5-day-ahead prediction of NESN.SW outperformance over the SMI. In this preliminary example, we focus our attention on the response variable $y^{(5)}$, since the reduced volatility of the returns over 5 days contributes to the stability of the resulting models. From the heat map, we exclude all those values which are below the threshold represented by the maximum entropy weight (i.e., in this case, $w_{\text{maxEnt}} = 0.0159$).

As we can observe, there is a certain degree of consistency in the feature selection process operated by eSPA, and only a few models consider relevant a subset of features which is discarded by the vast majority of the cross-validations. In particular, model 6 is the only one which significantly deviates from the rest and bases its classification solely on 10 of the available features, while assigning to some of them relatively high weights (close to 0.2) when compared with the other candidates. However, there is also consensus across all models that several features are not relevant for predicting whether the stock outperforms the market: among these we can find the slow and fast stochastic oscillator, the accumulation/distribution oscillator, the Chaikin oscillator, the William %R, the acceleration, the momentum, the volume rate of change and the Chaikin volatility. On the other hand, technical indicators like the positive and negative volume index, the upper, lower and middle Bollinger bands, the highest high, the

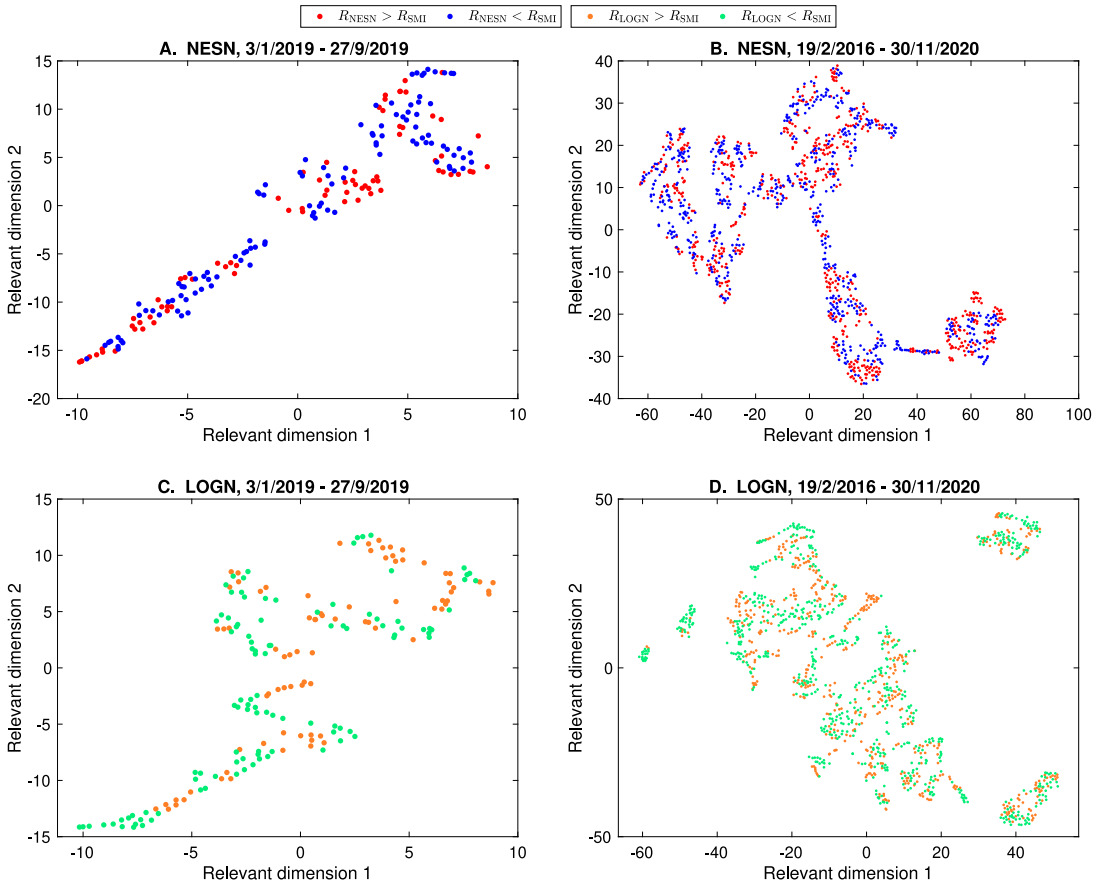

Fig. 10. Dimensionality reduction (2D) using t-SNE of the data space for the prediction of $y^{(5)}$, in case of NESN.SW (highest market cap) and LOGN.SW (lowest market cap). Panel A and C are computed on the reference data set, while panel B and D on the whole data set.

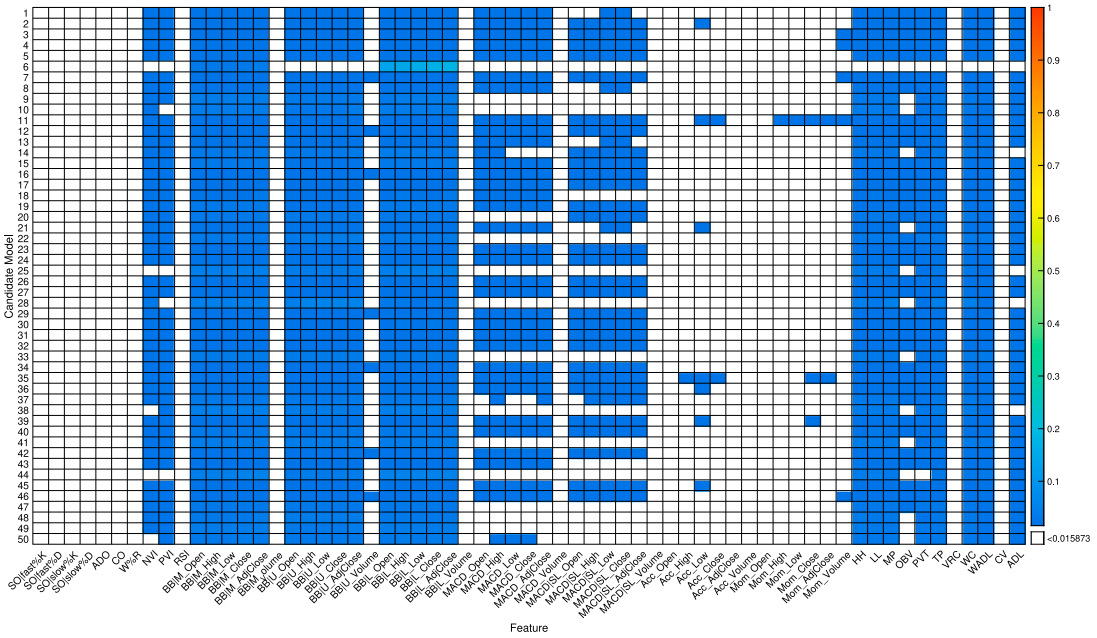

Fig. 11. Feature importance of the different models proposed by eSPA over 50 independent cross-validations for the prediction of NESN.SW outperformance after 5 days on the reference data set. The cut-off threshold corresponds to the weights of W under maximum entropy.

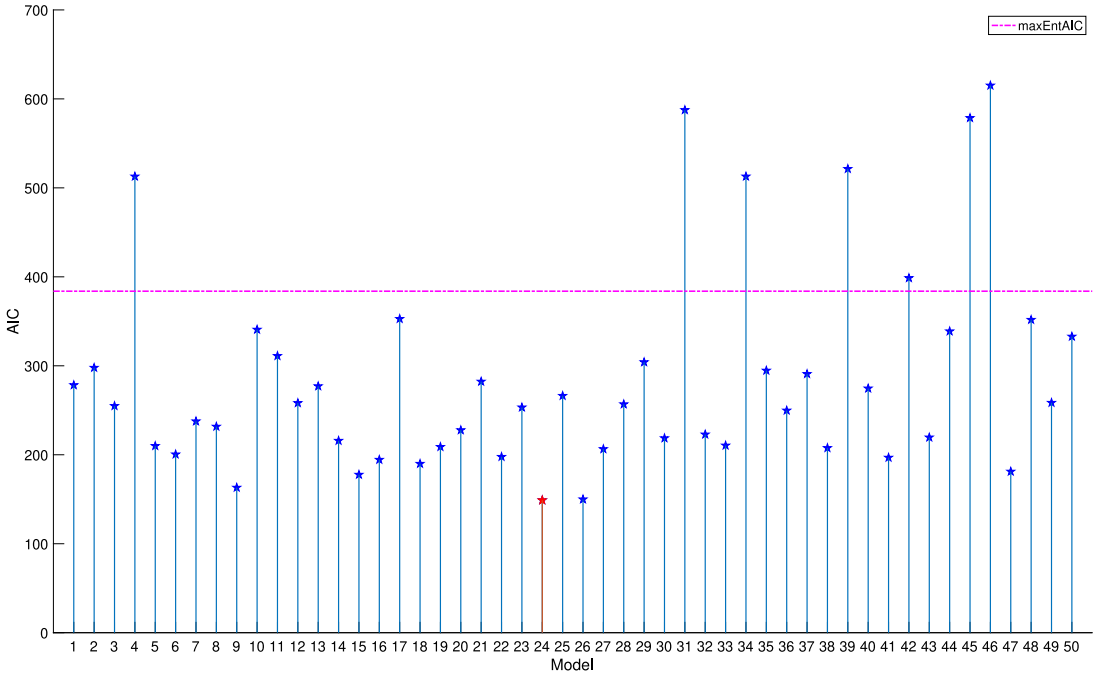

Fig. 12. Akaike Information Criterion (AIC) of the 50 models proposed by eSPA for the prediction of NESN.SW outperformance after 5 days on the reference data set. The horizontal threshold represents the AIC value for this problem in case of maximum entropy. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

lowest low, the median price, the price and volume trend, the weighted close, the Williams accumulation/distribution line and the accumulation/distribution line are consistently important across all models for the prediction of the stock outperformance over the market. For the complete list of the technical indicators used as features, we refer to Appendix A. It is important to stress that the features identified as most relevant for the classification process are strictly connected to the reference data set and that considering time intervals which are not sufficiently close to it could result in a different ranking of their relative importance. Actually, this behaviour is perfectly consistent with the presence of the concept drift phenomenon highlighted in Section 4.1, and leaves as an open problem the determination of several time-dependent groups of features, whose importance depends on the unobserved latent processes regulating the stock outperformance in that particular time frame. Regardless of these limitations, we are interested in deriving a set of decision rules that allows us to select, among the different candidates proposed by the eSPA algorithm, the model which better combines flexibility, parsimony and prediction performance.

