MASTER'S THESIS

Lauri Nevasalmi

Forecasting multinomial stock returns using machine learning methods

Tampere University
Faculty of Information Technology and Communication Sciences
May 2019

Tampere University

Faculty of Information Technology and Communication Sciences

NEVASALMI, LAURI: Forecasting multinomial stock returns using machine learning methods

Master's thesis, 43 p., 2 app. p.

Computational Big Data Analytics

May 2019

Abstract

In this thesis, the daily returns of the S&P 500 stock market index are predicted using a variety of different machine learning methods. We propose a new multinomial classification approach to forecasting stock returns. The multinomial approach can isolate the noisy fluctuation around zero and allows us to focus on predicting the more informative large absolute returns. Our in-sample and out-of-sample forecasting results indicate significant return predictability from a statistical point of view. Moreover, all the machine learning methods considered outperform the benchmark buy-and-hold strategy in a real-life trading simulation. The gradient boosting machine is the top-performer in terms of both the statistical and economic evaluation criteria.

Contents

1 Introduction

Forecasting stock returns has attracted a tremendous amount of interest ever since the introduction of computers to economic forecasting. Kendall (1953) was among the first to reach the conclusion of no predictability in stock prices. Later Fama (1970) stated in the famous efficient market hypothesis that abnormal returns should not be possible to make by using historical data. But has the ever-growing amount of information and computational power in recent decades changed this relationship? State of the art machine learning methods, which can handle large amounts of information and discover complex relationships in data, provide further insight if profitable trading strategies can be discovered using past information.

The predictability of stock returns is a controversial subject. In a comprehensive study Welch and Goyal (2008) argue that the predictability found for the level of stock returns in the previous literature is time inconsistent and does not hold when new data is introduced. More recent work by Neely, Rapach, Tu and Zhou (2014), among others, challenge the view of Welch and Goyal (2008) by reporting statistically significant predictability using more sophisticated forecasting methods.

Instead of the actual level of return another strand of literature focuses on predicting the binary sign of stock returns (i.e. directional predictability of stock returns). Leung, Daouk and Chen (2000) provide empirical results in favor of using the binary response variable instead of the actual level. Other studies reporting statistically significant predictability using monthly stock returns are for example Nyberg (2011) and Nyberg & Pönkä (2016). Christoffersen and Diebold (2006) show theoretically that sign predictability may exist even without the assumption of mean-predictability. Although the majority of the previous literature concerns predicting monthly returns, some more recent studies have reported predictability using daily returns as well (see e.g., Skabar, 2013; Fiévet & Sornette, 2018). The main objective in this thesis is to predict daily stock returns of the U.S. stock market (more specifically S&P 500 index returns) using different machine learning methods.

Directional prediction of stock returns is based on forecasting whether returns are greater than some pre-specified threshold. Previous research mainly focuses on sign prediction, where this treshold is equal to zero, but some other alternatives have also been considered. Linton and Whang (2007) use the estimated unconditional quantiles of the return as a threshold. Chung and Hong (2007) express the threshold as multiples of the estimated standard deviation when forecasting the direction of exchange rates. Both studies find evidence of directional predictability in asset returns using different statistical testing procedures.

Directional prediction of stock returns has a close connection to the market timing models considered by Merton (1981) and Pesaran & Timmermann (1995). Directional prediction of stock returns leads to simple binary trading strategies which can be used to assess the economic significance of the forecasting ability. Predicting the sign of stock returns involves a large amount of asset allocation decisions and the costs related to these transactions can be problematic when compared to the benchmark buy-and-hold strategy. This problem is even more alleviated with daily data (Becker & Leschinski, 2018).

By considering two different thresholds instead of just one the directional prediction problem becomes multinomial. The signal-to-noise ratio in stock returns is fairly low, especially with daily data (Becker & Leschinski, 2018). Chung and Hong (2007) argue that the informational content of large absolute returns may be more valuable whereas small returns are merely noise. It is also noted that the co-movement of individual stocks with the market portfolio is stronger with large absolute returns (see e.g., Longin & Solnik, 2001; Ang & Chen, 2002; Hong, Tu & Zhou, 2007). The multinomial response allows us to isolate some of the noise and put more emphasis on predicting the large absolute returns. The multinomial directional prediction also enables a richer set of possible trading strategies. For example one could choose between buying, holding and selling stocks instead of the binary buy or sell decision. To the best of our knowledge multinomial stock returns have not been utilized in previous economic research.

Our results confirm the previous findings of sign predictability in stock returns. All the machine learning methods considered in this thesis produce multinomial classification significant from both the statistical and economical point of view. Each method is able to outperform the benchmark buy-and-hold strategy in a real-life trading simulation when trading costs are taken into account. Among the machine learning methods considered an ensemble method called gradient boosting is the top-performer in terms of both the classification accuracy and the profits from a real-life trading simulation.

The results also show how the predictability of large absolute returns tend to cluster around certain periods of time. This is in line with the findings of Krauss, Do & Huck (2017) and Fiévet & Sornette (2018) who notice increased predictability during high market turmoil. A closely related observation often reported in the financial literature is the higher return predictability during recession periods (see e.g., Henkel, Martin & Nardari, 2011; Cujean & Hasler, 2017). Events such as the financial crisis or the European debt crisis involve high volatility in the stock markets

but also highly profitable trading opporturnities. Our results show that volatility in the stock market as measured by the VIX-index is the single most influential predictor of next days' stock returns.

