Expert Systems with Applications 41 (2014) 3651–3661

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Automated trading with performance weighted random forests and seasonality ☆

Ash Booth [a,*], Enrico Gerding [a], Frank McGroarty [b]

[a] Electronics and Computer Science, University of Southampton, SO17 1BJ, United Kingdom
[b] Southampton Management School, University of Southampton, SO17 1BJ, United Kingdom

ARTICLE INFO

Keywords:
Random forests
Ensemble learning
Stock price prediction
Machine learning
Algorithmic trading

ABSTRACT

Seasonality effects and empirical regularities in financial data have been well documented in the financial economics literature for over seven decades. This paper proposes an expert system that uses novel machine learning techniques to predict the price return over these seasonal events, and then uses these predictions to develop a profitable trading strategy. While simple approaches to trading these regularities can prove profitable, such trading leads to potential large drawdowns (peak-to-trough decline of an investment measured as a percentage between the peak and the trough) in profit. In this paper, we introduce an automated trading system based on performance weighted ensembles of random forests that improves the profitability and stability of trading seasonality events. An analysis of various regression techniques is performed as well as an exploration of the merits of various techniques for expert weighting. The performance of the models is analysed using a large sample of stocks from the DAX. The results show that recency-weighted ensembles of random forests produce superior results in terms of both profitability and prediction accuracy compared with other ensemble techniques. It is also found that using seasonality effects produces superior results than not having them modelled explicitly.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

A particularly active area of research among both academics and industry professionals over the last two decades has been the construction of systems that autonomously trade in securities and currencies. The goal of such endeavours is to produce artificial intelligence methods that can be employed to construct systems that are better than or at least as good as their human counterparts in recognising investment opportunities. Many such systems take as inputs past market prices and attempt to generate a trading signal indicating the direction (and sometimes magnitude) of movement of a given security relative to another.

To this end, we describe an automated trading system that, for the first time, uses performance-weighted ensembles of random forests to predict the price return during well documented seasonality events. Although these seasonal regularities are consistent enough to be used as a stand-alone strategy, rates of return are highly volatile. To address this problem, we propose an expert system that captures the current state of the market in its input and uses this information to both predict the profitability of a seasonality trade and act upon this information. In doing so, the system performs risk management while opening and closing trading positions in an attempt to improve the profit and reduce the volatility over a basic seasonality trading strategy.

Before describing such a system, we first investigate the DAX for the following regularities: upward biases at the turn-of-the-month and over exchange holidays, as well as a downward bias over the weekend. Given our findings, we develop an autonomous system for the trading of stocks over seasonality events. To that end, a recency-biased performance-weighted ensemble of random forests is used to predict the expected profit of a seasonality trade given the prevailing market conditions. The random forests are trained and added to the ensemble over time, in an online fashion, so as to capture various phases of the market.

While others have investigated the performance of ensemble systems and random forest algorithms for predicting using unfiltered daily stock prices, as yet, no study has explored the use of ensembles of random forests for building expert systems for trading empirical regularities in stock markets. In more detail, this paper makes the following contributions to the state of the art:

- We present the first ensemble learning system for trading seasonal trends and demonstrate its effectiveness on equity market data.

☆ This work was supported by an EPSRC Doctoral Training Centre Grant (EP/G03690X/1).

* Corresponding author. Tel.: +44 (0)23 8059 4510.
E-mail addresses: ash.booth@soton.ac.uk (A. Booth), eg@ecs.soton.ac.uk (E. Gerding), f.j.mcgroarty@soton.ac.uk (F. McGroarty).
URL: http://www.ashbooth.com (A. Booth).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.12.009

- The highly active fields of online ensemble generation and random forest predictors are fused to produce a novel expert system for stock trading formed from performance weighted ensembles of random forests.
- Using experiments based on real data, we show that our recency biased performance-weighted ensembles of random forests, used for trading seasonal events, outperforms other ensemble methods. These include cumulative weighting systems, simple averaging of regressors as well as a number of non-ensemble regression techniques.

The paper is structured as follows. Section 2 gives an overview of the relevant literature. Section 3 presents an analysis of the seasonality and empirical regulations found in the equity data and describes the features that we use as inputs to the learning system while Section 4 describes the trading algorithm itself. In Section 5 the results are summarised while Section 6 gives concluding remarks and discussed potential future work.

2. Literature

Many attempts have been made to devise a consistently profitable autonomous trading system. Inspiration for such trading systems comes from a variety of fields, ranging from fundamental analysis and econometric modelling to evolutionary computation, machine learning and even news mining (Nuij, Milea, Hogenboom, Frasincar, & Kaymak, 2013). A considerable number of these approaches use machine learning algorithms to generate trading rules. These are typically based on so-called technical analysis, which uses statistically notable short-term opportunities captured by technical indicators, such as momentum and trending. However, the absence of a solid mathematical foundation for technical analysis has meant that its presence in the academic literature is very limited. Furthermore, during the 1960s, trading rules based on technical indicators were studied and found not to be profitable (Fama & Blume, 1966; Alexander, 1961). It was this work that led notable academics to dismiss technical analysis and support the Efficient Market Hypothesis (Fama, 1970). However, the major problem with the early studies of technical analysis was the ad hoc specifications of the trading rules that suggested the use of data dredging.[1] Despite this, profitable technical trading strategies for inter-day trading in the S&P 500 have been found using genetic algorithms, although the strategies did not fair any better than simple buy and hold strategies (where the asset is bought at the start of the test period and sold at the end) when presented with out of sample data (Allen & Karjalainen, 1999). More recently, there has been a surge in work generating trading strategies by using technical indicators as inputs to machine learning models.

To this end, Artificial Neural Networks (ANNs) were first applied to stock prediction in 1988 when a feed forward network was used to analyse daily stock returns of IBM (White & Diego, 1988). Since then, a plethora of research has attempted to find predictive rules for US (Refenes, Zapranis, & Francis, 1995; Enke & Thawornwong, 2005), Japanese (Kamijo & Tanigawa, 1990) and many other stock prices as well as a multitude of other financial assets (Chen, Leung, & Daouk, 2003; Cheng, Wanger, & Lin, 1996). Although the studies mentioned above indicated that they outperform the relevant benchmarks, many showed that ANNs had some limitations in learning due to the vast noise and complex dimensionality of stock market data.

In addition, Support Vector Machines (SVMs) have been used to forecast stock market movements. The SVM is a non-parametric technique that was developed by Cortes and Vapnik (1995). It has since been used in many studies to predict the daily stock price movements and has been shown to outperform neural networks (Tay & Cao, 2001; Kim, 2003; Huang, Nakamori, & Wang, 2005; Chen & Shih, 2006). A variety of other machine learning approaches including reinforcement learning (O, Lee, & Zhang, 2006), evolutionary bootstrapping (Lebaron, 1998) and principle component analysis (Towers & Burgess, 1999) have been used to forecast financial markets. Although many of these studies indicate that they outperform their benchmarks, we have found that most approaches exhibit large drawdowns[2] in profits, and large and excessive switching behaviour resulting in very high transaction costs. These behaviours have often been attributed to overfitting. As we show in this paper, this problem can be overcome using random forest methods.

2.1. Random forests and financial prediction

Random forests (RFs) are a nonparametric and nonlinear classification and regression algorithm first proposed by Ho (1995) and further developed by Breiman (2001). They have been shown to always converge such that overfitting is not a problem (Breiman, 2001) and, as such, have proved successful in a number of domains including image classification (Bosch, Zisserman, & Munoz, 2007), ecological prediction (Prasad, Iverson, & Liaw, 2006) and microarray data classification (Díaz-Uriarte & Alvarez De Andrés, 2006).

In particular, Creamer and Freund (2004) used this technique to successfully predict the performance of companies and to measure corporate governance risk in Latin American banks. In their study, the performance of random forests was compared to logistic regression and Adaboost, finding that random forests consistently produced superior results. The merits of random forests in financial prediction were also demonstrated by Lariviere and Vandenpoel (2005) who showed that random forest regression could be used for exploring both customer retention and profitability. They analysed a sample of 100,000 customers using data from a large European financial services company finding that random forests techniques provided better prediction results for the validation and test samples compared to linear regression and logistic regression models.