4.3.2. Best eSPA model selection via Akaike and Bayesian information criteria

The selection of the best model among different candidates is not a trivial task and several approaches have been developed to tackle this issue (Claeskens and Hjort, 2008). Certainly, we can reasonably expect that the best model is characterised by a high prediction accuracy (ACC) or, even better, by a high AUC value, since the latter would indicate that it is able to correctly predict the presence or absence of stock outperformance over the market. However, deciding only on the basis of this factor could entail the risk of choosing a model which overfits a lucky training/test split and cannot really generalise beyond the original data set. For this reason, we propose to rely on the Akaike Information Criterion (AIC) and on the Bayesian Information Criterion (BIC) to select the most appropriate model (Claeskens and Hjort, 2008). In Appendix C.3 we show how to compute the latter two information criteria for a general eSPA model.

The AIC will tend to penalise all those models in which the probability of obtaining the true label is low, with the maximum possible penalisation achieved when an observation is misclassified with absolute certainty (i.e., when we have a probability equal to 1 of obtaining the opposite label, given the current model parameters). In Fig. 12, we provide an application of this decision criterion, by reporting the AIC of the different candidate models considered in Fig. 11.

In order to allow a preliminary evaluation of the overall model quality, we plot also the value of the AIC in case of maximum entropy, i.e., when the resulting model contains 63 parameters (since all features are relevant for classification) and the estimated label probabilities for the two classes are all equal to 0.5, thus maximising the uncertainty of the system. Of course, we aim at finding a model which can outperform this very naive and not informative probability allocation, and we automatically discard all those model candidates who are above this threshold. In Fig. 12, we can notice that only a few models have an AIC value which is considerably higher than the threshold, and this depends on the fact that one or more of the data points in the training or validation set were sharply misclassified. The latter misclassification, however, does not necessarily imply that these models achieve

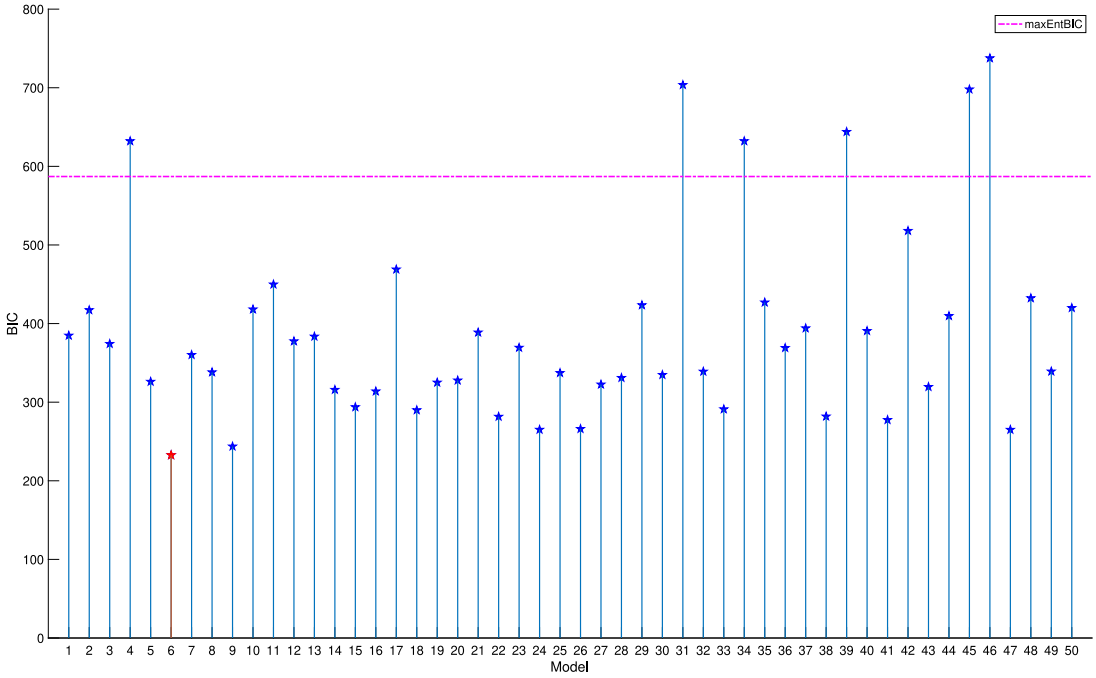

Fig. 13. Bayesian Information Criterion (BIC) of the 50 models proposed by eSPA for the prediction of NESN.SW outperformance after 5 days on the reference data set. The horizontal threshold represents the BIC value for this problem in case of maximum entropy. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

a bad performance in terms of AUC or ACC: indeed, all of them present at least an AUC of 0.85 and an ACC of 0.8, but the fact that they are completely unable to correctly classify some of the observations is heavily penalised in the computation of the AIC. As far as concerns the other models, we can notice that they present values of the AIC significantly below the reference threshold and that the minimum is achieved by the candidate number 24, which is highlighted in red in the figure. If we now consider again Fig. 11 and compare the relevant features of the best candidate model with those of the other models presenting a low value of the AIC (like, e.g., models 26, 9 and 47), we can notice that there is agreement on the vast majority of them, except for the moving average convergence/divergence and for the on-balance volume. These discrepancies between the candidate models are, however, rather marginal and the final choice is also dictated by how much we want to penalise the inclusion of additional parameters with respect to the maximisation of the likelihood. For the sake of comparison, in Fig. 13 we report the results obtained for the same models when considering the BIC.

The BIC actively penalises any increase in the number of parameters more than the AIC, thus actively selecting those models with as few parameters as possible. In Fig. 13 we include, as additional benchmark, the value of the BIC in case of maximum entropy, which can be computed similarly to the analogous AIC value proposed above. By observing the results, we can notice that – according to this new perspective – the best candidate model is number 6 (indicated in red in the plot), followed closely by candidate 9 and, a bit further away, by alternatives 24, 26 and 47. The latter models are actually the same identified as best candidates according to the AIC, with the addition of model 6, which presents the lowest number of relevant features across all cross-validations (only 10 out of 63) and is the only one giving a value close to 0.2 to 5 of its selected features. Since AIC and BIC are in disagreement in the identification of the best model, it is appropriate to investigate the model selection problem from another perspective, in order to make the most appropriate decision according to the available information. In Fig. 14, we can find all 50 candidate models sorted according to their AUC (panel A) and their ACC (panel B). In order to visualise their position in the overall ranking, we also indicate the optimal models according to AIC and BIC.

As we can observe, there is a huge discrepancy between the two best candidates, with model 24 achieving considerably higher values than model 6 in terms of both AUC and accuracy. By looking at the model selection problem from this perspective, we can then conclude that model 6 is the odd one out and probably an artefact induced by a particular train/test split, with seriously hampered predictive power beyond this very specific setting. Thus, our choice should fall – among the other best candidates – on model 24, since it is the one with the highest AUC and accuracy, at the price of a small increase in the number of parameters (e.g., it shows an increase of 15% in terms of AUC with respect to model 9, at the cost of 11 additional parameters). Regardless of these differences, the number and type of features selected serve only at aiding the financial decision-making process and have no real repercussion on the computational time of the chosen model, which is already extremely small. It is also interesting to briefly consider the candidates that have been completely dismissed by AIC and BIC, due to their high value in these statistics. Among them we can find, e.g., model 45, which is very close to the chosen candidate (model 24) in terms of AUC, and actually outperforms it

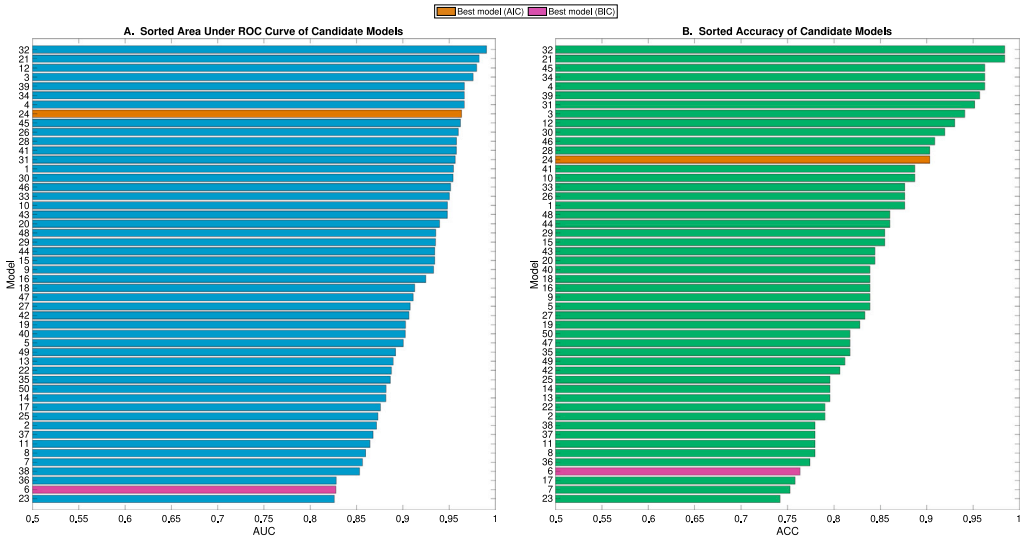

Fig. 14. Area under the ROC Curve (AUC) and accuracy of the 50 models proposed by eSPA for the prediction of NESN.SW outperformance after 5 days on the reference data set. The best candidates according to AIC and BIC are indicated in orange and pink, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

in terms of accuracy on the whole data set. This apparent superiority allows us to highlight, once more, how important it is to rely not only on the usual quality metrics, but also on information criteria like the AIC to select the most appropriate model. Despite the extremely good performance displayed in the reference data set, model 45 is completely unable to correctly classify some data instances.[5]

The prediction of the stock outperformance after 5 days considered above represents a well-behaved example, where the selected features are consistent across different cross-validations and in which the identification of the best model can be done quite straightforwardly. As a counter example, we consider the entropic feature selection for the prediction of LOGN.SW outperformance after 1 day, whose results over 50 cross-validations are reported in Fig. 15.