The remainder of this thesis is organized as follows. The prediction problem and the different machine learning methods are presented in section 2. The dataset and the model selection process for different machine learning methods are described in section 3. The empirical analysis and the results are covered in section 4. Section 5 concludes.

2 Methodology

2.1 Multinomial stock returns

Financial literature usually focuses on the reward of holding a risky asset such as stocks compared to the risk-free investment. This excess return is denoted as

$$Z_t = r_t - rf_t, \tag{1}$$

where r_t is the logarithmic daily return of the S&P 500 stock market index at time t and rf_t is the 3-month Treasury bill yield[1]. In directional prediction the binary dependent variable is created from the return series in equation (1) using an indicator function

$$B_t(c) = I(Z_t > c), \tag{2}$$

where c is a given threshold. The multinomial response variable with three classes can be derived from the continuous stock returns using two thresholds c_1 and c_2

$$R_t(c_1, c_2) = \begin{cases} 1, & \text{if } Z_t < c_1 \\ 2, & \text{if } c_1 \leq Z_t \leq c_2 \\ 3, & \text{if } Z_t > c_2 \end{cases} . \tag{3}$$

A natural question is how to choose the two thresholds that are basically arbitrary. To the best of our knowledge the multinomial approach with two thresholds as in equation (3) has not been considered in the previous literature regarding directional prediction of stock returns. Majority of the previous literature with single threshold as in equation (2) focus on binary sign prediction, where $c = 0$ (see e.g., Leung et al., 2000; Nyberg & Pönkä, 2016). Although previous literature on directional prediction of stock returns with a single non-zero threshold is quite scarce some alternatives have been considered.

Chung and Hong (2007) argue that the choice of c can be based on the observed data or alternatively held fixed using the magnitude of transaction costs for example. In their data based approach Chung and Hong (2007) use multiples of the estimated standard deviation as a threshold when forecasting the direction of exchange rates. Linton and Whang (2007) consider different unconditional quantiles of the return series when testing for directional prediction in stock returns. Linton and Whang (2007) report statistically significant predictability in daily returns for all but the most extreme quantiles, where the amount of data is insufficient.

[1] The Federal Reserve reports annualized yields using a 360-day year also known as the bank discount method. The daily yield is therefore calculated as $rf_t = tbill_t * \frac{1}{360}$.

Maheu and McCurdy (2004) show that large price changes of individual stocks are driven by important news and these large changes tend to be clustered together. It is also noted using market level data that large absolute stock returns contain stronger positive autocorrelation than small absolute returns do and are therefore more predictable (see e.g., Granger & Ding, 1996). Setting the thresholds c_1 and c_2 in equation (3) further apart from zero may result in more predictability but also in more imbalanced classes.

Since the main objective of this thesis is to compare the predictive ability of several different machine learning methods we have chosen to use the upper and lower quartiles of the return series as thresholds. This data based approach yields nicely balanced classes as one half of the observations are coming from the middle class in equation (3) and the other half from the "abnormal" classes. Well balanced classes also allow for similar rules to be used with each method in the classification process, where the probability estimates are transformed into classification.

Consider the stochastic processes R_t and $\mathbf{x}_{t-1}$, where R_t is the multinomial response variable described in equation (3) and $\mathbf{x}_{t-1}$ is a $p \times 1$ vector of predictors at time $t-1$. Conditional on the information set we assume the response variable to follow a categorical distribution

$$R_t|\Omega_{t-1} \sim Cat(\mathbf{p}_t),$$

where Ω_{t-1} is the information set available at time $t-1$ and $\mathbf{p}_t$ is a $k \times 1$ vector of conditional probabilities. Each element of $\mathbf{p}_t$ is the conditional probability of class k being the observed class at time t. More formally the conditional probability for each class k can be written as

$$p_{tk}(\mathbf{x}_{t-1}) = P(R_t = k|\Omega_{t-1}),\ k = 1, 2, ..., K. \tag{4}$$

The conditional probabilities in equation (4) must satisfy $0 \leq p_{tk} \leq 1$ and $\sum_{k=1}^{K} p_{tk} = 1$. These conditions are met by the symmetric multiple logistic transform. The conditional probabilities for each class k can be constructed using the functional estimates $F_k(\mathbf{x}_{t-1})$

$$p_{tk}(\mathbf{x}_{t-1}) = \frac{e^{F_k(\mathbf{x}_{t-1})}}{\sum_{l=1}^{K} e^{F_l(\mathbf{x}_{t-1})}}. \tag{5}$$

In the general K-class classification problem the goal is to find the function that minimizes the expected loss of some predefined loss function for each class k

$$\{F_k(\mathbf{x}_{t-1})\}_{k=1}^{K} = \arg \min_{\{F_k(\mathbf{x}_{t-1})\}_{k=1}^{K}} E[\mathcal{L}(R_t, \{F_k(\mathbf{x}_{t-1})\}_{k=1}^{K})].$$

For the majority of methods used in this thesis the loss function considered is the multinomial deviance

$$\mathcal{L}(R_t, \{F_k(\mathbf{x}_{t-1})\}_{k=1}^{K}) = -\sum_{k=1}^{K} I(R_t = k) \log p_{tk}(\mathbf{x}_{t-1}), \tag{6}$$

where $p_{tk}(\mathbf{x}_{t-1})$ is the logistic transform presented in equation (5).