Recently there has been a surge in interest in he use of random forests for stock market prediction. Maragoudakis and Serpanos (2010) used a method called Markov Blanket random forest to make predictions on the direction of stock markets. They report their proposed strategy to outperform a simple buy-and-hold investment strategy by an average of 12.5% to 26% for the initial period and from 16% to 48% for the remainder, as well as outperforming linear regression, SVMs and ANNs. More recently, Qin, Wang, Li, and Ge (2013) used gradient boosted random forests to make predictions on the direction of the Singapore Exchange. Using boosting to weight the individual trees of the forest and a forgetting factor to address market changes, their empirical results showed that their proposed methods were able to generate excess returns compared to a buy-and-hold strategy. Also, Zbikowski and Grzegorzewski (2013) used a novel online-adaptation method to allow random forests to adapt to non-stationary financial time series, while Xu, Li, and Luo (2013) demonstrated the random forest algorithms's ability to select features for trend prediction in stock prices.

[1] Data dredging represents the statistical bias that arises from the misuse of statistics. It implies the deliberate inappropriate use of statistics to uncover misleading relationships in data.

[2] We define drawdowns as the peak-to-trough decline over the duration of a strategy's test period. Drawdowns will be measured as the percentage change from the time a retrenchment begins to when a new high is reached.

The impressive out-of-sample results reported in these studies form the basis of our decision to use random forests as a baseline expert predictor. Although many of the trading models mentioned above are shown to produce an acceptable risk-return behaviour on recent historical data, there is no guarantee that such systems will continue generating the same returns through all market phases in the future without higher level adaptation. Algorithms that are unable to adapt to the changing market conditions will not succeed in the long run. To this end, we propose a novel approach that sees the online training of random forests to produce an ensemble of experts that is able to produce accurate predictions in non stationary financial time series data.

2.2. Financial prediction with ensembles

Over the last few years, the ensemble approach has been applied with great success in expert learning systems such as decision trees (Breiman, 1996) and neural networks (Hansen & Salamon, 1990). Ensembles of expert systems have already been successfully applied in many areas such as face recognition (Gutta & Wechsler, 1996), character recognition (Mao, 1998) and medical diagnosis (Zhou, Jiang, Yang, & Chen, 2002) among others.

In more detail, an ensemble of ANNs was investigated by Abdullah and Ganapathy (2000) as a method for classifying trends of the Kuala Lumpur Stock Exchange composite index. They compared three relatively simplistic methods of combining the classifiers – max, average and median – finding that median produced the higher classification score and mean return per trade. Chen, Yang, and Abraham (2007) used a similar approach of combining the outputs of an ensemble of ANNs but instead used particle swarm optimisation to determine the weight that each predictor was given.

In similar vein, Chun and Park (2005) used an ensemble of case-based-reasoning models to predict the daily movement of the Korean Stock Price Index. They found that choosing the expert with the best historical performance gave the best out-of-sample results, significantly beating a random walk model with $p < 0.01$.

In a particularly extensive study, Tsai, Lin, Yen, and Chen (2011) used classifier ensembles to forecast stock returns. Specifically, they compared the hybrid methods of majority voting and bagging finding that, although multiple classifiers outperformed single ones in terms of prediction accuracy and returns on investment, there was no significant difference between majority voting and bagging. Furthermore, performance using 'homogeneous' ensembles (e.g. an ensemble of neural networks) and 'heterogeneous' ensembles (e.g. an ensemble of neural networks, decision trees and logistic regression) was analysed to find that heterogeneous ensembles offered slightly better performance than homogeneous ones.

With similar goals in mind, Creamer and Freund (2010) used an alternating decision tree (ADT) learning algorithm, which was implemented with Logitboost, to generate buy and sell signals for a multitude of stocks in the S&P 500 index. Additionally, they implemented an on-line learning layer to combine the output of several ADTs and suggests a short or long position. They found that their algorithm was able to generate consistently positive returns and the online layer enabled the algorithm to constantly adapt in line with the market.

More recently, Xiao, Xiao, Lu, and Wang (2013) have demonstrated the power of ensemble in financial market for forecasting with three-stage nonlinear ensemble model that composes three different types of neural-network based models optimised with improved particle swarm optimisation. The show that the flexibility of ensemble approach is key to their ability to capture complex nonlinear relationships.

All of the studies of ensembles mentioned above demonstrate the ability to avoid overfitting compared to single expert systems. Against this background, we develop an automated system for predicting the price return during the seasonal regularities in stock prices that are described in the next section. The system is composed of an ensemble of performance-weighted random forests where new experts are trained on the previously unseen data. These experts are then added to the ensemble in an online fashion to allow the algorithm to adapt to changing market regimes. A risk management layer is implemented to prevent the large drawdowns commonly seen in many systems. This novel technique of online generation of random forests and combining predictions in a recent-performance weighted average draws on the proven capabilities of random forests and ensemble approaches to avoid overfitting.

2.3. Empirical regularities of equity data

Many seasonalities and empirical regularities of financial markets have been documented in the literature with events including turn-of-the month, weekends effects and exchange holidays. In the following we discuss these events in full.

In more detail, turn-of-the-month effects were first documented by Ariel (1987) who found that the return for the latter half of the average month is negative and that positive returns occur in the first part of the month. This result was further explored by Lakonishok and Smidt (1988). Using a 90-year series for the Dow Jones Industrial Average (DJIA), they showed that the returns for the four days around the turn of the month, starting with the last day of the prior month, is 0.473% with the average return for any four-day period being 0.0612%.

The first study of weekend effects in security markets appeared in 1931. In this study, Fields (1931) examined the pattern of the DJIA for the period 1915–1930 to see if the unwillingness of traders to carry their holdings over weekends lead to a liquidation of long accounts and a consequent decline of security prices. He compared the closing price of the DJIA for Saturday with the closing prices on the adjacent Friday and Monday. He found that, in fact, prices tended to increase on Saturdays. Since then, numerous studies have reinforced Fields' findings, confirming that asset prices tend to be lower on Mondays than the preceding Fridays (Cross, 1973; French, 1980; Rogalski, 1984).

In French's investigation of weekend effects he also looked at the price behaviour after exchange holidays, finding no empirical regularities. However, in another very early study, Fields (1934) found that the DJIA showed a high proportion of positive returns one day prior to holidays. These results were confirmed by Ariel (1990) who looked at the returns on the days that surrounded exchange holidays during 1963–1982. It was found that the mean return on the preholidays was significantly greater than other days; a results that has been further supported by others (Cadsby & Ratner, 1992; Lakonishok & Smidt, 1988).

3. Seasonality effects and feature selection

In this section, we first explore the persistence and reliability of the aforementioned seasonality effects. Next, we propose a number of features for capturing seasonality and the state of the market before describing our feature selection method.

3.1. Seasonality

Before exploring the space of possible features for the expert system, we first analyse data from the Deutsche Borse Ag German Stock Index (DAX) over the years 2000 to 2010 to ensure the

Table 1
Table showing the percentage of times that each of the following seasonality effects was observed in the DAX for the period of 2000–2010: turn-of-the-month, exchange holiday, weekend effect. This is compared to the percentage of upward market movements for all days.

Event	Percentage of times trend was observed (%)
Turn-of-month	62.6
Holiday	73.2
Weekend	67.5
All days	51.2

existence and reliability of the regularities described in Section 2.3. We test the hypothesis that these events lead to an average positive or negative return for the index. To do this we measure returns over a particular event, reporting the percentage of times the expected trend was observed from: the third week of a month to the first week of the next month; the day before exchange holiday until the day after; Friday to the following Monday. Results are given in Table 1. We can see that all of the aforementioned regularities are apparent in the DAX data with holiday effect being particularly consistent.