The models are sorted according to the decreasing value of their AIC, which is reported in the upper part of the figure. The best model is indicated with the number 1, while the worst model is number 50. Due to the sharp misclassification of many of the data instances, the majority of the models is beyond the maximum entropy threshold for the AIC, thus showing that the 1-day-ahead stock outperformance prediction is extremely challenging. Indeed, as we can observe from the heat map, in this case there is much less agreement between the cross-validations about which are the important features. In particular, there are several candidate models that rely on a single feature for solving the classification problem. As an additional control, we considered, together with the technical indicators, also the returns and lagged returns introduced in Section 4.1. The results reported in Fig. 15 are consistent with our previous findings: neither the returns nor the lagged returns are relevant for the prediction of the response variable $y^{(1)}$. Indeed, the weights assigned to these features by the eSPA algorithm are always significantly below the maximum entropy threshold – except in a few instances – and we can thus safely exclude these variables from the model. In order to proceed with the selection of the best model across the 50 candidates, we can compare the ranking performed by the AIC with the one done according to the BIC, which is reported in Fig. 16.

In general, we can observe that also the BIC presents considerably higher values than the one achieved in the prediction of $y^{(5)}$ for NESN.SW, due to the additional challenges imposed by the short term prediction. Once again, many of the proposed models are beyond the maximum entropy threshold, due to their sharp misclassification of several data instances. We can notice that, in this example, BIC agrees on the fact that the best candidate is model 1, which is based on a single feature. Thus, according to the latter model, the only relevant dimension according to which the classification problem should be solved is the price and volume trend, while all other features should be dismissed without any major impact on the classification result. Certainly, the fact of having just one parameter is highly rewarded by AIC and, especially, by BIC, but it also potentially entails the risk of having a less robust model across different data splits. In order to make the best choice, in Fig. 17 we consider, for all candidates models, the corresponding AUC and the ACC. The best model according to these performance metrics is indicated in orange in both panels. As we can observe, model 1 achieves intermediate results in terms of both AUC and accuracy, but there are also several models that perform significantly

[5] This is caused by the fact that its matrix Λ presents a sharp 0 or 1 probability for some of the discretisation boxes. While certainty in the classification outcome can be extremely valuable, it also entails less flexibility and a risk of hampering the decision-making process.

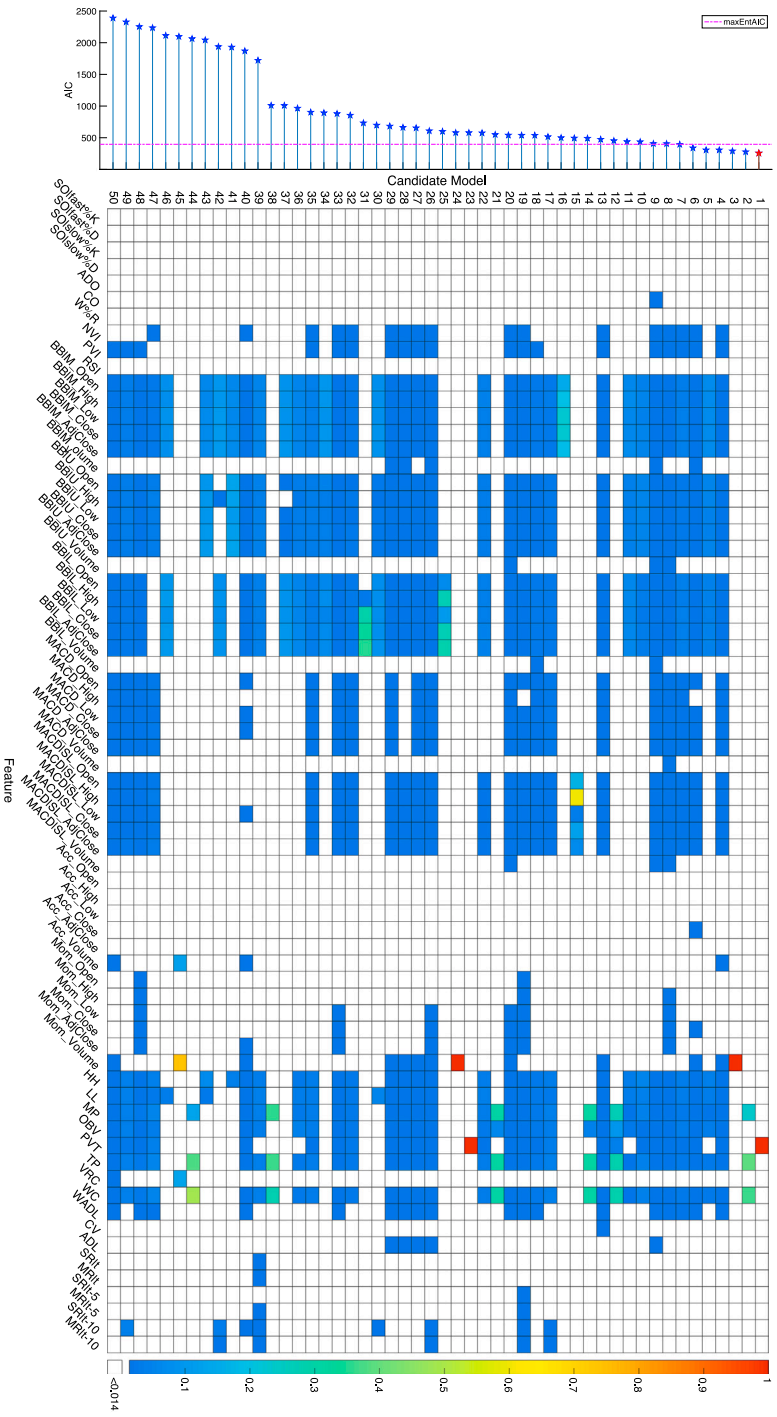

Fig. 15. Feature importance of the different models proposed by eSPA over 50 independent cross-validations for the prediction of LOGN.SW outperformance after 1 day on the reference data set. The cut-off threshold corresponds to the weights of W under maximum entropy. The models are sorted from worst to best according to their AIC, reported above.

better than it. We can then consider the other best candidates according to AIC and BIC- – like models 2, 3, 4, 5 and 6 – to see if we are able to find a better model. Among these, our choice falls on model 2: indeed, the latter has a significantly better AUC and ACC than model 1, while minimising, at the same time, the number of parameters. The three relevant features for the stock

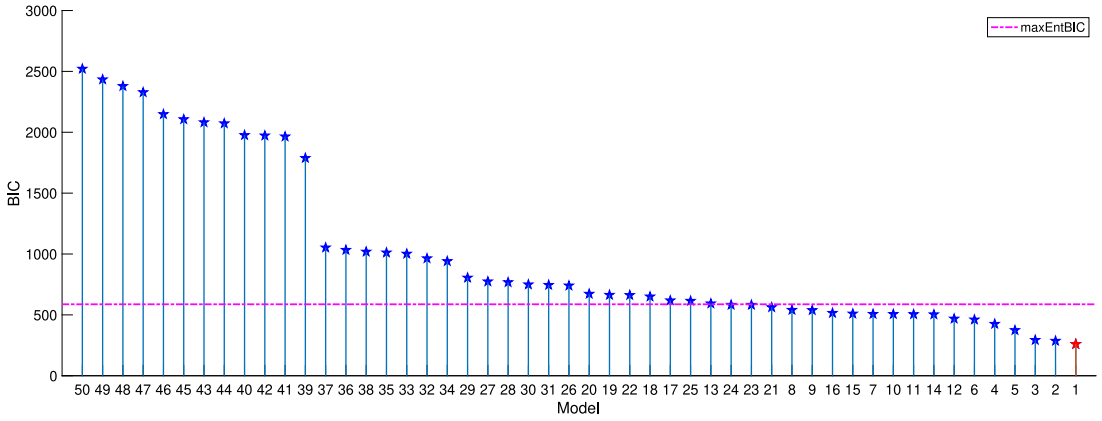

Fig. 16. Bayesian Information Criterion (BIC) of the 50 models proposed by eSPA for the prediction of LOGN.SW outperformance after 1 day on the reference data set. The horizontal threshold represents the BIC value for this problem in case of maximum entropy.

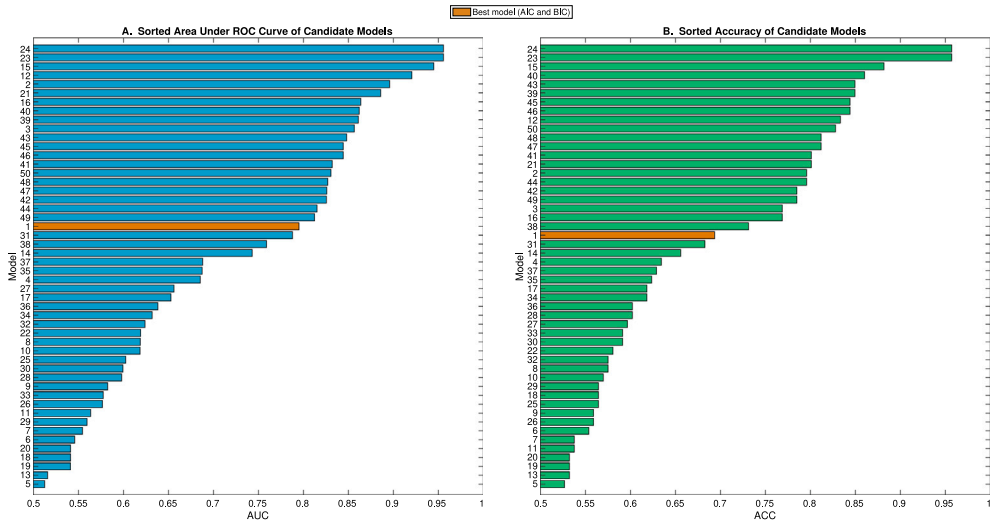

Fig. 17. Area under the ROC Curve (AUC) and accuracy of the 50 models proposed by eSPA for the 1-day-ahead prediction of LOGN.SW outperformance on the reference data set. The best candidate according to both AIC and BIC is indicated in orange. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

outperformance prediction are, according to model 2, the median price, the typical price and the weighted close. The exact same set of features, albeit with different relative weights, is used by other candidates – like models 12, 14, 21, 38, 44 – thus supporting even further their inclusion in the final classification model.

In the previous two examples, we have shown some statistically derived heuristics to select the best model among the different candidates, through the combination of quality metrics and information criteria. For example, by carefully considering the models in terms of AIC and BIC, we can rule out those candidates that – despite the good results achieved in terms of predictive performance – have a null likelihood for certain data points. Furthermore, identifying the important features for classification and assigning a precise weight to each one of them gives insights into the model structure and fosters even further the financial decision-making process with clear and accessible information.