Accuracy is used as the evaluation metric in this thesis to compare the classification performance of different machine learning methods. Accuracy is calculated as the proportion of correctly classified data points in the considered sample

$$Acc = \frac{1}{N} \sum_{t=1}^{N} I(\hat{R}_t = R_t),$$

where $\hat{R}_t$ is the predicted class and R_t the true class label at time t.

2.2 Machine learning methods

2.2.1 k-Nearest neighbor classifier

The k-nearest neighbor originally presented by Fix and Hodges (1951) can be considered a model-free classification method since the classification of a new observation is based purely on the data points of the training set. Our training set consists of N pairs $\{(\mathbf{x}_{t-1}, R_t)\}_{t=1}^{N}$, where $\mathbf{x}_{t-1}$ is the vector of feature values and R_t is the multinomial response variable given in equation (3). In order to classify a new data point $\mathbf{x}_N$ we need to find the k data points in the training data closest to the new data point based on some distance measure. The Euclidean distance is the most commonly used alternative. These k data points are called the nearest neighbors of $\mathbf{x}_N$. The final classification is based on a majority vote of the response values of these k nearest neighbors. Ties are broken at random. This process is repeated for each data point in the test set.

Figure 1 illustrates how the k-nearest neighbor classification works with a small artificial dataset containing three classes. The left-hand side of Figure 1 illustrates the nearest neighbors of two new data points based on Euclidean distance. The amount of nearest neighbors k is assumed to be either 1 or 3. For two example locations marked with X, the solid gray circle shows the one nearest neighbor and

dashed circle illustrates the neighbors when k equals three. The right-hand side of Figure 1 shows the decision boundary for this artificial data when k equals one.

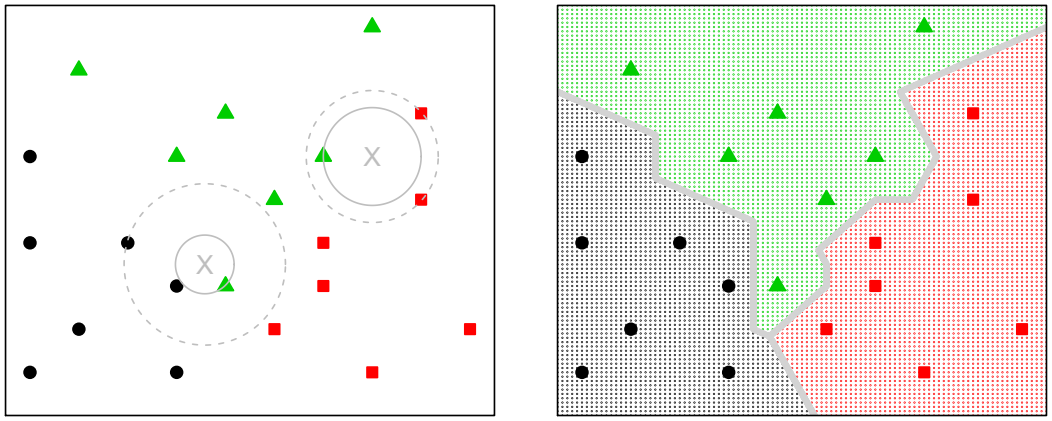

Figure 1: k-Nearest neighbor classification

The k-nearest neighbor classification is used as a benchmark method in this thesis because it is fairly easy to finetune. The only tuning parameter of the method is the amount of neighbors k. Larger values of k lead to smoother and less detailed decision boundaries. Despite its simplicity k-nearest neighbor has shown success in different kinds of classification problems such as handwritten digits or satellite image scenes. Since the features in the dataset could have a variety of different scales each feature is typically re-scaled to have mean zero and variance equal to one. (Hastie, Tibshirani & Friedman, 2009).

2.2.2 Gradient boosting

The classification algorithm called adaboost was first introduced by Freund and Schapire (1996). For a long time the classification ability of the adaboost algorithm remained controversial. This was until Friedman, Hastie and Tibshirani (2000) created a statistical framework for the boosting procedure and showed how the adaboost algorithm fits an additive logistic regression model. The more general gradient boosting algorithm was discovered as Friedman (2001) introduced the connection to numerical optimization in function space. The more general gradient boosting algorithm can be used for both classification and regression problems.

In gradient boosting the goal is to find the function minimizing the expected loss of some predetermined loss function

$$\hat{F}(\mathbf{x}_{t-1}) = \arg \min_{F(\mathbf{x}_{t-1})} E\left[\mathcal{L}\left(y_t, F(\mathbf{x}_{t-1})\right)\right].$$

In order to keep the notation fairly simple let us consider the binary classification problem, where $y_t \in \{0, 1\}$ and $\mathcal{L}(y_t, F(\mathbf{x}_{t-1}))$ is the binomial deviance. With the multinomial response variable presented in equation (3) a separate function is estimated for each class, which complicates the notation.