In order to make a seasonality trading strategy more robust, it seems appropriate to investigate whether these regularities are consistent through time or whether they are more pronounced at particular times. For this we have calculated the percentage of time that the index has followed the seasonality effect during each month over the entire dataset the result is shown in Fig. 1. The percentage for each month includes all events (weekend, turn-of-month or exchange holiday) that occurred within that month while events that span two months are included in the month during which the event began.

As Fig. 1 demonstrates, there are significant differences in the consistency of the regularities between months. It is evident that the regularities hold strong during both the first and last few months of each year. However, following a simple seasonality strategy would be highly unprofitable during the middle third of the year.[3] Thus, in order to generate a consistently profitable trading strategy based on seasonality trading, it is important to take into account the month of the year and the type of event in question. To this end, we explore the relative importance of various criteria on predicting the price change over a seasonality event.

3.2. Features

We propose a number of features (see Table B.8) to capture the state of the market, as best we can, at a given time. First, the three seasonality events discussed in Section 3.1 are distinguished: the exchange holidays, turn-of-the-month, and the weekend. Additionally, the open, high low, and close values of the day before the event are explored, as well as a number of technical indicators, with various parameterisations, which are described in Appendix A.

As well as the aforementioned technical indicators we include the current month, the close price of the index to which the stock belongs, and our own risk-based indicator. We take a concept from the value-at-risk (VaR) literature to engineer a feature that signals whether it is likely to be a particularly risky time to trade. The idea of this is to equip the expert with a feature that acts in a self regulatory way, indicating when the market is acting in an unusual way. To do this, we calculate a parametric VaR of a buy-and-hold strategy on the asset we are trading. Specifically, we assume that the daily price changes of an asset are drawn from a Gaussian distribution, the variance of which we estimate with an exponentially weighted moving average:

$$\sigma_t^2 = \Lambda\sigma_{t-1}^2 + (1-\Lambda)r_{g,t}^2 \tag{1}$$

where σ_t^2 is the variance at time t, $r_{g,t}$ is the return of stock g, and Λ is the decay factor. We set Λ to 0.94. Once we have an estimate of the variance, a standard 95% VaR is estimated as 1.96 times the square root of the variance. This value can be interpreted as follows: on any given day there is a 5% chance that the buy-and-hold strategy will lose more that the VaR estimate. However, the standard VaR does not cope well with the heteroscedasticity inherent in financial returns data. To overcome this, we introduce a VaR scaling factor. Each day that a VaR break occurs (a drawdown exceeding our 95% VaR) we add one to the VaR scaling factor while each day that a VaR break does not occur we multiply the VaR scaling factor by Λ. Thus, the VaR is calculated as follows:

$$\mathrm{VaR}_t = 1.96 v_t \sqrt{\sigma_t^2} \tag{2}$$

where the *VaR scaling factor*, v_t, is calculated as:

$$v_t = \begin{cases} v_{t-1} + 1 & \text{if a VaR break occurred at } t-1 \\ \Lambda v_{t-1} & \text{otherwise} \end{cases} \tag{3}$$

This violent reaction of the VaR scaling factor to any unusual event and the gradual settling down afterwards is what allows our measure to cope with heteroscedasticity.

3.3. Feature selection

In order to select only the most effective features, we eliminate those features that have little or no impact on the performance of the predictive model. To this end, a method of feature importance ranking, first suggested by Breiman (2001), is used to eliminate the least important features.

In particular, to rank the features, a single random forest is trained using the training data. In doing so, the root mean squared error RMSE on the out-of-bag portion of the data[4] is recorded for each tree. This is repeated after randomly permuting the values of each of the features. The difference in error between the permuted and non permuted trees are then averaged and normalised by the standard deviation of the differences. To this end, the importance of feature j, $\mathcal{VI}_j$, is calculated as:

$$\mathcal{VI}_j = \frac{\sum_{\theta=1}^{\Theta}(e_{\theta,j} - e_{\theta,\pi j})}{\Theta \cdot \tilde{\sigma}} \tag{4}$$

where Θ is the total number of trees in the forest, $e_{\theta,j}$ is the RMSE of tree θ before permuting j, $e_{\theta,\pi j}$ is the RMSE of that tree after permutation of j, and $\tilde{\sigma}$ is the standard deviation of the differences between $e_{\theta,j}$ and $e_{\theta,\pi j}$. A feature that produces a large $\mathcal{VI}$ value is considered more important than a feature with a lower $\mathcal{VI}$.

To select the features to be used in our expert system, we propose a backwards elimination method. We do so as it has been shown that such methods provide a stronger feature subset than similar alternatives (Guyon & Elisseeff, 2003). The feature selection algorithm that we use is described fully below:

[3] When referring to a "simple seasonality strategy" we mean always adhering to the following: buy a stock in the third week of the month and sell in the first week of the next, sell stock on Friday and buy it back the proceeding Monday and buy stock the day before an exchange holiday selling it back the day after.

[4] Given that each tree in a random forest is constructed using a unique bootstrap sample of about 60% of the data, the rest of the data can be used to generate unbiased error estimates.

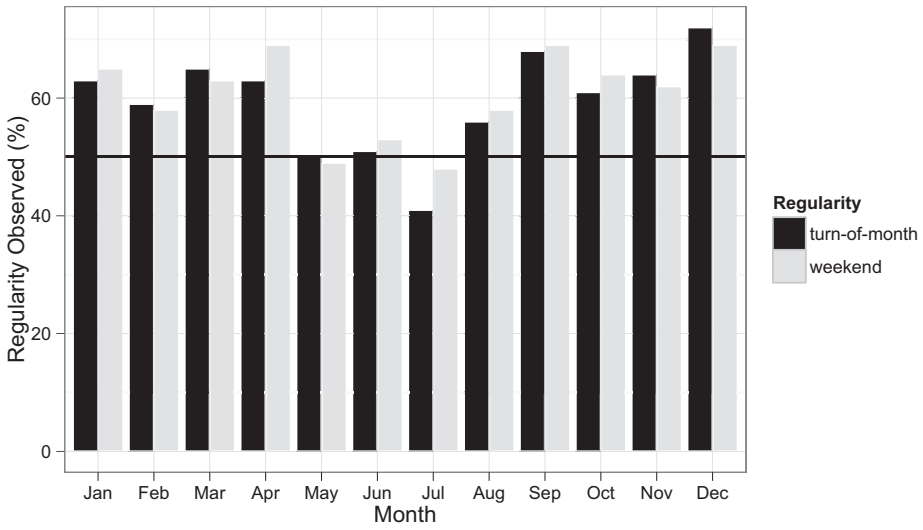

Fig. 1. The difference in the consistency of the weekend effect and the turn-of-month effect between months over the period of 2000–2010.

Algorithm 1.

1: Train a random forrest using the training data with all J features
2: Compute the average RMSE of the model on cross validation CV data
3: Rank the features according to performance as described by Eq. (4)
4: **for** each subset of variables $J_i = J-1,\ J-2,\ldots 1$ **do**
5: Train a new forest using J_i features with the highest $\mathcal{VI}$
6: Compute the average RMSE of model on CV set
7: Rerank the features
8: **end for**
9: Determine which J_i yielded the smallest RMSE

The algorithm above was applied to the training and CV data and a plot of the RMSE at each stage of the elimination is given in Fig. 2. It is evident that there is a initial increase in performance (decrease in RMSE) as the initial weak features are dropped. Following this, there is a rapid decline in performance as features that are essential for prediction are eliminated. The feature selection algorithm yields the optimal set of 23 features shaded grey in table listed in Table B.8 in Appendix B. Note that all continuous valued inputs are taken as logs and normalised with the stock's closing price.

4. Trading model

The trading system discussed in this paper is designed to trade stocks and consists of three layers: random forest prediction, expert weighting and risk management. While each of these layers is discussed in greater detail below, an overview of the system is given in Fig. 3.

4.1. Layer 1: the random forest prediction algorithm

The role of the prediction layer is to generate an expert every d days to form an ensemble of experts that predicts the amount of profit that would be realised from buying a stock the day before a seasonality event and selling it two days after (a seasonality trade). In this paper an expert is represented by a random forest regression algorithm.