5. Conclusions and limitations

In this paper, we discussed the potential application of state-of-the-art ML algorithms to the solution of financial decision-making problems, by considering the prediction of the outperformance of three stocks belonging to the SMI over the market index for three different time horizons (1-day-, 5-day- and 20-day-ahead). The prediction of stock outperformance has been modelled as a binary classification problem and we used, as explanatory features, a set of technical indicators based on the historical information of each stock. Among the different time horizons considered for the response variables, the 1-day-ahead stock outperformance prediction proved to be the most challenging for all the ML methods considered, due to the higher volatility of daily returns with respect to

those computed over longer time horizons. This resulted, in particular, in a continuous switching of the class label between presence and absence of stock outperformance, while the values of the predictive technical indicators were relatively stable. With this paper we have achieved – from a financial standpoint – several important results, which we can summarise as follows. First of all, we were able to contribute to the ongoing discussion in the literature about the actual relevance of technical indicators in stock forecasting: our results suggest that these variables provide valuable information and allow a good prediction of the presence or absence of stock outperformance. In particular, combining the technical indicators with methods that are capable of actively filtering subset of them to predict the response variable (like eSPA) allows to achieve a higher prediction accuracy also in the 1-day-ahead-prediction scenario. As a second point, we investigated the impact of the amount of training data on the performance of the different ML models and explained why this financial forecasting problem, as many others from finance and economics, actually pertains to the small data regime. In particular, this is due to the fact that, on the one hand, we tend to have more explanatory variables (e.g., technical indicators) than observations on response variables (e.g., presence/absence of stock outperformance over time) and, on the other hand, an increase in the amount of training data – by including, e.g., longer time series – tends to be detrimental on the classification performance due to the presence of concept drift. In this regard, we showed that including several years of data actually decreased the predictive ability of the ML methods, due to the presence of clusters of time-dependent groups of features that, in different time intervals, act as the main drivers of the stock outperformance process.

A thorough comparison of the ML models has seen the entropy-based eSPA algorithm achieve the highest performance among all methods considered, both in terms of prediction accuracy and computational time. In particular, the low computational cost results particularly interesting for some types of financial applications, in which it could prove useful to quickly retrain the model on the latest available data (e.g., high-frequency trading). Furthermore, we discussed the interpretability of the different methods and their ability to discriminate which features are relevant for the stock outperformance prediction and which can be safely excluded from the final model. Also in this case, the eSPA algorithm resulted easily interpretable and the only one capable of filtering the technical indicators information to improve the prediction quality. The latter feature is particularly important from a financial perspective, since it gives us the opportunity to provide the model with several different input explanatory variables and let it automatically select the most relevant ones for the solution of the problem under consideration. This allows us to avoid introducing any bias when pre-selecting the relevant features (either manually or through a dimensionality-reduction approach), and also to test whether the method can discover "hidden patterns" in the data that cannot be easily identified from a human perspective. In the last part of the paper, we provided some statistically derived heuristics, in order to select the model which combines a high predictive accuracy and a reasonable number of parameters. In particular, by pairing quality metrics with model selection techniques, like the Akaike and Bayesian Information Criteria, we have shown how to identify the best model, while eliminating apparently well-performing candidates that lack generalisation beyond the training data.

Despite the relevant results achieved in this contribution, this study has also several limitations that we plan to address in our future research. First of all, due to the challenges imposed by the small data learning regime, in the current analysis we focused only on 63 technical indicators as explanatory variables. However, given the good performance shown by the eSPA algorithm and its ability to filter the input information, we would like to consider several more classification features in our analysis, and check if the high performance displayed in our experiments holds also in the new simulation setting. Another possible extension consists in performing a comparative analysis of different markets and check whether the technical indicators highlighted as important tend to be the same or not. Or, maybe, we could change completely the time horizon of our problem and expand the feature space by including other regressors beyond the technical indicators (like, e.g., macroeconomic variables or proxies from fundamental analysis). We finally plan to address more closely the problem of concept drift, since its solution would have major implications on a more fruitful application of ML algorithms to financial decision-making. Indeed, understanding the process according to which the groups of relevant features change in time would allow us, on the one hand, to increase the size of the training data set and, on the other hand, to rely on several years of historical information to improve our forecast. While one possible solution to this problem could consist on retraining the eSPA model on predefined time frames, the best approach would be to devise a new method capable of simultaneously modelling the time-dependence of the classification features and their impact on the response variable.

CRediT authorship contribution statement

Edoardo Vecchi: Writing – original draft, Writing – review & editing, Conceptualization, Methodology, Visualization, Software, Validation. **Gabriele Berra:** Writing – original draft, Conceptualization, Methodology, Visualization, Software, Data curation. **Steffen Albrecht:** Software. **Patrick Gagliardini:** Conceptualization, Methodology, Supervision. **Illia Horenko:** Conceptualization, Methodology, Supervision, Software, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data analysed in this study are publicly available on Yahoo Finance: https://finance.yahoo.com/

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix A. Computation of technical indicators and features

A.1. Description and computation of the technical indicators

In this section, we briefly describe the technical indicators considered in our analysis, and provide technical details about their computation and the reference labels used in some of the figures. A complete list of the explanatory features used for the solution of the binary classification problem can be found in Appendix A.2. All indicators have been computed by using the technical analysis functions contained in MATLAB *Financial Toolbox*, and the eventual tuneable parameters are specified directly in their description.

Oscillators. We considered the following oscillators:

- **Accumulation/distribution oscillator (`ADO`)** is a momentum oscillator based on the assumption that the more volume accompanies a price move – either up or down – the greater the movement will be Kaufman (1978). It tries to confirm changes in the direction of the prices. It is computed through the MATLAB function `adosc()` according to the formulas:

$$\mathrm{ADO}_t = \frac{(\mathrm{Close}_t - \mathrm{Low}_t) - (\mathrm{High}_t - \mathrm{Close}_t)}{\mathrm{High}_t - \mathrm{Low}_t} \cdot \mathrm{Volume}_t, \tag{A.1}$$

$$\mathrm{ADO}_{t+1} = \mathrm{ADO}_t + \frac{(\mathrm{Close}_{t+1} - \mathrm{Low}_{t+1}) - (\mathrm{High}_{t+1} - \mathrm{Close}_{t+1})}{\mathrm{High}_{t+1} - \mathrm{Low}_{t+1}} \cdot \mathrm{Volume}_{t+1}. \tag{A.2}$$

- **Chaikin Oscillator (`CO`)** is a momentum indicator that attempts to anticipate changes in the direction of the market by considering the difference between the 10-day- and 3-day- exponential moving averages (EMA) of the accumulation/distribution lines (ADL) (Achelis, 2013). The ADL formula is reported among the indicators. The Chaikin Oscillator is computed through the MATLAB function `chaikosc()` according to the formula:

$$\mathrm{CO} = (\text{3-day-EMA of ADL}) - (\text{10-day-EMA of ADL}). \tag{A.3}$$

- **Moving Average Convergence/Divergence (`MACD`)** is a trend-following indicator, which is usually paired with a "signal line" that is used to show buy or sell opportunities. As suggested in Achelis (2013), it is calculated as the difference between a 12-day- and a 26-day-EMA of a security price, while a 9-day-EMA of the MACD is plotted on top of the indicator as a signal line. It is computed through the MATLAB function `macd()` according to:

$$\mathrm{MACD} = \text{12-day-EMA of Close} - \text{26-day-EMA of Close}, \tag{A.4}$$

$$\text{Signal Line} = \text{9-day-EMA of MACD}. \tag{A.5}$$

- **Stochastic Oscillator (`SO`)** is a momentum indicator – with a value bounded between 0 and 100 – that compares the closing price of a security with its range of prices in a particular period of time (Achelis, 2013). The stochastic oscillator is composed of two parts: fast stochastics and slow stochastics. The fast stochastics (Fast %K and Fast %D) are more sensitive to price variations and react more quickly, thus incurring in the risk of providing false signals. On the other hand, the slow stochastics (Slow %K and Slow %D) are based on a smoothing of the fast stochastics, and provide less but more accurate signals. The `SO` is computed through the MATLAB function `stochosc()` according to the formulas:

$$\text{Fast \%K} = \frac{\mathrm{Close}_t - \text{LL in 10 days}}{\text{HH in 10 days} - \text{LL in 10 days}} \cdot 100, \tag{A.6}$$

$$\text{Fast \%D} = \text{3-day-EMA of Fast \%K}, \tag{A.7}$$

$$\text{Slow \%K} = \text{3-day-EMA of Fast \%D}, \tag{A.8}$$

$$\text{Slow \%D} = \text{3-day-EMA of Slow \%K}. \tag{A.9}$$

  As specified above, we base the computations on a 10-day-window.
- **Acceleration (`Acc`)** is an indicator that identifies strong movements in the market, by computing the difference of the current momentum with that computed n periods ago (Kaufman, 1978). The `Acc` is computed through the MATLAB function `tsaccel()` over a reference period of $n = 12$ days, according to the following formula based on momentum (see next point):

$$\mathrm{Acc}_t = \mathrm{Mom}_t - \mathrm{Mom}_{t-12}. \tag{A.10}$$

- **Momentum (`Mom`)** is an indicator which computes the difference between the current data and the data n periods ago (Kaufman, 1978). As exemplified in the reference, we consider a period of $n = 12$ days in the computations. We used the MATLAB function `tsmom()`, which relies on the formula:

$$\mathrm{Mom}_t = \mathrm{Price}_t - \mathrm{Price}_{t-12}. \tag{A.11}$$

Stochastics. We considered the following stochastics:

- **Chaikin Volatility (CV)** compares the difference between the high and low of a given security price (Achelis, 2013). Following Chaikin recommendation, we consider a 10-day-EMA in the computations. We used the MATLAB function `chaikvolat()`, which used the formula:

$$CV_t = \frac{\text{10-day-EMA of(High} - \text{Low)} - \text{10-day-EMA of (High}_{t-10} - \text{Low}_{t-10})}{\text{10-day-EMA of (High}_{t-10} - \text{Low}_{t-10})} \cdot 100. \tag{A.12}$$