Gradient boosting is an ensemble method, where the possibly very complex final model is a combination of simple models called base learners.

$$F_M(\mathbf{x}_{t-1}) = \sum_{m=1}^{M} f_m(\mathbf{x}_{t-1}) \tag{7}$$

The base learners $f(\mathbf{x}_{t-1})$ are assumed to belong to some parameterized class of functions. These could be for example simple linear models, spline functions or regression trees. The base learner used in this study is the J-terminal node regression tree, which splits the predictor space into J disjoint regions and attaches a constant to each region. Mathematically the J-terminal node regression tree base learner can be written as

$$f(\mathbf{x}_{t-1}; \{c_j, R_j\}_{j=1}^{J}) = \sum_{j=1}^{J} c_j I(\mathbf{x}_{t-1} \in R_j), \tag{8}$$

where $c_j \in \mathbb{R}$ is the functional estimate in region R_j.

Figure 2 illustrates J-terminal node regression trees graphically and plots a 4-terminal node regression tree and the terminal node regions created by this tree. The left-hand side depicts the classical tree shape. Each of the three split points is a function of the splitting variables and split locations. The right-hand side of Figure 2 shows the terminal node regions $\{R_j\}_{j=1}^{4}$ and the split locations $\{t_l\}_{l=1}^{3}$in a 2-dimensional space.

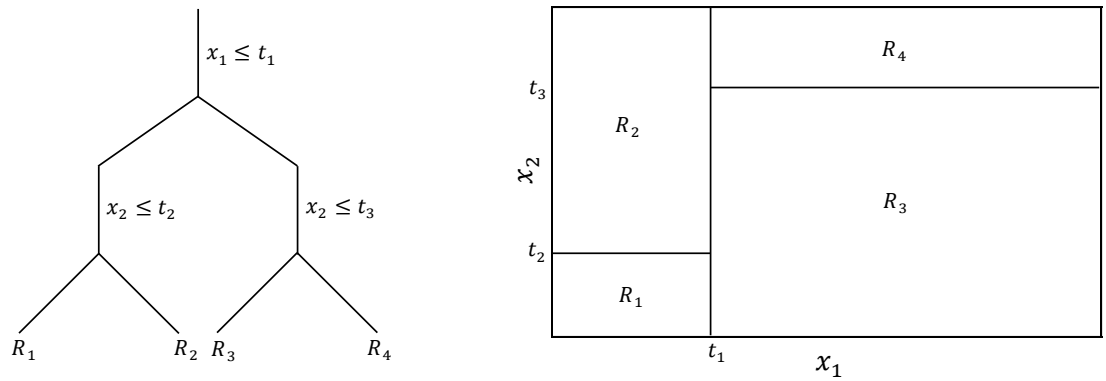

Figure 2: 4-terminal node regression tree

The final ensemble in equation (7) is estimated in a greedy stagewise fashion using a method called forward stagewise additive modeling. The estimation of a gradient boosting model is described in Algorithm 1. The algorithm starts with an initial value, which is a simple constant based on the considered loss function. At each iteration m of the gradient boosting algorithm a new base learner function which best fits the negative gradient of the loss function is selected and added to the current ensemble F_{m-1}. With the J-terminal node regression tree in equation (8) as the base learner this corresponds to finding the J non-overlapping terminal node regions $\{R_{jm}\}_{j=1}^{J}$ using a least squares criterion. After finding the terminal node regions the functional estimates $\hat{c}_{jm}$ are obtained in a simple minimization problem. The current ensemble F_{m-1} is then updated with the functional estimates before calculating the pseudo responses $\tilde{y}_t$ for the next round of the algorithm.

Algorithm 1 *Gradient boosting using J-terminal node regression trees*

$F_0(\mathbf{x}_{t-1}) = \arg\min_{\rho} \frac{1}{N} \sum_{t=1}^{N} L\left(y_t, \rho\right)$

for $m \leftarrow 1$ *to* M ***do:***

$\tilde{y}_t = - \left. \frac{\partial L(y_t, F(\mathbf{x}_{t-1}))}{\partial F(\mathbf{x}_{t-1})} \right|_{F(\mathbf{x}_{t-1}) = F_{m-1}(\mathbf{x}_{t-1})}, \qquad t = 1, .., N$

estimate $\{R_{jm}\}_{j=1}^{J}$ *using the least squares criterion*

$\hat{c}_{jm} = \arg\min_{c_{jm}} \sum_{\mathbf{x}_{t-1} \in R_{jm}} L(y_t, F_{m-1}(\mathbf{x}_{t-1}) + c_{jm}), \qquad j = 1, .., J$

$$F_m(\mathbf{x}_{t-1}) = F_{m-1}(\mathbf{x}_{t-1}) + \upsilon \sum_{j=1}^{J} \hat{c}_{jm} I(\mathbf{x}_{t-1} \in R_{jm})$$

end for

Algorithm 1 also illustrates the tuning parameters related to the gradient boosting method. The amount of iterations M and the learning rate $\upsilon \in]0, 1]$ control the learning process. Setting M too low can result in underfitting whereas too many repeats can lead to overfitting. Setting the learning rate smaller than one can be seen as a shrinkage strategy as the parameter υ shrinks each functional estimate towards zero and thereby controls the speed of the learning process. These two parameters are inversely related to each other. A smaller learning rate usually requires more trees to be built (Hastie et al., 2009).

The amount of complexity related to the J-terminal node regression tree base learner function can be controlled by the amount of terminal nodes J and the amount of observations required at each terminal node region. Requiring more observations in each terminal node region narrows down the amount of potential split points and therefore controls the complexity of each tree. Building larger trees with more terminal nodes results in more complex models but the risk of overfitting also grows.