A random forest, as the name suggests, is an ensemble of many classification or regression trees designed to produce accurate predictions that do not overfit the data (Breiman, 2001). In this paper we build a forest using a type of decision tree known as regression trees. Regression trees use a tree structure to recursively partition the features space until the subsets of feature-space are tame enough to fit simple models to them. The tree model therefore has two parts: the recursive partition and a simple model for each cell of the partition. Each of the terminal nodes (leaves) of a tree represents a small subset of the feature space, and is attached to a simple model which applies only to that subset. The cell to which a datapoint belongs is identified by starting at the root node of the tree, and answering a sequence of questions about the feature values. The structure of a regression tree is shown in Fig. 4.

In training a random forest, bootstrap samples are drawn from the training data to construct multiple trees, where each tree is grown using a randomised subset of features. Specifically, we produce forests of regression trees using the procedure below, where the training set is defined as $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ and the aim to find a function $f : X \to Y$, where X is the feature space and Y is the output space. Furthermore, let M denote the number of features.

1. Randomly select n observations from D with replacement to form a bootstrap sample.
2. If there are M features, $m \ll M$ features are selected to grow a regression tree such that, at each node, m features are selected at random and the best split (as measured by Gini impurity) on these m is used to split the node.
3. Each tree is grown to the largest extent possible without pruning.

In growing each tree without pruning, and selecting the best split among a random subset at each node, random forests maintain prediction strength while inducing diversity among trees (Breiman, 2001). Furthermore, the random selection of features reduces correlation among the unpruned trees and keeps overall model bias low. With an ensemble of such trees, variance is also reduced.

Now, for any new observation, the random forest produces an overall prediction by taking the average of the predictions of the individual trees in the forest. In the following, this prediction is denoted by $S_{i,t}$, where i is the random forest, and t is the time of the observation. Thus, the prediction $S_{i,t}$ is the output of this layer.

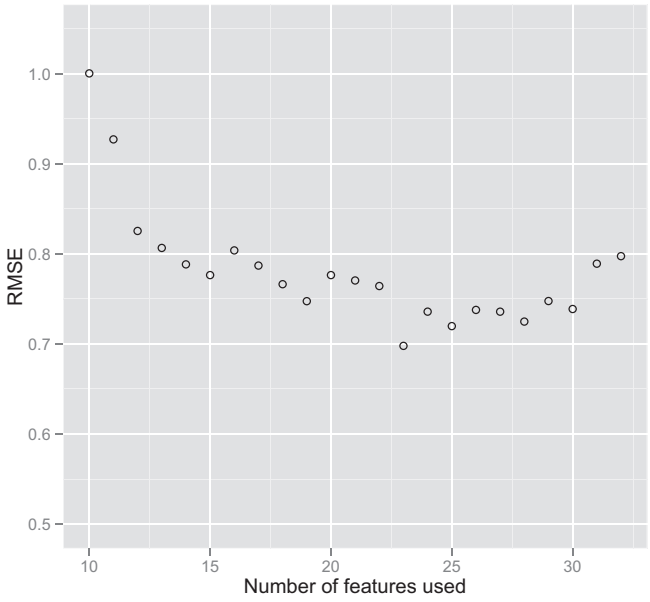

Fig. 2. Plot of RMSE for each round of the feature elimination algorithm. It can be seen that, as features are removed, there is an initial, slight improvement in performance before a rapid decline.

The features given in Table B.8 form the input to the random forest prediction algorithm. In our study, the goal of the prediction layer is to generate a function of these inputs that predicts the dollar profit from a seasonality trade.

4.2. Layer 2: expert weighting: an ensemble of ensembles

The previous section describes the process of generating and predicting with a single (random forest) expert. We now improve the performance of the prediction system by creating multiple random forests and adding an online learning feature. In this paper, a new expert (i.e, random forest) is trained every d days on a moving window of training data to form an ensemble of experts. The outputs of all of the experts are then combined using expert weighting to produce a buy, sell or hold signal before passing the signal through to the risk management layer described in Section 4.3.

The expert weighting algorithm described in this paper is derived from a previous algorithm suggested by Creamer and Freund (2010) that predicts by taking an average of the predictions of all the experts weighted by their training error. Their formula for the exponential weights is derived from a weighted majority algorithm introduced by Littlestone and Warmuth (1989). We explore two variants of an expert weighting system, one proposed by Creamer and Freund (2010) and one introduced in this paper, that both center around an exponential weighting of performance.

In more detail, the historical performance of expert i at time t is denoted by $k_{i,t}$ with more weight given to experts with higher values of $k_{i,t}$. Two methods for measuring this performance are investigated:

1. The original cumulative performance measure:

$$k_{i,t} = \sum_{s=t_{i+1}}^{t} r_{i,s} \tag{5}$$

where $t_1 = 0$, t_i is the time step at which expert i was added to the ensemble and $r_{i,s}$ is the abnormal return on investment for expert i at time s.

2. Our novel exponentially-weighted performance measure:

$$k_{i,t} = \lambda r_{i,t-1} + (1-\lambda)k_{i,t-1} \tag{6}$$

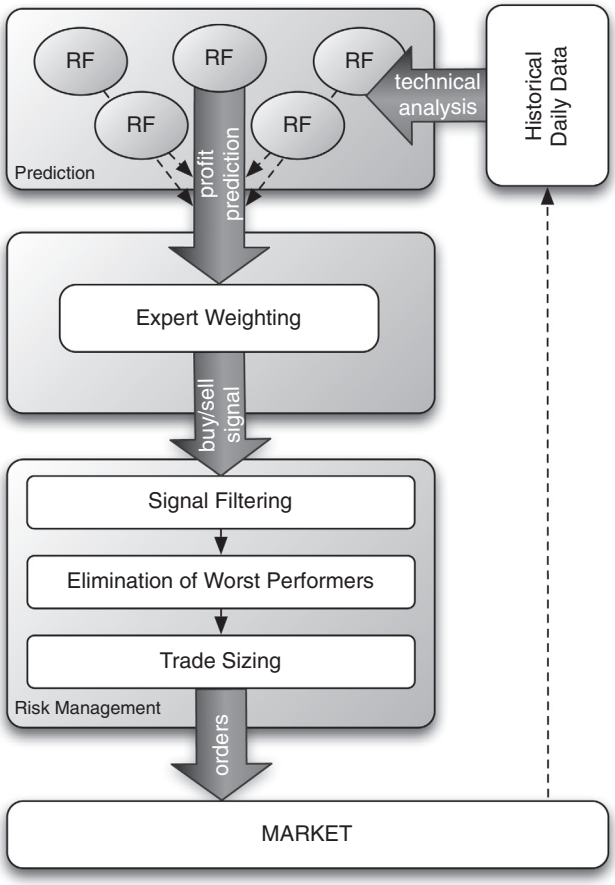

Fig. 3. Diagrammatic representation of the layered workings of the expert trading system.

where $r_{i,t}$ is the abnormal return for expert i at time t and λ is a smoothing parameter that allows us to control the recency weighting of the performance measure.

As mentioned earlier, more weight is given to experts that have a higher $k_{i,t}$ as, in both cases, this represents better historical performance. To this end, we us the exponential weighting algorithm described by Creamer and Freund (2010) to generate the weights for each expert. Specifically, the weight of the first expert is given by:

$$w_{1,t} = \exp\left(\frac{k_{1,t-1}}{\sqrt{t}}\right) \tag{7}$$

The weight of all future experts at time t is:

$$w_{i,t} = I_i \cdot ramp(t-t_i) \cdot \exp\left(\frac{k_{i,t-1}}{\sqrt{t-t_i}}\right) \tag{8}$$

where I_i is the initial weight assigned to a new expert and is the mean of all of the current experts weights; $ramp(t-t_i) = \min\left(\frac{t-t_i}{t_{i+1}-t_i}, 1\right)$ brings the new expert in gradually; and t_{i+1} is the time the next expert is added.