- **William %R (W%R)** is a momentum indicator that measures overbought and oversold levels of a given security (Achelis, 2013). Following the example reference, we consider a time window of 14 days. We used the MATLAB function `willpctr()`, which relies on the formula:

$$\frac{\text{HH in 14 days} - \text{Close}_t}{\text{HH in 14 days} - \text{LL in 14 days}} \cdot 100. \tag{A.13}$$

Indexes. We considered the following indexes:

- **Negative Volume Index (NVI)** is a cumulative indicator using volume changes to predict when the "smart money" (i.e., the capital controlled by experienced agents like, e.g., institutions, central banks, funds) takes position (Achelis, 2013). We used the MATLAB function `negvolidx()` and, starting arbitrarily with $\text{NVI}_{t-1} = 100$, it is computed as follows:

$$\text{NVI}_t = \begin{cases} \text{NVI}_{t-1} + \frac{\text{Close}_t - \text{Close}_{t-1}}{\text{Close}_{t-1}} \cdot \text{NVI}_{t-1}, & \text{if Volume}_t < \text{Volume}_{t-1}, \\ \text{NVI}_{t-1}, & \text{otherwise.} \end{cases} \tag{A.14}$$

- **Positive Volume Index (PVI)** provides signals for changes in the trend of the prices, and it is also used for confirming price reversal (Achelis, 2013). We used the MATLAB function `posvolidx()` and, starting arbitrarily with $\text{PVI}_{t-1} = 100$, it is computed as follows:

$$\text{PVI}_t = \begin{cases} \text{PVI}_{t-1} + \frac{\text{Close}_t - \text{Close}_{t-1}}{\text{Close}_{t-1}} \cdot \text{PVI}_{t-1}, & \text{if Volume}_t > \text{Volume}_{t-1}, \\ \text{PVI}_{t-1}, & \text{otherwise.} \end{cases} \tag{A.15}$$

- **Relative Strength Index (RSI)** is a momentum indicator used to find overbought and oversold conditions from the magnitude of recent price changes of a given stock or asset (Achelis, 2013). Following Wilder original recommendation, we decided to consider a 14-day-RSI.[6] We used the MATLAB function `rsindex()`, which computes the RSI according to:

$$\text{RSI} = 100 - \frac{100}{1 - \text{RS}}, \tag{A.16}$$

$$\text{RS} = \frac{\frac{1}{14} \sum \text{upward price change}}{\frac{1}{14} \sum |\text{downward price change}|}. \tag{A.17}$$

Indicators. We considered the following indicators:

- **Accumulation/Distribution Line ADL** is a cumulative indicator that, similarly to the Williams Accumulation/Distribution, identifies the accumulation and distribution of a security, while also taking into account the volume (Kaufman, 1978; Achelis, 2013). In the implementation, we used the MATLAB function `adline()`, which relies on the recursive formula:

$$\text{ADL}_t = \sum_{\tau=0}^{t} \left(\frac{(\text{Close}_\tau - \text{Low}_\tau) - (\text{High}_\tau - \text{Close}_\tau)}{(\text{High}_\tau - \text{Low}_\tau)} \cdot \text{Volume}_\tau \right). \tag{A.18}$$

- **Bollinger Band (BB)** is an indicator of price volatility (Achelis, 2013). It is based on three bands: upper band (U), middle band (M) and lower band (L). The upper and lower bands are, usually, two standard deviations away from the price, and the wider the distance between the upper and the lower band, the more volatile is the asset. We considered a time window of 10 days, and used, in the implementation, the MATLAB function `bollinger()`, which relies on the formulas:

$$\text{M} = \frac{\sum_{i=1}^{10} \text{Close}_i}{10}, \tag{A.19}$$

$$\text{U} = \text{M} + 2 * \sqrt{\frac{\sum_{i=1}^{10} (\text{Close}_i - \text{M})^2}{10}}, \tag{A.20}$$

$$\text{L} = \text{M} - 2 * \sqrt{\frac{\sum_{i=1}^{10} (\text{Close}_i - \text{M})^2}{10}}. \tag{A.21}$$

[6] Further information can be found in the book where the RSI was originally introduced: J. W. Wilder, *New concepts in technical trading systems*, Trend Research, 1978.

- **Highest High (HH)** keeps track of the highest high over n periods (Achelis, 2013). Here, we consider $n = 14$ days and use the MATLAB function `hhigh()`.
- **Lowest Low (LL)** keeps track of the lowest low over n periods (Achelis, 2013). Here, we consider $n = 14$ days and use the MATLAB function `llow()`.
- **Median Price (MP)** computes the median price over n periods (Achelis, 2013). Here, we consider $n = 14$ days and use the MATLAB function `medprice()`.
- **On-Balance Volume (OBV)** is a momentum indicator that relates the direction of the volume flow to the price of the security (Achelis, 2013). For the implementation, we used the MATLAB function `onbalvol()`, which, starting from the value of the Volume at time t, computes the OBV as:

$$\mathrm{OBV}_{t+1} = \begin{cases} \mathrm{OBV}_t + \mathrm{Volume}_{t+1} & \text{if } \mathrm{Close}_{t+1} > \mathrm{Close}_t, \\ \mathrm{OBV}_t - \mathrm{Volume}_{t+1} & \text{if } \mathrm{Close}_{t+1} < \mathrm{Close}_t, \\ \mathrm{OBV}_t & \text{if } \mathrm{Close}_{t+1} = \mathrm{Close}_t. \end{cases} \tag{A.22}$$

- **Price-Volume Trend (PVT)** is a cumulative indicator aimed at identifying the direction and trend of a security from its volume (Achelis, 2013). We used the MATLAB function `pvtrend`, which recursively computes the indicator as:

$$\mathrm{PVT}_{t+1} = \sum_{\tau=0}^{t} \left(\frac{\mathrm{Close}_{\tau+1} - \mathrm{Close}_\tau}{\mathrm{Close}_\tau} \cdot \mathrm{Volume}_{\tau+1} \right). \tag{A.23}$$

- **Typical Price (TP)** is a simple average of the prices over n periods (Achelis, 2013). Here, we consider $n = 3$ days and use the MATLAB function `typprice()`, which computes the indicator according to the formula:

$$\mathrm{TP}_t = \frac{\mathrm{High}_t + \mathrm{Low}_t + \mathrm{Close}_t}{3}. \tag{A.24}$$

- **Volume Rate of Change (VRC)** is an indicator consisting in the rate of change of the volume over n periods (Achelis, 2013). Sticking to the provided reference value, we consider $n = 12$ days and use the MATLAB function `volroc()`, which relies on the formula:

$$\mathrm{VRC}_t = \frac{\mathrm{Volume}_t - \mathrm{Volume}_{t-12}}{\mathrm{Volume}_{t-12}} \cdot 100. \tag{A.25}$$

- **Weighted Close (WC)** computes the weighted closing price based on the Open, High, Low and Close of any given day (Achelis, 2013). The implementation relies on the MATLAB function `wclose()`, which is based on the formula:

$$\mathrm{WC}_t = \frac{2 \cdot \mathrm{Close}_t + \mathrm{High}_t + \mathrm{Low}_t}{4}. \tag{A.26}$$

- **Williams Accumulation/Distribution Line (WADL)** is a cumulative indicator that tries to define whether a stock is being distributed or accumulated (Achelis, 2013). It is similar to the ADL, but WADL does not take into consideration the volume. The chosen implementation relies on the MATLAB function `willad()`, which computes the indicator according to the following steps. We start by defining the quantities:

$$\mathrm{TRH} = t - 1 \text{ close or } t \text{ high, whichever is greater,} \tag{A.27}$$

$$\mathrm{TRL} = t - 1 \text{ close or } t \text{ low, whichever is less.} \tag{A.28}$$

Then, the accumulation/distribution is computed as:

$$\mathrm{A/D}_t = \begin{cases} \mathrm{Close}_t - \mathrm{TRL}, & \text{if } \mathrm{Close}_t > \mathrm{Close}_{t-1}, \\ \mathrm{Close}_t - \mathrm{TRH}, & \text{if } \mathrm{Close}_t < \mathrm{Close}_{t-1}, \\ 0, & \text{if } \mathrm{Close}_t = \mathrm{Close}_{t-1}. \end{cases} \tag{A.29}$$

Finally, given a starting point, WADL is recursively computed as:

$$\mathrm{WADL}_t = \mathrm{A/D}_t + \mathrm{WADL}_{t-1}. \tag{A.30}$$

A.2. Complete list of features used in the experiments

In Table A.2, we report the complete list of features used in the solution of the stock outperformance prediction as a binary classification problem. The reference labels of the technical indicators can be found in Appendix A.1. In order to test how well the ML method can identify "hidden patterns" in the explanatory variables and filter out redundant or irrelevant predictors, we computed some of the technical indicators (namely BB, MACD, Acc and Mom) on all the available historical information of each stock.

Table A.2
Complete list of features used for the prediction of the stock outperformance.

Idx	Indicator	Idx	Indicator	Idx	Indicator
1	SO Fast %K	22	BB|U_Volume	43	Acc_Low
2	SO Fast %D	23	BB|L_Open	44	Acc_Close
3	SO Slow %K	24	BB|L_High	45	Acc_AdjClose
4	SO Slow %D	25	BB|L_Low	46	Acc_Volume
5	ADO	26	BB|L_Close	47	Mom_Open
6	CO	27	BB|L_AdjClose	48	Mom_High
7	W%R	28	BB|L_Volume	49	Mom_Low
8	NVI	29	MACD_Open	50	Mom_Close
9	PVI	30	MACD_High	51	Mom_AdjClose
10	RSI	31	MACD_Low	52	Mom_Volume
11	BB|M_Open	32	MACD_Close	53	HH
12	BB|M_High	33	MACD_AdjClose	54	LL
13	BB|M_Low	34	MACD_Volume	55	MP
14	BB|M_Close	35	MACD|SL_Open	56	OBV
15	BB|M_AdjClose	36	MACD|SL_High	57	PVT
16	BB|M_Volume	37	MACD|SL_Low	58	TP
17	BB|U_Open	38	MACD|SL_Close	59	VRC
18	BB|U_High	39	MACD|SL_AdjClose	60	WC
19	BB|U_Low	40	MACD|SL_Volume	61	WADL
20	BB|U_Close	41	Acc_Open	62	CV
21	BB|U_AdjClose	42	Acc_High	63	ADL

A.3. Additional features for Fig. 5

In Fig. 5, apart from the technical indicators listed in Table A.2, we consider the following explanatory features:

- **Case "Returns":** For each stock, we have two additional features (for a total of 65), which correspond – respectively – to the return of the stock (either NESN.SW, GIVN.SW or LOGN.SW, indicated with the label `SR|t`) and to the return of the market (SMI, indicated with the label `MR|t`). The returns have been computed, both for the stocks and the index, according to the following equation:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}, \tag{A.31}$$

  where P_{t-1} and P_t indicate the price yesterday and today, respectively.
- **Case "Lagged returns":** For each stock, we have six additional features (for a total of 69), which correspond to the return and lagged returns of the stock (either NESN.SW, GIVN.SW or LOGN.SW) and of the market (SMI). Specifically, we have `SR|t` and `MR|t`, which correspond to the returns defined and computed according to the formula reported above, and `SR|t-5`, `SR|t-10`, `MR|t-5` and `MR|t-10`, which indicate the lagged returns of the stock and the index, computed as:

$$R_{t-5} = \frac{P_t - P_{t-5}}{P_{t-5}} \quad \text{and} \quad R_{t-10} = \frac{P_t - P_{t-10}}{P_{t-10}}, \tag{A.32}$$

  where P_t, P_{t-5} and P_{t-10} indicate, respectively, the price today, 5 days ago and 10 days ago for both the stocks and the index.