Note from the graphical illustration in Figure 2 that in order to build a J-terminal node regression tree $J - 1$ split points are needed and the size of the regression tree also controls the amount of interactions allowed between different predictors. Instead of requiring the exact amount of terminal nodes J some software implementations use the depth of the tree D as a tuning parameter. The depth of the regression tree is the maximum amount of inner nodes between the root and leaf nodes. The depth of the regression tree in Figure 2 for example is two since there are two split points between the root and each leaf node.

Different subsampling strategies can also be used for reqularization with the gradient boosting model. The subsampling is usually done row-wise, where only a certain fraction η_{row} of training samples are used when estimating the parameters of the base learner function at each round of the algorithm. By using row-wise subsampling the regression trees at each round tend to be less similar. Additionally column-wise subsampling is also available, where only a certain fraction η_{col} of the available predictors are used at each round of the gradient boosting algorithm. The exact amount of subsampling used both row-wise and column-wise are finetuned using cross-validation.

2.2.3 Random forest

The random forest algorithm of Breiman (2001) has a close connection to both bagging and the adaboost classification algorithm. The final model with each of these three methods is an ensemble of simple models. The original idea of random forest is to improve the classification ability of bagging by reducing the correlation between each component in the final ensemble. This is done by injecting additional randomness when building each component of the final model.

Similarly as with boosting the base learner function used at each step of the random forest algorithm is a tree-based model. Unlike the regression tree presented in equation (8) the base learner with random forest classification algorithm is a classification tree. The graphical illustration given in Figure 2 holds for the classification tree as well, but now the functional estimate in each terminal node region of the J-terminal node classification tree is the predicted class

$$f(\mathbf{x}_{t-1}; \{C_j, R_j\}_{j=1}^{J}) = \sum_{j=1}^{J} C_j I(\mathbf{x}_{t-1} \in R_j), \tag{9}$$

where $\mathbf{x}_{t-1}$ is a vector of inputs at time $t-1$ and C_j is the predicted class in region R_j.

Instead of fixing the number of terminal nodes J as with gradient boosting the complexity of each tree in the random forest is typically controlled by requiring a certain number of observations at each terminal node. In the random forest algorithm the depth of each tree is increased by adding additional split points for as long as the number of observations in the terminal node is greater than a prespecified constant $n_{\min}$. This constant is a tuning parameter related to the random forest algorithm as all the terminal nodes must hold at least $n_{\min}$ data points. Especially with classification problems the trees in the random forest are often grown to the full size requiring only one observation in each terminal node region. Hastie et al. (2009) argue that letting the trees in the random forest to grow to the maximum size seldom costs much and results in one less tuning parameter.

The power of random forests comes from combining the predictions of many accurate individual trees that are as diverse as possible. In order to make the trees in the random forest ensemble less correlated only a subset of features are considered when new split points are added to the classification tree. Suppose the number of features in the dataset is p then only m ($m \leq p$) randomly chosen features are

considered as candidates when selecting a new split point. The exact amount m depends on the problem at hand and is treated as a tuning parameter. Especially in problems where the proportion of relevant features in the whole feature set is small, setting m too low may result in poor performance (Hastie et al., 2009).

Similarly as in bagging, a bootstrap sample Z^* of size N is drawn from the training data at each round $b \in \{1, ..., B\}$ of the algorithm. A new decision tree $f_b(\mathbf{x}_{t-1}, \mathbf{\Theta})$ is fit using this bootstrap sample, where the parameter vector $\mathbf{\Theta}$ holds the parameters of the decision tree presented in equation (9). The split points of this decision tree are found recursively by considering only the m randomly chosen features at each step. The decision tree is grown to the maximum possible size controlled by the parameter $n_{\min}$, which sets the minimum number of observations needed in each node of the tree. This process is summarized in Algorithm 2.

Algorithm 2 *Random Forest classification*

for $b \leftarrow 1$ *to* B ***do:***

draw a bootstrap sample $\mathbf{Z}^*$ *of size* N *from the training data*

create an empty decision tree $f_b(\mathbf{x}_{t-1} \in \mathbf{Z}^*, \mathbf{\Theta} = \emptyset)$

while *(the number of observations in some node* $> n_{\min}$*)* ***do:***

randomly select m *variables*

pick the best split point among these

split the current node into two daughter nodes

end while

include $f_b(\mathbf{x}_{t-1,}\mathbf{\Theta})$ *in the ensemble* F_b

end for

The final ensemble in the random forest algorithm is a combination of the individual trees found in each round b of the algorithm

$$F(\mathbf{x}_{t-1}) = \{f_b(\mathbf{x}_{t-1}, \mathbf{\Theta})\}_{b=1}^{B}.$$

The classification of a new data point $\mathbf{x}_N$ is based on a majority vote between the classifications induced by each individual tree

$$\hat{C}_{rf}(\mathbf{x}_N) = majority\ vote\{\hat{C}_b(\mathbf{x}_N)\}_{b=1}^{B},$$

where $\hat{C}_b(\mathbf{x}_N)$ is the predicted class given by the b_{th} decision tree in the random forest ensemble.