The combined output of the ensemble of experts at time t is then the weighted average of the predictions of each expert at time t:

$$P_{g,t} = \frac{\sum_i S_{i,t} w_{i,t}}{n} \tag{9}$$

where $S_{i,t}$ is the prediction of expert i at time t, n is the current total number of experts and $P_{g,t}$ is a prediction of the profit that would result from buying stock g the day before a seasonality event and selling it two days after.

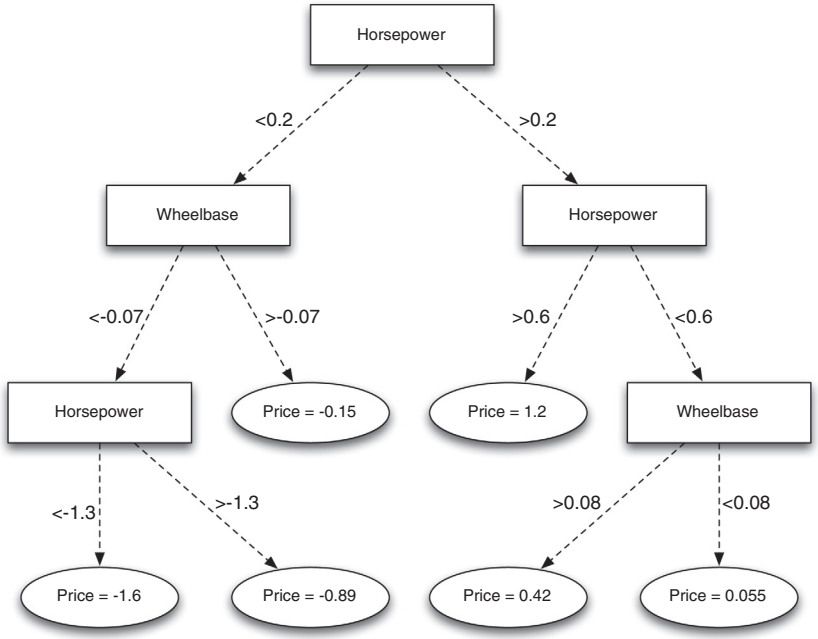

Fig. 4. Example of a regression tree for predicting price of cars built in 1993. Both the target and the features have been standardised to have zero mean and unit variance.

4.3. Layer 3: signal filtering and risk management

Once a potential profit has been predicted, appropriate risk management is of vital importance to prevent large drawdowns. To this end, our algorithm evaluates the prediction, considering market factors, before making an investment decision.

To rule out weak signals we also calculate the difference between the fraction of experts that suggested buying the asset and those that suggested selling as follows:

$$D_t = \text{Buy}_t - \text{Sell}_t \tag{10}$$

where:

$$\text{Buy}_t = \frac{\sum_{i:S_t^i>0} w_t^i}{\sum_i w_t^i} \tag{11}$$

and $\text{Sell}_t = 1 - \text{Buy}_t$. As the value of D_t is normalised between fully short (-1) and long $(+1)$, weak signals may be eliminated by ruling out predictions where $|D_t|$ falls below a threshold α_0.

Also, it cannot be assumed that predictions will be accurate for all assets at all times. Thus, we follow Creamer and Freund (2010) and monitor the performance of the algorithm on each stock g using the maximum drawdown:

$$d_{g,t} = \max(r_{g,t_x} - r_{g,t_y} | t_0 \leqslant t_x \leqslant t_y \leqslant t) \tag{12}$$

where r_{g,t_x} and r_{g,t_y} are the return of stock g from time t_0 to t_x and t_y respectively. The maximum drawdown represents the maximum possible loss in a certain period of time and the vector of maximum drawdowns for all stocks is represented as $\mathbf{d}_t$. If $d_{g,t} < \lambda_1$ for thirty days the system hold its current position and if $d_{g,t-1} = \min(\mathbf{d}_{t-1})$, the system liquidates the position in that stock.

The final output of this layer, $X_{g,t}$, is a buy or sell order whose size is determined by the magnitude of the profit prediction, $P_{g,t}$. Specifically, more is invested in stocks where a large price movement is forecast:

$$X_{g,t} = \begin{cases} -1 \cdot q_{g,t} & \text{if } (P_{g,t} > 0 \wedge q_{g,t} < 0) \vee (P_{g,t} < 0 \wedge q_{g,t} > 0) \\ C_t * \frac{P_{g,t}}{\sum_g |P_{g,t}|} & \text{otherwise} \end{cases} \tag{13}$$

where $q_{g,t}$ is the size of the current position in stock g and C_t is the total dollar wealth at time t. Thus, if the order size, $X_{g,t}$ is positive, this refers to a buy order; if this is negative, it refers to a sell order.

5. Experiments and empirical results

The dataset used in these experiments involves daily open, close, high, low and volume (OCHLV) from 30 stocks from the DAX stock index over the period of 01/01/2000–01/01/2013. Before conducting experiments, the data set is split into training, CV and test sets as follows: 2000–2008 training, 2008–2010 CV and 2010–2012 test. It should be noted that no parameters are tuned using the test set. This set is used only to report the performance of the model found to perform best on the CV data.

A single experimental run proceeds as follows:

- 15 stocks are randomly selected from the DAX with equal probability.
- Features are generated from the data.
- The model iterates through the training set, generating three new random forest experts every 50 days each with moving windows of size 50, 100 and 200 days.
- The prediction performance of each expert is monitored and the performance weights are updated after every event according to Eq. (8).
- When the model reaches the CV data it generates a performance weighted ensemble prediction for each event using Eq. (9). If this prediction passes through the risk management layer then the model takes a long, short or hold position.
- At the end of the CV period the performance of the model is reported as cumulative abnormal return and Sharpe ratio. Simulated transaction costs are incorporated into each trade at a value of We tested our results with transaction costs of $0.003 per stock. This value is in line with the current NASDAQ pricing policy.

We use the experimental procedure outlined above to find the optimal parameterisation of our model, as well as the relevant benchmarks, using only the training and CV data. There are only two parameters to consider for the random forests themselves:

the number of trees in each forest (*n_trees*) and the size of the random subset of features to consider when splitting a node (m). For the former, the larger the better. We use $n\_trees = 200$ as we find no improvement in performance beyond this value. Generally, lower values of m lead to a reduction of variance but also an increase in bias. We thus use cross validation to find optimal values for this parameter.

As well as finding optimal values for m, we use cross validation to explore values of λ in Eq. (6). Recall that this value controls the recency bias of the performance metric for each of the experts. To asses the performance of the aforementioned variables we perform a two dimensional grid-search of the parameter space. For each point in parameter space we perform 100 experiments as defined above to generate an average annualised return.

5.1. Results

The results of the CV parameter grid-search for our performance-weighted random forest ensemble model are illustrated in Fig. 5. It can be seen that the best parametrisation for the recency-weighted model lies in the vicinity of $\lambda = 0.90 - 0.95$ and $m = 18$. This value for m is equivalent to using all possible features for each node split, and it has been shown that such a parametrisation is often optimal for random forest regression (Liaw & Wiener, 2002). The optimal value of λ is also interesting as it is a very commonly used value for exponential moving averages and is also used by the financial risk management firm RiskMetrics™ for their moving average estimation of volatility.

Now that we have tuned the parameters, we compare our algorithms with the state-of-the-art using the test data. We start by exploring the merits of the random forest algorithm compared with other state of the art regression algorithms. To this end, Table 2 shows the performance of a number of regression algorithms in predicting the return of stocks over a seasonality event given the inputs in Table B.8. Experiments were conducted as described above and results are averaged over 50 runs for each model. We measure the mean absolute percentage error (MAPE) and the RMSE. For both the MAPE and RMSE, a lower output means better forecasting accuracy of the model.

Next, the merit of ensembles and associated methodologies is explored. To this end, Table 3 shows the prediction performance and profitability of a solo random forest as well as a number of models that involve ensembles trained using the online method described at the beginning of this section. The results represent the average performance of each model over 50 experimental runs.