Appendix B. Description of the models and implementation details

In this section, we provide additional information about the ML models used in the experiments and their implementation, while also discussing the hyperparameter tuning performed for each method during the training phase. All the methods are implemented in MATLAB and, for each of them, we report either the source of the algorithm or the name of the relevant MATLAB function. As a general remark, in order to avoid extreme differences in the range of values, each explanatory feature has been rescaled in the interval $[0, 1]$ before the simulations. All the experiments have been performed by running scripts written entirely in MATLAB (version: R2020b) on a laptop with a 2.2 GHz 6-Core Intel Core i7 processor with 16 GB 2400 MHz DDR4 RAM. In order to speed up the computations, MATLAB *Parallel Computing Toolbox* functionalities have been used to parallelise the computations on all the 6 available cores. On the other hand, the computational cost comparison of the different methods has been run on a single computational thread, by disabling MATLAB automatic thread level parallelism through the command `maxNumCompThreads(1)`.

B.1. Mathematical formulation of eSPA

Since the eSPA method, as shown in Section 4, markedly outperforms all its competitors in the prediction of stock outperformance over the market and allows a direct interpretation of the results obtained through the entropic filtering of the explanatory features, we deem necessary to provide the reader with specific details about its structure. Instead of the original implementation of the eSPA algorithm (Horenko, 2020), in this analysis we considered eSPA+, an improved version introduced in Vecchi et al. (2022), due to

its higher stability and lower computational cost. Generally speaking this method tackles, simultaneously, three distinct problems: (i) finding the optimal discrete representation of the input data through a set of non-overlapping geometric boxes; (ii) maximising the Shannon entropy of the explanatory features while filtering out the dimensions irrelevant for classification; (iii) solving the supervised classification problem, by linking the discretisation boxes with the provided label probabilities. More specifically, the eSPA+ algorithm aims at finding the optimal discretisation of the data space $X \in \mathbb{R}^{D\times T}$ through K non-overlapping geometric boxes, while simultaneously solving the supervised classification problem for the given label probabilities $\Pi \in \mathbb{R}^{M\times T}$, by minimising the entropic functional:

$$\mathcal{L}_{\text{eSPA+}} = \frac{1}{T}\sum_{d=1}^{D} W_d \sum_{k=1}^{K}\sum_{t=1}^{T} \Gamma_{k,t}\left(X_{d,t} - S_{d,k}\right)^2 + \frac{\varepsilon_{\text{E}}}{D}\sum_{d=1}^{D} W_d \log(W_d) - \frac{\varepsilon_{\text{CL}}}{TM}\sum_{m=1}^{M}\sum_{t=1}^{T}\Pi_{m,t}\sum_{k=1}^{K}\Gamma_{k,t}\log\left(\Lambda_{m,k}\right), \tag{B.1}$$

with respect to $\Gamma \in \Omega_\Gamma$, $\Lambda \in \Omega_\Lambda$ and $W \in \Omega_W$, under the following set of stochasticity constraints:

$$\Omega_\Gamma := \left\{ \Gamma \in \mathbb{R}^{K\times T} \,\middle|\, \forall k,t:\ \Gamma_{k,t} \in \{0,1\} \ \wedge\ \forall t:\ \sum_{k=1}^{K}\Gamma_{k,t} = 1 \right\}, \tag{B.2}$$

$$\Omega_\Lambda := \left\{ \Lambda \in \mathbb{R}^{M\times K} \,\middle|\, \forall m,k:\ \Lambda_{m,k} \in [0,1] \ \wedge\ \forall k:\ \sum_{m=1}^{M}\Lambda_{m,k} = 1 \right\}, \tag{B.3}$$

$$\Omega_W := \left\{ W \in \mathbb{R}^{D} \,\middle|\, \forall d:\ W_d \in [0,1] \ \wedge\ \sum_{d=1}^{D} W_d = 1 \right\}. \tag{B.4}$$

In the minimisation problem above, $S \in \mathbb{R}^{D\times K}$ is a matrix containing the coordinates of the centres of the discretisation boxes, $\Gamma \in \mathbb{R}^{K\times T}$ is a stochastic affiliation matrix which links every data point to a specific box, $W \in \mathbb{R}^{D}$ is a stochastic vector which quantifies the importance of every feature in the prediction model, $\Lambda \in \mathbb{R}^{M\times K}$ is a matrix of conditional probabilities linking the labels and the discretisation boxes and $\Pi \in \mathbb{R}^{M\times T}$ is a given matrix containing the data labels probabilities for the supervised learning problem. It is important to notice that the eSPA+ formulation, unlike the original eSPA algorithm, considers a set of discrete box affiliations Γ: in other words, a single data instance can belong only to a given box, and not to more than one box simultaneously. This different segmentation of the space allows to derive a closed-form solution for all the minimisation sub-problems solved at every iteration, thus dramatically reducing the overall computational cost of the algorithm without hindering its predictive performance. More specifically, the functional is composed of three parts: (i) the first term looks for the optimal discrete representation of the input data X, which, in our application, contains the values of the technical indicators for every time instance; (ii) the second term maximises the Shannon entropy of the feature selection vector W and actively contributes at filtering out those dimensions without a significant impact on the optimal discretisation of the data space; (iii) the third term handles the supervised classification problem by linking the boxes and the labels through the exact Law of total probability – represented by the equality $\Pi = \Lambda\Gamma$ – and by minimising the Kullback–Leibler divergence of the classification error, i.e., $D_{KL} = (\Pi \parallel \Lambda\Gamma)$. In Appendix C.2, we show that the weights of vector W allow a direct interpretation of the algorithm output, and we provide additional details on the entropic filtering. Concerning the implementation details, the eSPA algorithm has been implemented entirely in the MATLAB language, according to the closed-form solutions of the minimisation sub-problems provided in Theorem 2 from Vecchi et al. (2022).

B.1.1. Hyperparameter tuning for eSPA

The eSPA algorithm presents three hyperparameters that need to be tuned: (i) the number K of geometric boxes used in the discretisation of the feature space; (ii) the value of the constant $\varepsilon_{\text{E}} \geq 0$, which regulates the relative importance of entropy maximisation in the overall optimisation procedure; (iii) the value of the constant $\varepsilon_{\text{CL}} \geq 0$, which tunes the relevance of the supervised classification problem. In order to assess the optimal value of K, we explored a set of integer values ranging from 10 to 30, with the value $K = 17$ representing the best choice in the majority of the simulations. As far as concerns ε_{E}, we considered the following values: {0.0001, 0.001, 0.002, 0.005, 0.01, 0.02, 0.04, 0.1, 1, 10}. Finally, for ε_{CL}, our grid search considered the following values: {0.000001, 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.05, 0.1}.

B.2. Implementation details of the other ML algorithms

Here, we provide details about the grid search performed to tune the hyperparameters of the different state-of-the-art ML algorithms. Due to the challenging limitations imposed by the small data learning regime, we took particular care of avoiding value combinations that would lead to overfitting. In general, as already mentioned in the discussion of the results in Section 4, we can stress how the careful hyperparameter tuning of the different methods did not allow to observe significant improvements in the predictive performance. In some cases, it reduced the reciprocating behaviour of the different methods (i.e., systematic prediction of the opposite labels), while in others turned out to be detrimental (see, e.g., the discussion reported below for LDA).

RF. The implementation relies on the MATLAB class `TreeBagger()` for classification, which belongs to the *Statistics and Machine Learning Toolbox*. As splitting criterion (`SplitCriterion`), we considered both the Gini diversity index and the cross-entropy (or maximum deviance reduction). For the maximum number of trees, we investigated the following values: {100, 250, 500, 750, 1000}. As potential values of the minimum number of observations pertaining to each tree leaf (`MinLeafSize`) we considered: {3, 5, 10, 15, 20}. For the maximum number of splits (`MaxNumSplits`), given the size of the reference data set, we used the values: {74, 147}.

Lasso GLM. The implementation relies on the MATLAB function `lassoglm()` – belonging to *Statistics and Machine Learning Toolbox* – which solves the minimisation problem:

$$\min_{\beta_0,\beta} \frac{1}{N}\text{Deviance}(\beta_0,\beta) + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{B.5}$$

where the deviance of the model fit to the responses is computed by considering the intercept β_0 and the predictor coefficients β according to the distribution specified by the function parameter `distr`. In this case, since we are dealing with a binary classification problem, we consider an underlying binomial distribution with a logit link function between the mean of the response μ and the linear predictor Xb ($\log(\mu/(1-\mu)) = Xb$). The only hyperparameter that can be tuned in this context is the non-negative regularisation constant λ, aimed at tuning the sparsity promoted by the L^1-norm. For λ, we consider the following values: {0.0000001, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100}.