2.2.4 Neural networks

Neural networks were originally designed as a tool to model the information processing capabilities of the human brain and the earliest attempts go as far as the 1940s (Rojas, 1996). There are a vast amount of neural network models with different assumptions regarding the structure of the network and how information flows through the network. The model used in this thesis is one of the most commonly used neural network models called a single hidden layer feed-forward neural network (Bishop, 2006).

The network consists of three layers which are typically named as the input layer, hidden layer and the output layer. Each layer in a feed-forward network is connected with the subsequent layer through weights as is visualized in Figure 3. The directed edges represent the weights and the direction of information flow in the network.

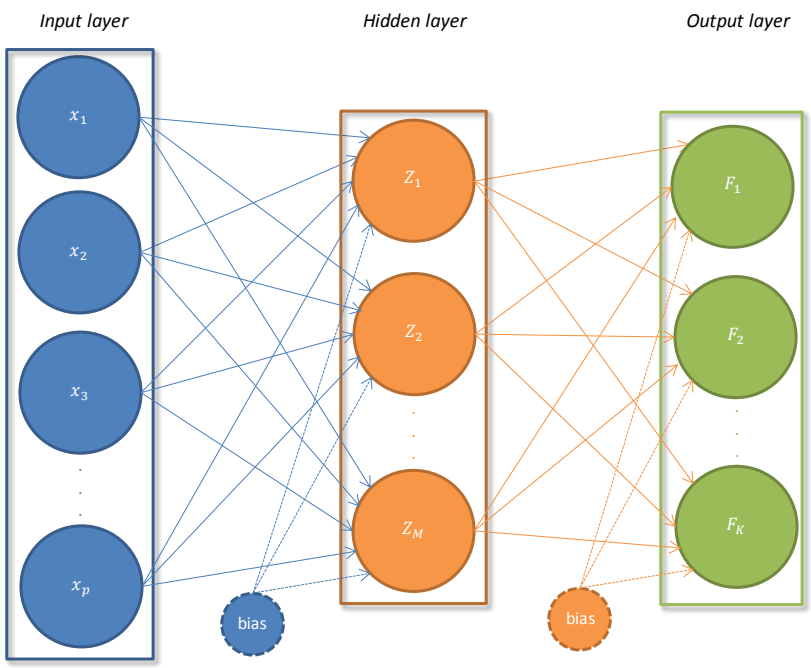

Figure 3: Artificial neural network

In general there could be several hidden layers creating a deeper and more complex network. Each unit in the hidden layer of Figure 3 is called a hidden unit since these

are typically unobserved. These hidden units are linear combinations of the input variables $\mathbf{x} = (x_1, x_2, \ldots, x_p)'$ followed by a non-linear activation function:

$$Z_m = h(\alpha_{0m} + \boldsymbol{\alpha}_m' \mathbf{x}), \qquad m = 1, ..., M, \tag{10}$$

where α_{0m} is the weight from the bias unit, $\boldsymbol{\alpha}_m$ is a $p \times 1$ vector of weights coming into hidden unit Z_m and $h()$ is the activation function. The sigmoid function is typically used to transform the linear combinations of inputs into a non-linear form. Another common choice for the activation function is the hyperbolic tangent function. Note that the total amount of weights connecting the units in the input layer and the hidden layer is $M \times (p + 1)$, where M is the number of units in the hidden layer. The exact amount for M is treated as a tuning parameter of the model.

The final output for each class k is formed as a linear combination of the hidden units $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_M)'$, which is transformed in the interval $[0, 1]$ using the softmax function:

$$F_k = g(\beta_{0k} + \boldsymbol{\beta}_k' \mathbf{Z}) = \frac{e^{\beta_{0k} + \boldsymbol{\beta}_k' \mathbf{Z}}}{\sum_{l=1}^{K} e^{\beta_{0l} + \boldsymbol{\beta}_l' \mathbf{Z}}}, \qquad k = 1, ..., K.$$

Similarly as in equation (10) β_{0k} is the weight from the bias unit and $\boldsymbol{\beta}_k$ is a $M \times 1$ vector of weights connecting the units in the hidden layer to the output unit F_k.

The optimal weights in the network minimize the considered loss function. In the multinomial classification problem the loss function to be minimized is the sample counterpart of the multinomial deviance shown in equation (6). By denoting the complete set of weights in the network by a weight vector $\boldsymbol{\theta}$ the loss function can be written as

$$L(\boldsymbol{\theta}, F_k) = -\sum_{t=1}^{N} \sum_{k=1}^{K} I(R_t = k) \log F_k, \tag{11}$$

where F_k is the output for class k. The set of weights in the network can be searched using a gradient descent based method called backpropagation. In backpropagation the gradient of the loss function in equation (11) is calculated at each iteration. The weights in the network are then updated according to the direction given by the negative gradient. For a more detailed description of the backpropagation algorithm see e.g. Rojas (1996).

A simple regularization strategy called weight decay has been suggested to avoid overfitting while estimating the optimal weights in the network. In weight decay an

additional penalty term, which penalizes large weights, is added to the loss function presented in equation (11)

$$\tilde{L}(\boldsymbol{\theta},F_k) = L(\boldsymbol{\theta},F_k) + \lambda J(\boldsymbol{\theta}),$$

where λ is the weight decay parameter. The penalization function $J(\boldsymbol{\theta})$ can take various forms. A common choice is to impose quadratic penalization, where $J(\boldsymbol{\theta}) = \boldsymbol{\theta}'\boldsymbol{\theta}$ (Bishop, 2006). Larger values for λ thereby shrink the weights towards zero unless traditional backpropagation reinforces the weights. The exact amount of penalization needed is finetuned using cross-validation.