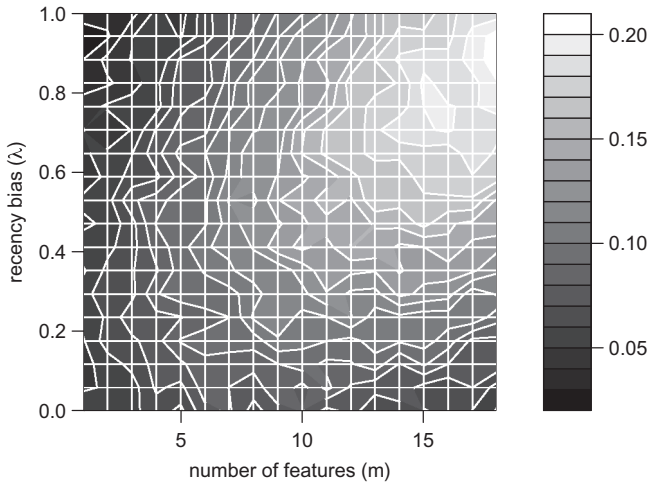

Fig. 5. Heat map showing results of the cross validation gridsearch for the performance-weighted random forest ensemble.

Table 2
A comparison of the performance of linear regression, regression trees, multi-layer feed forward neural network (MLNN), support vector regression (SVR) and random forest algorithms in predicting the return of a stock over a seasonality event.

Model	MAPE	RMSE
Training phase		
Linear regression	0.018*	0.022*
Regression tree	0.015*	0.017*
MLNN	0.008	0.015*
SVR	0.008	0.013
Random forest	0.009	0.013
CV phase		
Linear regression	0.020*	0.023*
Regression tree	0.019*	0.020*
MLNN	0.022*	0.023*
SVR	0.026*	0.037*
Random forest	0.011	0.015
Test phase		
Linear regression	0.021*	0.024*
Regression tree	0.019*	0.019*
MLNN	0.021*	0.024*
SVR	0.030*	0.041*
Random forest	0.010	0.013

* It denotes a statistical significance compared to the random forest model of $p < 0.02$.

Table 3
A comparison of model performance metrics. A description of each model is given in Table 5.

Model	Annualised return	Sharpe ratio	MD	MAPE	RMSE
Training phase					
Buy & hold	0.023*	0.96*	−22.2*	–	–
Basic seasonality	0.059*	0.67*	−18.5*	–	–
Solo RF	0.022*	0.98*	−22.4*	0.009*	0.013*
RF ensemble	0.11*	1.28*	−16.0*	0.007*	0.009*
CW RF ensemble	0.14*	1.60*	−13.2*	0.004*	0.007*
RW RF ensemble	0.17	1.65	−12.8	0.003	0.005
CV phase					
Buy & hold	−0.13*	−0.63*	−30.1*	–	–
Basic seasonality	0.061*	0.68*	−17.6*	–	–
Solo RF	0.0128*	0.96*	−21.4*	0.011*	0.015*
RF ensemble	0.08*	1.24*	−18.3*	0.008*	0.010*
CW RF ensemble	0.13*	1.52	−13.9	0.004	0.007
RW RF ensemble	0.14	1.60	−12.9	0.004	0.007
Test phase					
Buy & hold	−0.012*	−0.82* 8	−27.9*	–	–
Basic seasonality	0.057*	0.68*	−18.2*	–	–
Solo RF	0.01*	0.93*	−23.5*	0.01*	0.013*
RF ensemble	0.09*	1.27*	−16.3*	0.006*	0.010*
CW RF ensemble	0.12*	1.44*	−14.1	0.004	0.008*
RW RF ensemble	0.15	1.64	−12.3	0.004	0.006

* It denotes a statistical significance compared to the RW RF ensemble model of $p < 0.02$.

Table 4
A comparison of RW RF ensemble performance when trading only over seasonal events vs. trading every day.

Model	Annualised return	Sharpe ratio	MD	MAPE	RMSE
Training phase					
Daily	0.13	1.42	−14.5	0.005	0.008
Seasonal	0.16	1.66	−12.8	0.003	0.006
CV phase					
Daily	0.14	1.47	−18.9	0.006	0.009
Seasonal	0.14	1.59	−13.0	0.004	0.008
Test phase					
Daily	0.11	1.36	−23.0	0.006	0.010
Seasonal	0.15	1.63	−12.7	0.003	0.007

Table 5
A description of the models whose results are described in Table 3.

Model	Description
Buy & hold	A long position in the stocks is taken at the beginning of the test period and closed out at the end
Basic seasonality	Positions are taken in line with the empirical regularities discussed in Section 2.3, e.g., always take a short position in a stock over a weekend
Solo RF	A single random forest with 200 trees and $m = 17$ is used to predict the return of a stock over a seasonality event
RF ensemble	An ensemble of random forests is generated in an online fashion by training a new random forest every 50 days. A prediction for the return of a stock is generated by taking an average of the predictions of all of the random forests in the ensemble
PW RF ensemble	Performance-weighted random Forest ensemble. As above except that the performance of each random forest in the ensemble is monitored as its cumulative return, as described in Eq. (7), such that the final prediction of the ensemble is a performance-weighted average of the experts
RW RF ensemble	Recency-weighted random forest ensemble. As above, except that, instead of the cumulative return, an exponential moving average of returns, as in Eq. (8), for each expert is used for the performance weighted-prediction

Table 6
Description of the performance measures reported in Table 3.

Statistical measure	Description
Annualised return	$R_A = \left(\prod_{i=1}^{n}(1+r_i)\right)^{\frac{1}{n}} - 1$
Sharpe ratio	$S = \frac{E[r_m - r_b]}{\sigma}$
Maximum drawdown	$\mathrm{MD} = \max_{\tau\in(0,T)}\left[\max_{\tau\in(t,\tau)} X_t - X_\tau\right]$
Mean abs. % error	$\mathrm{MAPE} = \frac{1}{n}\sum_{i=0}^{n-1}\left\lvert\frac{r_s - \hat{r}_s}{r_s}\right\rvert$
RMSE	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(r_s - \hat{r}_s)^2}$

Where r_i is the return in year i; r_m is the model's return; r_b is return of the DAX (the benchmark); $X(t$ is the return at time t; r_s is the return of stock s over a seasonality event and $\hat{r}_s$ is the predicted return over that event.

A description of the various models is given in Table 5. For the max drawdown (MD), MAPE and RMSE, a lower output means a better the forecasting accuracy for the model. With the annualised return and Sharpe ratio, a higher output is better. A description of all statistical measures is given in Table 6.

We note that our recency-weighted random forest ensemble (RW RF ensemble) outperformed all other models in terms of both profitability and prediction accuracy. It produces slightly higher annualised return and Sharpe ratio than the ensemble that is weighted using cumulative returns (CW RF ensemble) and almost 70% higher returns than an equally weighted random forest ensemble (RF ensemble), the third best performing model. A clear advantage of online ensemble generation can be seen when comparing the results of a single random forest (Solo RF) to the RF ensemble. Introducing online training of experts makes the difference between barely breaking even, with annualised returns of 0.01, and generating a good return (0.09) with a strong Sharpe ratio (1.27).

To investigate the effect of focussing only on seasonality events, we compare the performance of the RW RF ensemble model over seasonality events only (as in the experiments reported in Table 3) with its performance when predicting and trading every day that the markets are open. The results of the comparison are shown in Table 4 where it can be seen that the RW RF ensemble model is more profitable, has greater prediction accuracy and produces small drawdowns when used only during seasonality events.

6. Concluding remarks

In this paper we apply, for the first time, performance weighted ensembles of random forests to the prediction of price returns during known seasonality events with stocks from the DAX. This approach consists of three layers: expert generation, expert weighting, and risk management. In more detail, in the first layer, random forests are continually generated and make predictions about the magnitude of stock price changes. Then, the expert weighting layer generates an overall prediction by averaging the prediction of all forests weighted based on their recent performance. Finally, the risk management layer rules out weak signals and liquidates positions in stocks that are proving difficult to predict. This approach is then benchmarked against constant-weight random forests, a solo random forest, a naïve seasonality strategy and a buy-and-hold strategy. The models are trained during a period from 2000–2008, cross-validated from 2008–2010 and tested out-of-sample from 2010–2012. We also explore and compare two variants for expert weighting: cumulative performance of an expert and recency weighted performance.