LDA. The implementation uses the MATLAB function `fitcdiscr()` – belonging to the *Statistics and Machine Learning Toolbox* – which aims at minimising the predicted classification cost, according to the formula:

$$\hat{y} = \arg\min_{y=1,2} \sum_{k=1}^{2} \hat{P}(k|x)C(y|k), \tag{B.6}$$

where $\hat{y}$ is the predicted classification, $\hat{P}(k|x)$ is the posterior probability of class k for observation x and $C(y|k)$ is the misclassification cost (which is equal to 0 for correct classification, and to 1 for wrong classification). Each one of the two classes Y_1 and Y_2 is assumed to generate the data X through a multivariate normal distribution and, since we are considering a linear discriminant, both classes are assumed to have the same variance–covariance matrix. In this case, we set the parameter `DiscrimType` to 'pseudolinear', in order to allow the computation of the pseudo inverse of the variance–covariance matrix. We can optimise two hyperparameters: `Delta`, i.e., the linear coefficient threshold, which filters out the coefficient smaller than its value; `Gamma`, which indicates the amount of regularisation applied to the estimation of the variance–covariance matrix of predictors. Both parameters have been optimised by setting the variable `OptimizeHyperparameters` to 'auto'. However, empirical experiments showed that finding the optimal values of the hyperparameters was actually decreasing the model performance, pushing the AUC towards 0.5 (i.e., random guess about stock outperformance). Thus, in many simulations we set both of them to 0.

SVM. The implementation uses the MATLAB function `fitcsvm` – included in the *Statistics and Machine Learning Toolbox* – which has been tested against the following specifics. First of all, we considered, as kernel functions (parameter `KernelFunction`), both a Radial Basis Function (RBF) kernel and a polynomial kernel (with the order – `PolynomialOrder` – ranging from 2 to 4). The other two tuneable hyperparameters are: `KernelScale`, which indicates the threshold according to which are divided the elements of the predictor matrix (values: {0.000001, 0.00001, 0.0001, 0.01, 1, 100}); `BoxConstraint`, which corresponds to the maximum possible penalty imposed on those observations that violate the margins (values: {0.0001, 0.001, 0.01, 0.1, 1, 100, 10000}).

Shallow NN. The implementation relies on two MATLAB functions from the *Deep Learning Toolbox*. First of all, the function `feedforwardnet()` has been used to create the `net` object representing the shallow NN, and different sizes have been considered for the hidden layer (`hiddenSizes` assumed the values {4, 8, 12, 16}). As a second and final step, the `net` object has been properly trained by using the `train()` function and then used to classify the provided data.

DL with LSTM. The network object has been assembled with the functionalities provided by the *Deep Learning Toolbox* and the MATLAB function `trainNetwork()` has been used to train the model, by providing, as input, the following hidden layers. We used 63 sequence input layers (i.e., as many as our technical indicators) and different numbers of hidden units in the long short-term memory layers (for `numHiddenUnits`, we considered the values {2, 5, 10}). Since we are dealing with a binary classification problem, the number of components in the fully connected layers has been set to 2 (i.e., the number of classes). The size of the mini-batch used during each training iteration (i.e., the subset of training values used to compute the gradient of the loss function and to update the weights), has been tuned by setting the parameter `MiniBatchSize` to {8, 16, 32}.

Appendix C. Model interpretability and selection criteria

In this appendix, we provide additional details on the computation of the different metrics used to interpret the results of the ML algorithms used in the experiments. We also delve deeper in the entropic filtering process and briefly present the computation of AIC and BIC for the eSPA models.

C.1. Computation and implementation of PDP, ICEP and SHAP

The model interpretability has been evaluated according to:

- **PDP:** it is an a posteriori examination of the behaviour of a given model. Specifically, it summarises the average relationship between the model output and a selected feature, by assessing how the model output changes as a consequence of changes in the feature value. The implementation relies on the MATLAB function `plotPartialDependence()`. In order to compute the PDP, for each model we had to build a custom function handle with different inputs, depending on the outputs of the

MATLAB functions explained in Appendix B. Specifically, for RF, SVM, and LDA, we used, as input, directly the model output computed with the related function. For shallow NN and LSTM, we used the output net objects of each model to compute directly the probabilities. For GLM, we had to save and use the parameters `B` and `Intercept` containing, respectively, the fitted coefficients and the intercept. Finally, for eSPA, we had to use the parameters Λ, `K`, `W`, and `S` to compute the data segmentation Γ and, subsequently, we estimated the classification label probabilities by computing the matrix product $\Lambda\Gamma$.

- **ICEP:** it can be seen as a further insight on the computations from which the PDP is assembled. Indeed, the red line represented in the ICEP corresponds exactly to the one plotted in the PDP, and it is computed by taking the average of the impact on each observation (represented by the grey lines). For the actual computation, we used the same MATLAB function – i.e., `plotPartialDependence()` – and the same function handles defined above for the PDP, by specifying, for the parameter `Conditional`, the value `'centred'` to produce the desired plot.
- **SHAP:** it is an a posteriori method, proposed in Lundberg and Lee (2017), which helps understanding of the behaviour of a model by outputting the impact of each feature on the prediction. For the implementation, we used the MATLAB function `shapley()` with the model-agnostic Kernel SHAP method. As in the case of the PDP, the function needs a model that outputs a probability, and, thus, we used the same function handles created *ad-hoc* for the PDP also for the computation of the SHAP values.

C.2. Description of the entropic filtering process

The eSPA algorithm specifically computes the relevance of each feature on the optimal box discretisation and filters out those dimensions which are irrelevant for the solution of the classification problem. If we take a closer look at the functional in Eq. (B.1), we can notice that the first two terms contain the stochastic vector $W \in \mathbb{R}^D$ (i.e., its components are probabilities), which – at every iteration of the algorithm – assigns a weight to the features by maximising the entropy of their distribution. It is worth mentioning that the maximum entropy limit of W – achieved when the regularisation parameter ε_E tends to ∞ – consists in a vector where every component has a value of $1/D$, with $D = 63$ representing the number of features (Gerber et al., 2020). This distribution of W leads to a scenario in which the uncertainty is maximised, since the algorithm would be unable to distinguish which features have an active role on the stock outperformance prediction and which are just noise, and, thus, irrelevant for the classification process. The overall importance of entropy maximisation on the whole constrained minimisation of the eSPA functional $\mathcal{L}_{\text{eSPA+}}$ is tuned by the regularisation parameter ε_E, whose value – as explained in Appendix B – cannot be determined a priori, but needs to be assessed during the hyperparameter tuning of the algorithm.

C.3. Computation of the AIC and BIC

Here, we show that all the quantities necessary for the computation of the AIC and BIC can be straightforwardly derived for the eSPA method. The AIC estimates the quality of a statistical model through the formula:

$$\text{AIC} = 2k - 2\ell(\Theta|x), \tag{C.1}$$

where k represents the number of model parameters $\Theta = [\theta_1, \dots, \theta_k]$ and $\ell(\Theta|x)$ indicates the log-likelihood function of these parameters given the observed outcome x of the data X. Following the notation used for the eSPA algorithm, the AIC can be rewritten as:

$$\text{AIC} = 2\tilde{d} - 2\ell(\Lambda\Gamma|\Pi), \tag{C.2}$$

where $\tilde{d}$ represents the number of parameters with a value higher than the maximum entropy threshold w_{maxEnt} defined in the previous section and in the manuscript. In it is worth noting that the maximum entropy scenario would be easily achieved by considering a conditional probability matrix Λ which assigns to every label (i.e., presence or absence of stock outperformance over the market) the same probability regardless of the box affiliation. The value of the log-likelihood $\ell(\Lambda\Gamma|\Pi)$ can be easily computed by comparing the estimated label probabilities $\hat{\Pi} = \Lambda\Gamma$ with the true label probabilities Π. In order to allow the computation of the AIC also in case of sharp misclassification, we set the minimum value achievable by every observation to $\log(2.2204 \cdot 10^{-16})$, which represents MATLAB lowest threshold in double precision arithmetic. As far as concerns the BIC, it estimates the quality of a statistical model with the formula:

$$\text{BIC} = k\ln(n) - 2\ell(\Theta|x), \tag{C.3}$$

where n represents the number of observations in the sample. Adopting again a notation consistent with the rest of the paper, we can redefine the BIC as:

$$\text{BIC} = \tilde{d}\ln(T) - 2\ell(\Lambda\Gamma|\Pi). \tag{C.4}$$

The computation of the related eSPA quantities and the MATLAB double-precision threshold has been performed as described in the AIC case. From Eq. (C.4), we can notice that the BIC tends to penalise more than the AIC the inclusion of additional parameters into the model. Indeed, given the size of the reference data set, any additional parameter weights roughly 2.5 times more in this indicator than in the AIC, and the selection of those models with as few parameters as possible is highly preferred.

References

Achelis, S.B., 2013. Technical Analysis from A to Z, second ed. McGraw-Hill.

Agarwal, S., Kumar, S., Goel, U., 2019. Stock market response to information diffusion through internet sources: A literature review. Int. J. Inf. Manage. 45, 118–131. http://dx.doi.org/10.1016/j.ijinfomgt.2018.11.002.

Allen, F., Karjalainen, R., 1999. Using genetic algorithms to find technical trading rules. J. Financ. Econ. 51 (2), 245–271. http://dx.doi.org/10.1016/s0304-405x(98)00052-x.

Ammar, I.B., Hellara, S., Ghadhab, I., 2020. High-frequency trading and stock liquidity: An intraday analysis. Res. Int. Bus. Finance 53, 101235. http://dx.doi.org/10.1016/j.ribaf.2020.101235.

Aslam, F., Hunjra, A.I., Ftiti, Z., Louhichi, W., Shams, T., 2022. Insurance fraud detection: Evidence from artificial intelligence and machine learning. Res. Int. Bus. Finance 62, 101744. http://dx.doi.org/10.1016/j.ribaf.2022.101744.

Avramov, D., Chordia, T., Goyal, A., 2006. Liquidity and autocorrelations in individual stock returns. J. Finance 61 (5), 2365–2394. http://dx.doi.org/10.1111/j.1540-6261.2006.01060.x.

Ben Jabeur, S., Serret, V., 2023. Bankruptcy prediction using fuzzy convolutional neural networks. Res. Int. Bus. Finance 64, 101844. http://dx.doi.org/10.1016/j.ribaf.2022.101844.

Bianchini, M., Scarselli, F., 2014. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. IEEE Trans. Neural Netw. Learn. Syst. 25 (8), 1553–1565. http://dx.doi.org/10.1109/tnnls.2013.2293637.

Borovkova, S., Tsiamas, I., 2019. An ensemble of LSTM neural networks for high-frequency stock market classification. J. Forecast. 38 (6), 600–619. http://dx.doi.org/10.1002/for.2585.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 2017. Classification And Regression Trees. Routledge, http://dx.doi.org/10.1201/9781315139470.