2.2.5 Support vector machines

The support vector machines originally presented by Vapnik (1995) can be used for both classification and regression problems. The basic idea and the terminology of support vector machines can be illustrated using a two-class classification problem with a linear decision boundary. The left-hand side of Figure 4 illustrates the case with perfect separability. The right panel in Figure 4 shows the nonseparable case, where some data points are misclassified by the linear decision boundary.

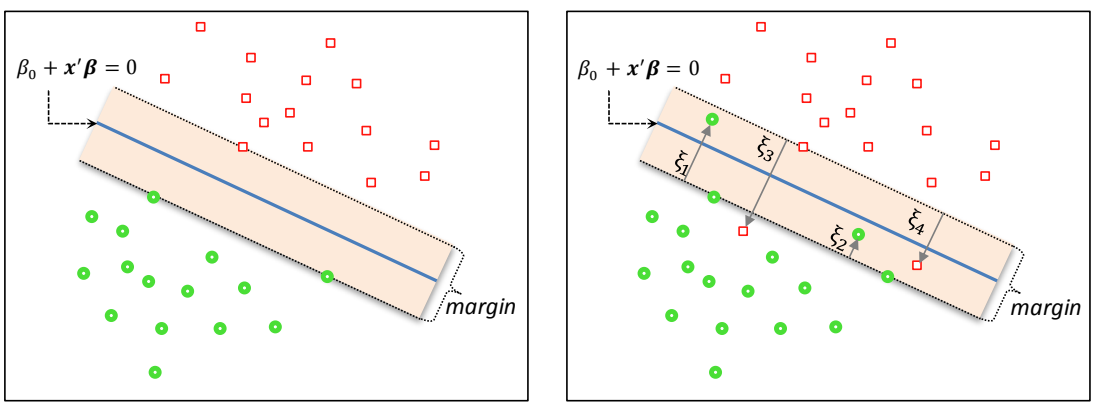

Figure 4: Support vector machine

The solid blue line in Figure 4 is the decision boundary separating the two classes

$$F(\mathbf{x}) = \beta_0 + \mathbf{x}'\boldsymbol{\beta} = 0,$$

where β_0 is a constant term, $\boldsymbol{\beta}$ is a unit vector and $\mathbf{x}$ is a $p \times 1$ vector of input variables. For notational reasons let us focus on the binary classification case and denote the binary response as $y_i \in \{-1, 1\}, i = 1, ..., N$, where i can be associated to time t. The one-against-one method used in this thesis for K-class classification with support vector machines is a direct extension to the binary case as the final classification is based on a voting scheme between the $K(K-1)/2$ binary classifiers constructed for each class pair. For more information on the multiclass classification with support vector machines see e.g. Hsu and Lin (2002).

The goal with support vector machines is to find the decision boundary with maximum area on both sides of the boundary also known as the margin

$$M = \frac{2}{\|\boldsymbol{\beta}\|}.$$

In Figure 4 the dashed lines illustrate the margin and the points located at the dashed lines are known as support vectors. These support vectors play a crucial role when searching for the optimal decision boundary as is shown mathematically later on.

Hastie et al. (2009) show that instead of maximizing the margin the optimization problem can be written in terms of minimizing $\|\boldsymbol{\beta}\|$

$$\min_{\beta_0, \boldsymbol{\beta}} \|\boldsymbol{\beta}\| \quad s.t. \quad y_i(\beta_0 + \mathbf{x}_i^{'}\boldsymbol{\beta}) \geq 1, \quad i = 1, ..., N. \tag{12}$$

The constraint in equation (12) requires each observation to be on the right side of the margin. This constraint does not hold for the nonseparable case shown on the right panel of Figure 4 since some observations are on the wrong side of the margin. For this reason we need to define a vector of slack variables $\boldsymbol{\xi} = (\xi_1, ..., \xi_N)$. For the data points on the correct side of the margin the slack variable is equal to zero and for the misclassified observations $\xi_i > 1$. The slack variable is between zero and one when the data point is on the incorrect side of the margin but correctly classified. All of these cases are illustrated on the right-hand side of Figure 4.

In the nonseparable case the constraint in the optimization problem of equation (12) is re-formulated so that some of the observations can be on the incorrect side of the margin. The proportional amount of observations on the wrong side of the margin is bound by requiring the sum of slack variables to be smaller than some

predefined constant. The minimization problem then becomes

$$\min_{\beta_0,\boldsymbol{\beta}} \|\boldsymbol{\beta}\| \quad s.t. \begin{cases} y_i(\beta_0 + \mathbf{x}_i^{'}\boldsymbol{\beta}) \geq 1 - \xi_i, \ i = 1, ..., N, \\ \xi_i \geq 0, \ \sum_{i=1}^{N} \xi_i \leq \text{constant}. \end{cases} \tag{13}$$

The convex optimization problem with quadratic objective and linear inequality constraints in equation (13) can be solved using quadratic programming. The Lagrange primal function can be written as

$$L_P = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(\beta_0 + \mathbf{x}_i^{'}\boldsymbol{\beta}) - (1 - \xi_i)] - \sum_{i=1}^{N}\nu_i\xi_i, \tag{14}$$

where C is now inplace of the predetermined constant in equation (13). The parameters α_i and ν_i are the Lagrange multipliers. The cost parameter C is a tuning parameter of the procedure and controls how wide the margin is. A larger value of C puts more emphasis on the points near the decision boundary and requires a tighter margin.