In out-of-sample trading simulations, both forms of performance-weighted random forest ensembles outperform all other models in terms of both prediction accuracy and profitability. At the same time, we found that, using a recency weighting of experts' performance, as opposed to the cumulative performance, resulted in over a 10% improvement in risk adjusted return, as well as decreased drawdowns and prediction error. This demonstrates that the ability of the online-generated ensemble to capture different phases of the market allow both approaches of random forest ensemble to excel in forecasting compared with static ensemble approaches. This is because using a moving average of performance to compare experts, allows the prediction algorithm to adjust more swiftly to changing market conditions and hence improve prediction accuracy and profitability.

In addition to the increased performance, the constant training of new experts for the performance weighted random forest ensemble approach has a number of other advantages. First of all, new experts can be added rapidly with minimal training time, a very desirable quality for a quantitative trading application. Secondly, each random forest, and thus the entire ensemble, exists as a human readable decision tree. Thus, our strategy is not a 'black box' and, as such, analysis and simulation can give managers and regulators insight into the risks involved in such a trading strategy. Specifically, the human readable nature of the random forest ensemble ensures that, at any given time, a risk manager is able to ascertain exactly what the system will do and what decisions it will make.

In future work we intend to evaluate the ensemble approach, where experts are trained using techniques such as neural networks, support vector machines and genetic algorithms. Although, such techniques were shown to produce inferior results compared to the random forests, it is possible that, when combined in an online ensemble fashion, they may capture some of the more subtle states of the market than the less tightly fitting random forests. Also, a potential direction for the future is the application of recent-performance-weighted ensemble to the estimation of order impact in optimal order execution.

Appendix A. Technical indicators

See Table A.7.

Table A.7
A description of the technical indicators with parameters shown in parentheses.

Indicator	Description	Calculation
$\text{EMA}_t^c(n)$	Exponential moving average of the last n observations of series p^c	$\text{EMA}(p^c,n)=\lambda\sum_{s=0}^{\infty}(1-\lambda)^s p_{t-s}^c\lambda=\frac{2}{n+1}$ where $n=10,16$ and 22
$\text{SMA}_t^c(n)$	Simple moving average of the last n observations of p^c	$\text{SMA}_t(p^c,n)=\frac{1}{n}\sum_{s=0}^{n-1}p_t^c-s$ where $n=10,16$ and 22
$\text{Boll}_t^u(n,s)$ and $\text{Boll}_t^d(n,s)$	The Bollinger bands are calculated as a function of s standard deviations from the reference $\text{SMA}_t^c(n)$. When the price crosses above the upper Bollinger band, it is a sign that the market is overbought and vice versa	$\text{Boll}_t^u(n,s)=\text{SMA}_t^c(n)+s\sigma_t^2(n)$ $\text{Boll}_t^d(n,s)=\text{SMA}_t^c(n)-s\sigma_t^2(n)$ where $s=2,n=20,26$ and 32
Momentum indicators		
$\text{MOM}_t(n)$	Momentum represents the price change over the last n periods. If it is above (below) zero, it indicates the market is trending up (down)	$p_t^c-p_{t-n}^c$ where $n=12,18$ and 24
$\text{ACC}_t(n)$	Acceleration represents the change in momentum	$\text{MOM}_t(n)-\text{MOM}_{t-1}(n)$ where $n=12,18$ and 24
$\text{ROC}_t(n)$	The rate of change of a time series p_t^c	$100.\frac{p_t^c-p_{t-n}^c}{p_{t-n}^c}$ where $n=10,16$ and 22
$\text{MACD}_t(s,f)$	The moving average convergence divergence is the difference between two moving average of varying periods (s,f)	$\text{EMA}_t(p^c,s)-\text{EMA}_t(p^c,f)$ where f = 12 and $s=18,24$ and 30
$\text{MACDS}_t(s,f,n)$	The MACD signal is the moving average of the $\text{MACD}_t(s,f)$ of the last n periods. Usually a buy signal is generated when the $\text{MACD}_t(s,f)$ crosses the signal line	$\text{EMA}_t(\text{MACD}_t(s,f),n)$ where $f=12,n=9$ and $s=18,24$ and 30
$\text{RSI}_t(n)$	The relative strength index compares the days that a price finishes up with those that it finishes down. Normally a buy signal occurs when $\text{RSI}_t(n)$ crosses below a lower band of 30 and a sell signal occurs when it crosses an upper band of 70	$100-100/\left(1+\frac{\text{SMA}_t(p_n^{up},n_1)}{\text{SMA}_t(p_n^{dn},n_1)}\right)$ where $n=8,14$ and 20
$\text{FAST\%K}_t(n)$	The fast stochastic K represents a percent measure of the last close price in relation to the highest high and the lowest low of the last n periods	$\frac{p_t^{uc}-\min(p_n^l)}{\max(p_n^h)-\min(p_n^l)}$ where $n=12,18$ and 24
$\text{FAST\%D}_t(n)$	Moving average of the $\text{FAST\%K}_t(n)$	$\text{SMA}_t(\text{FAST\%K}_t(n),3)$
$\text{SLOW\%K}_t(n)$	Moving average of the $\text{FAST\%D}_t(n)$	$\text{SMA}_t(\text{FAST\%D}_t(n),3)$
$\text{SLOW\%D}_t(n)$	Moving average of the $\text{SLOW\%K}_t(n)$	$\text{SMA}_t(\text{SLOW\%K}_t(n),3)$
Volatility indicators		
$\text{CHV}_t(n,n_1)$	The Chaikin volatility evaluates the breadth of the range between high and low prices. It also calculates the ROC of the moving average of the difference between the high and low prices. Often a very fast increase (decrease) of this indicator is a signal that the bottom (top) of the market is near	$\frac{\text{EMA}_t(p^h-p^l,n)}{\text{EMA}_{t-n_1}(p^h-p^l,n)}-1$ where $n_1=10$
Volume indicators		
ADL_t	The accumulation/distribution line was also developed by Chaikin. It is calculated using the close location value (CLV). This indicator compares the close with the range of prices from the same period. A positive value indicates a buying pressure and a negative the opposite	$\sum_{t=1}^{n}\text{CLV}_t.\text{VOL}_t$ where $CLV_t=\frac{2p^u c_t-p_t^l-p_t^h}{p_t^h-p_t^l}$
OBV_t	The on balance volume evaluates the impact of positive and negative volume flows. It adds the volume when the close has increased and subtracts it when the close has decreased	if $p_t^c>p_{t-1}^c$: then $\text{OBV}_t=\text{OBV}_{t-1}+\text{VOL}_t$ if $p_t^c<p_{t-1}^c$: then $\text{OBV}_t=\text{OBV}_{t-1}-\text{VOL}_t$
CHO_t	The Chaikin oscillator is the MACD of the ADL. It represents the difference between a short and long $\text{EMA}_t(\text{ADL},n)$. This indicator is to be interpreted similarly to the MACD	$\text{EMA}_t(\text{ADL},n_1)-\text{EMA}_t(\text{ADL},n_2)$ where $n_1=3$, and $n_2=10$

Table B.8
A list of all features considered for input to the prediction model. Those shaded grey are used in the experiments reported in Section 5.

Features	Variants
Month	
Closing price at $t-1$	
Highest price on day $t-1$	
Lowest price on day $t-1$	
Volume traded on day $t-1$	
Index closing price at $t-1$	
Simple moving average of closing prices	
Exponential moving average of closing prices	$n=10,16$ and 22
Bollinger bands	
Momentum	
Acceleration	$n=24$
Rate of change	$n=10$ and 16
MACD	$s=24$ and 30
MACD signal	$s=24$
Relative strength index	$n=14$ and 20
FAST%K	$n=24$
FAST%D	
SLOW%K	
SLOW%D	
Chaikin volatility	
Accumulation distribution line	
On balance volume	
Chaikin Oscillator	
VaR-breaks measure (see Eq. (2))	

Appendix B. Inputs

See Table B.8.