Chan, L.K., Jegadeesh, N., Lakonishok, J., 1996. Momentum strategies. J. Finance 51 (5), 1681–1713. http://dx.doi.org/10.1111/j.1540-6261.1996.tb05222.x.

Claeskens, G., Hjort, N.L., 2008. Model Selection and Model Averaging. In: Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, http://dx.doi.org/10.1017/CBO9780511790485.

Costola, M., Hinz, O., Nofer, M., Pelizzon, L., 2023. Machine learning sentiment analysis, COVID-19 news and stock market reactions. Res. Int. Bus. Finance 64, 101881. http://dx.doi.org/10.1016/j.ribaf.2023.101881.

De Bondt, W.F., Thaler, R.H., 1995. Financial decision-making in markets and firms: A behavioral perspective. In: Handbooks in Operations Research and Management Science, Vol. 9. Elsevier, pp. 385–410. http://dx.doi.org/10.1016/s0927-0507(05)80057-x.

Fama, E.F., 1970. Efficient capital markets: A review of theory and empirical work. J. Finance 25 (2), 383–417. http://dx.doi.org/10.2307/2325486.

Fama, E.F., Fisher, L., Jensen, M.C., Roll, R., 1969. The adjustment of stock prices to new information. Int. Econ. Rev. 10 (1), 1–21. http://dx.doi.org/10.2307/2525569.

Fu, T.-C., Chung, C.-P., Chung, F.-L., 2013. Adopting genetic algorithms for technical analysis and portfolio management. Comput. Math. Appl. 66 (10), 1743–1757. http://dx.doi.org/10.1016/j.camwa.2013.08.012.

Gagliardini, P., Gourieroux, C., Rubin, M., 2021. Positional portfolio management. J. Financ. Econom. 19 (4), 650–706. http://dx.doi.org/10.1093/jjfinec/nbz022.

Gao, T., Chai, Y., 2018. Improving stock closing price prediction using recurrent neural network and technical indicators. Neural Comput. 30 (10), 2833–2854. http://dx.doi.org/10.1162/neco_a_01124.

Gerber, S., Pospíšil, L., Navandar, M., Horenko, I., 2020. Low-cost scalable discretization, prediction, and feature selection for complex systems. Sci. Adv. 6 (5), eaaw0961. http://dx.doi.org/10.1126/sciadv.aaw0961.

Goldstein, A., Kapelner, A., Bleich, J., Pitkin, E., 2015. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. J. Comput. Graph. Statist. 24 (1), 44–65. http://dx.doi.org/10.1080/10618600.2014.907095.

Goodell, J.W., Kumar, S., Lim, W.M., Pattnaik, D., 2021. Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis. J. Behav. Exp. Finance 32, 100577. http://dx.doi.org/10.1016/j.jbef.2021.100577.

Goodell, J.W., Kumar, S., Rao, P., Verma, S., 2023. Emotions and stock market anomalies: A systematic review. J. Behav. Exp. Finance 37, 100722. http://dx.doi.org/10.1016/j.jbef.2022.100722.

Hasan, M.M., Popp, J., Oláh, J., 2020. Current landscape and influence of big data on finance. J. Big Data 7 (1), http://dx.doi.org/10.1186/s40537-020-00291-z.

Hawley, D.D., Johnson, J.D., Raina, D., 1990. Artificial neural systems: A new tool for financial decision-making. Financ. Anal. J. 46 (6), 63–72. http://dx.doi.org/10.2469/faj.v46.n6.63.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Horenko, I., 2020. On a scalable entropic breaching of the overfitting barrier for small data problems in machine learning. Neural Comput. 32 (8), 1563–1579. http://dx.doi.org/10.1162/neco_a_01296.

Huang, J.-Z., Huang, W., Ni, J., 2019. Predicting bitcoin returns using high-dimensional technical indicators. J. Finance Data Sci. 5 (3), 140–155. http://dx.doi.org/10.1016/j.jfds.2018.10.001.

Ingersoll, J.E., 1987. Theory of Financial Decision Making. Volume 3. Rowman & Littlefield.

Israel, R., Kelly, B.T., Moskowitz, T.J., 2020. Can machines 'learn' finance? J. Invest. Manag. 18 (2), 23–36. http://dx.doi.org/10.2139/ssrn.3624052.

James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An Introduction to Statistical Learning, Vol. 112. Springer, http://dx.doi.org/10.1007/978-1-4614-7138-7.

Jegadeesh, N., Titman, S., 1993. Returns to buying winners and selling losers: Implications for stock market efficiency. J. Finance 48 (1), 65–91. http://dx.doi.org/10.1111/j.1540-6261.1993.tb04702.x.

Kaufman, P.J., 1978. The New Commodity Trading Systems and Methods. John Wiley and Sons.

Li, A.W., Bastos, G.S., 2020. Stock market forecasting using deep learning and technical analysis: A systematic review. IEEE Access 8, 185232–185242. http://dx.doi.org/10.1109/access.2020.3030226.

Li, X., Wu, P., Wang, W., 2020. Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong. Inf. Process. Manage. 57 (5), 102212. http://dx.doi.org/10.1016/j.ipm.2020.102212.

Liu, Y., Li, Z., Nekhili, R., Sultan, J., 2023. Forecasting cryptocurrency returns with machine learning. Res. Int. Bus. Finance 64, 101905. http://dx.doi.org/10.1016/j.ribaf.2023.101905.

Lo, A.W., Mamaysky, H., Wang, J., 2000. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. J. Finance 55 (4), 1705–1765. http://dx.doi.org/10.1111/0022-1082.00265.

Lukac, L.P., Brorsen, B.W., Irwin, S.H., 1988. A test of futures market disequilibrium using twelve different technical trading systems. Appl. Econ. 20 (5), 623–639. http://dx.doi.org/10.1080/00036848800000113.

Lundberg, S.M., Lee, S.-I., 2017. A unified approach to interpreting model predictions. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Vol. 30. Curran Associates, Inc..

Van der Maaten, L., Hinton, G., 2008. Visualizing data using t-SNE. J. Mach. Learn. Res. 9 (11), 2579–2605.

Markowitz, H.M., 1952. Portfolio selection. J. Finance 7 (1), 77–91. http://dx.doi.org/10.2307/2975974.

Nabipour, M., Nayyeri, P., Jabani, H., S., S., Mosavi, A., 2020. Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis. IEEE Access 8, 150199–150212. http://dx.doi.org/10.1109/access.2020.3015966.

Neely, C.J., Rapach, D.E., Tu, J., Zhou, G., 2014. Forecasting the equity risk premium: The role of technical indicators. Manage. Sci. 60 (7), 1772–1791. http://dx.doi.org/10.1287/mnsc.2013.1838.

Noble, W.S., 2006. What is a support vector machine? Nature Biotechnol. 24 (12), 1565–1567. http://dx.doi.org/10.1038/nbt1206-1565.

Park, C.H., Park, H., 2008. A comparison of generalized linear discriminant analysis algorithms. Pattern Recognit. 41 (3), 1083–1097. http://dx.doi.org/10.1016/j.patcog.2007.07.022.

Pesaran, M.H., Pick, A., Pranovich, M., 2013. Optimal forecasts in the presence of structural breaks. J. Econometrics 177 (2), 134–152. http://dx.doi.org/10.1016/j.jeconom.2013.04.002.

Rasekhschaffe, K.C., Jones, R.C., 2019. Machine learning for stock selection. Financ. Anal. J. 75 (3), 70–88. http://dx.doi.org/10.1080/0015198x.2019.1596678.

Rudin, C., 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat. Mach. Intell. 1 (5), 206–215. http://dx.doi.org/10.1038/s42256-019-0048-x.

Shorten, C., Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. J. Big Data 6 (1), 60. http://dx.doi.org/10.1186/s40537-019-0197-0.

Shynkevich, A., 2012. Performance of technical analysis in growth and small cap segments of the US equity market. J. Bank. Financ. 36 (1), 193–208. http://dx.doi.org/10.1016/j.jbankfin.2011.07.001.

Shynkevich, Y., McGinnity, T., Coleman, S.A., Belatreche, A., Li, Y., 2017. Forecasting price movements using technical indicators: Investigating the impact of varying input window length. Neurocomputing 264, 71–88. http://dx.doi.org/10.1016/j.neucom.2016.11.095.

Subrahmanyam, A., 2019. Big data in finance: Evidence and challenges. Borsa Istanbul Rev. 19 (4), 283–287. http://dx.doi.org/10.1016/j.bir.2019.07.007.

Taffler, R., 2018. Emotional finance: Investment and the unconscious. Eur. J. Finance 24 (7–8), 630–653. http://dx.doi.org/10.1080/1351847X.2017.1369445.

Tan, Z., Yan, Z., Zhu, G., 2019. Stock selection with random forest: An exploitation of excess return in the Chinese stock market. Heliyon 5 (8), e02310. http://dx.doi.org/10.1016/j.heliyon.2019.e02310.

Tibshirani, R., 1996. Regression Shrinkage and selection via the Lasso. J. R. Stat. Soc. Ser. B Stat. Methodol. 58 (1), 267–288. http://dx.doi.org/10.1111/j.2517-6161.1996.tb02080.x.

Vecchi, E., Pospíšil, L., Albrecht, S., O'Kane, T.J., Horenko, I., 2022. ESPA+: Scalable entropy-optimal machine learning classification for small data problems. Neural Comput. 34 (5), 1220–1255. http://dx.doi.org/10.1162/neco_a_01490.

Wang, Y., Liu, L., Wu, C., 2020. Forecasting commodity prices out-of-sample: Can technical indicators help? Int. J. Forecast. 36 (2), 666–683. http://dx.doi.org/10.1016/j.ijforecast.2019.08.004.

Webb, G.I., Hyde, R., Cao, H., Nguyen, H.L., Petitjean, F., 2016. Characterizing concept drift. Data Min. Knowl. Discov. 30 (4), 964–994. http://dx.doi.org/10.1007/s10618-015-0448-4.

Weiss, K., Khoshgoftaar, T.M., Wang, D., 2016. A survey of transfer learning. J. Big Data 3 (1), 9. http://dx.doi.org/10.1186/s40537-016-0043-6.

Zhao, A.B., Cheng, T., 2022. Stock return prediction: Stacking a variety of models. J. Empir. Financ. 67, 288–317. http://dx.doi.org/10.1016/j.jempfin.2022.04.001.