References

Abdullah, M., & Ganapathy, V. (2000). Neural network ensemble for financial trend prediction. In *TENCON Proceedings IEEE* (pp. 157–161). IEEE.

Alexander, S. S. (1961). Price movements in speculative markets: Trends or random walks. *Industrial Management Review, 2*, 7–26.

Allen, F., & Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. *Journal of Financial Economics, 51*, 245–271.

Ariel, R. A. (1987). A monthly effect in stock returns. *Journal of Financial Economics, 18*, 161–174.

Ariel, R. A. (1990). High stock returns before holidays: Existence and evidence on possible causes. *Journal of Finance, 45*, 1611–1626.

Bosch, A., Zisserman, A., & Munoz, X. (2007). Image classification using random forests and ferns. In *IEEE 11th international conference on computer vision* (pp. 1–8). IEEE.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*, 123–140.

Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32.

Cadsby, C. B., & Ratner, M. (1992). Turn-of-month and pre-holiday effects on stock returns: Some international evidence. *Journal of Banking & Finance, 16*, 497–509.

Chen, A. S., Leung, M. T., & Daouk, H. (2003). Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan Stock Index. *Computers & Operations Research, 30*, 901–923.

Chen, W. h., & Shih, J. y. (2006). Comparison of support vector machines and back propagation neural networks in forecasting the six major Asian Stock Markets. *European Journal Of Operational Research, 1*, 49–67.

Chen, Y., Yang, B., & Abraham, A. (2007). Flexible neural trees ensemble for stock index modeling. *Neurocomputing, 70*, 697–703.

Cheng, W., Wanger, L., & Lin, C. (1996). Forecasting the 30-year US treasury bond with a system of neural networks. *Journal of Computational Intelligence in Finance, 4*, 10–16.
Chun, S. H., & Park, Y. J. (2005). Dynamic adaptive ensemble case-based reasoning: Application to stock market prediction. *Expert Systems with Applications, 28*, 435–443.
Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*, 273–297.
Creamer, G., & Freund, Y. (2004). *Predicting performance and quantifying corporate governance risk for latin american ADRs and banks*. *Financial engineering and applications*. Cambridge: MIT.
Creamer, G., & Freund, Y. (2010). Automated trading with boosting and expert weighting. *Quantitative Finance, 4*, 401–420.
Cross, F. (1973). The behavior of stock prices on fridays and mondays. *Financial Analysts Journal, 29*, 67–69.
Díaz-Uriarte, R., & Alvarez De Andrés, S. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics, 7*, 3.
Enke, D., & Thawornwong, S. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications, 29*, 927–940.
Fama, E. F. (1970). Efficient capital markets: A review of theory and emperical work. *Journal of Finance, 25*, 383–417.
Fama, E. F., & Blume, M. E. (1966). Filter rules and stock-market trading. *Journal of Business, 39*, 226–241.
Fields, M. J. (1931). Stock prices: A problem in verification. *The Journal of Business of the University of Chicago, 4*, 415–418.
Fields, M. J. (1934). Security prices and stock exchange holidays in relation to short selling. *The Journal of Business of the University of Chicago, 7*, 328–338.
French, K. R. (1980). Stock returns and the weekend effect. *Journal of Financial Economics, 8*, 55–69.
Gutta, S., & Wechsler, H. (1996). Face recognition using hybrid classifier systems. In *IEEE international conference on neural networks* (pp. 1017–1022). IEEE.
Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.
Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*, 993–1001.
Ho, T. K. (1995). Random decision forests. In *Proceedings of the third international conference on document analysis and recognition* (pp. 278–282).
Huang, W., Nakamori, Y., & Wang, S. Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research, 32*, 2513–2522.
Kamijo, K., & Tanigawa, T. (1990). Stock price pattern recognition – a recurrent neural network approach. In *International joint conference on neural networks* (pp. 215–221). IEEE.
Kim, K. j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing, 55*, 307–319.
Lakonishok, J., & Smidt, S. (1988). Are seasonal anomalies real? A ninety-year perspective. *Review of Financial Studies, 1*, 403–425.
Lariviere, B., & Vandenpoel, D. (2005). Predicting customer retention and profitability by using random forests and regression forests techniques. *Expert Systems with Applications, 29*, 472–484.
Lebaron, B. (1998). An evolutionary bootstrap method for selecting dynamic trading strategies. Social Systems Research Institute, University of Wisconsin Norwell, MA.
Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news, 2*, 18–22.
Littlestone, N., & Warmuth, M. K. (1989). The weighted majority algorithm. In *30th Annual symposium on foundations of computer science* (pp. 256–261).
Mao, J. (1998). A case study on bagging, boosting and basic ensembles of neural networks for OCR. In *The 1998 IEEE international joint conference on neural networks proceedings, IEEE world congress on computational intelligence* (pp. 1828–1833). IEEE.
Maragoudakis, M., & Serpanos, D. (2010). Towards stock market data mining using enriched random forests from textual resources and technical indicators. In H. Papadopoulos, A. Andreou, & M. Bramer (Eds.), *Artificial intelligence applications and innovations. IFIP advances in information and communication technology* (Vol. 339, pp. 278–286). Springer.
Nuij, W., Milea, V., Hogenboom, F., Frasincar, F., & Kaymak, U. (2013). An automated framework for incorporating news into stock trading strategies. *IEEE Transactions on Knowledge and Data Engineering*.
O, J. M., Lee, J. W., & Zhang, B. T. (2006). Adaptive stock trading with dynamic asset allocation using reinforcement learning. *Information Sciences, 176*, 2121–2147.
Prasad, A. M., Iverson, L. R., & Liaw, A. (2006). Newer classification and regression tree techniques: Bagging and random forests for ecological prediction. *Ecosystems, 9*, 181–199.
Qin, Q., Wang, Q. G., Li, J., & Ge, S. S. (2013). Linear and nonlinear trading models with gradient boosted random forests and application to Singapore stock market. *Journal of Intelligent Learning Systems and Applications, 5*, 1–10.
Refenes, A. P., Zapranis, A., & Francis, G. (1995). Modelling stock returns in the framework of APT: A comparative study with regression models. In A. P. Refenes (Ed.), *Neural networks in the capital markets* (pp. 101–125). John Wiley and Sons.
Rogalski, R. J. (1984). New findings regarding day-of-the-week returns over trading and non-trading periods: A note. *The Journal of Finance, 39*, 1603–1614.
Tay, F. E. H., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega, 29*, 309–317.
Towers, N., & Burgess, A. N. (1999). Implementing trading strategies for forecasting models. In *Proceedings computational finance*. The MIT Press.
Tsai, C. F., Lin, Y. C., Yen, D. C., & Chen, Y. M. (2011). Predicting stock returns by classifier ensembles. *Applied Soft Computing, 11*, 2452–2459.
White, H., & Diego, S. (1988). Economic prediction using neural networks: The case of IBM daily stock returns. In *IEEE international conference on neural networks* (pp. 451–458). IEEE.
Xiao, Y., Xiao, J., Lu, F., & Wang, S. (2013). Ensemble ANNs-PSO-GA approach for day-ahead stock e-exchange prices forecasting. *International Journal of Computational Intelligence Systems, 6*, 96–114.
Xu, Y., Li, Z., & Luo, L. (2013). A study on feature selection for trend prediction of stock trading price. In *2013 International Conference on Computational and Information Sciences* (pp. 579–582).
Zbikowski, K., & Grzegorzewski, P. (2013). Stock trading with random forests, trend detection tests and force index volume indicators. *Artificial intelligence and soft computing* (Vol. I, pp. 441–452). Springer.
Zhou, Z. H., Jiang, Y., Yang, Y. B., & Chen, S. F. (2002). Lung cancer cell identification based on artificial neural network ensembles. *Artificial Intelligence in Medicine, 24*, 25–36.