The Lagrange dual objective function can be formulated by plugging the solved parameter values from the first order conditions of the primal problem back to equation (14)

$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \tag{15}$$

where alphas are the Lagrange multipliers from the minimization problem in equation (14) and $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is the inner product of vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. The dual problem in equation (15) is often easier to solve than the primal (Hastie et al., 2009). Without going into the details of solving the minimization problem the optimal solution turns out to be the following:

$$\hat{\boldsymbol{\beta}} = \sum_{i=1}^{N}\hat{\alpha}_i y_i \mathbf{x}_i. \tag{16}$$

Equation (16) shows how the optimal decision boundary is determined by the estimated alphas. These estimated alphas are non-zero only for the observations characterized as the support vectors. The support vectors therefore have a direct impact on the location of the decision boundary. It should be noted that the support vectors can also be located inside their margin in the nonseparable case.

So far we have only dealt with linear decision boundaries. To consider nonlinear decision boundaries the original input feature space is typically transformed

into an enlarged space using e.g. polynomials or splines since the data could be linearly separable in this higher dimensional feature space. Without specifying the exact transformation the dual problem in equation (15) can be written using these transformed feature vectors $h(\mathbf{x}_i)$

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle h(\mathbf{x}_i), h(\mathbf{x}_j) \rangle, \tag{17}$$

where $\langle h(\mathbf{x}_i), h(\mathbf{x}_j) \rangle$ is the inner product of the transformed input vectors i and j. The solution to the dual Lagrangian in equation (17) depends on the transformed higher dimensional data only through inner products. Instead of the exact transformation $h()$ a kernel function, which computes inner products in the transformed space, is sufficient. A radial basis function and a d_{th}-degree polynomial are typical choices for the kernel function. The radial basis function can be written as

$$K(\mathbf{x}, \mathbf{x}_i) = \langle h(\mathbf{x}), h(\mathbf{x}_i) \rangle = \exp(-\gamma ||\mathbf{x} - \mathbf{x}_i||^2), \tag{18}$$

where γ is a tuning parameter related to the radial basis function kernel. The d_{th}-degree polynomial kernel function involves one extra tuning parameter compared to the radial basis kernel presented in equation (18). The degree of the polynomial d needs to be finetuned in addition to a scale parameter s

$$K(\mathbf{x}, \mathbf{x}_i) = (1 + s\langle \mathbf{x}, \mathbf{x}_i \rangle)^d. \tag{19}$$

The final classification in the support vector machine is produced by the following equation

$$\hat{G}(x) = sign(\hat{F}(x)) = sign(\sum_{i=1}^{N} \hat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i) + \hat{\beta}_0),$$

where $K(\mathbf{x}, \mathbf{x}_i)$ is one of the kernel functions presented in equations (18) and (19). $\hat{\alpha}_i$ and $\hat{\beta}_0$ are the solved coefficients from the optimization problem. Similarly as with the linear decision boundary the coefficients $\hat{\alpha}_i$ are non-zero only for the data points marked as support vectors.

2.3 Previous literature

Forecasting stock prices has attracted a great amount of interest in the recent decades. Especially the machine learning community has been very actively producing new research in this field. The typical research explores if a more flexible

machine learning method could improve the predictions by exploiting non-linear relationships in the data. The previous literature is quite vast. Atsalakis and Valavanis (2009) produce a comprehensive survey of more than one hundred different published articles related to stock market forecasting using neural network based techniques alone. For a more recent survey on financial forecasting using a variety of different machine learning methods see e.g., Cavalcante, Brasileiro, Souza, Nobrega and Oliveira (2016).

Because of the huge amount of previous research the following literature review focuses on the particular strand of literature where the dependent variable is binary. To the best of our knowledge the multinomial response variable presented in equation (3) has not been considered in previous literature related to the directional prediction of stock returns. The closest alternative can be found in Boonpeng and Jeatrakul (2016), where they compose the daily price information in the stock market into three different trading signals using technical analysis technique called pivoting. These different types of trading signals are then predicted using alternative technical analysis indicators as input.

Kara, Boyacioglu and Baykan (2011) compare the performance of artificial neural networks (ANN) and support vector machines (SVM) in predicting the daily direction of change in the Istanbul stock exchange. Various technical analysis indicators are used as the input variables. In a very similar study Patel, Shah, Thakkar and Kotecha (2015) forecast the direction of two Indian stock indices and two individual stocks. In addition to ANN and SVM Patel et al. (2015) also consider random forests. The single hidden layer feed-forward network produces slightly more accurate results than the SVM with polynomial kernel in Kara et al. (2011). Patel et al. (2015) reach another conclusion with the Indian data as SVM is seen to outperform ANN. Random forest however provides the most accurate directional predictions for both the indices and individual stocks considered in Patel et al. (2015).

Both Kara et al. (2011) and Patel et al. (2015) conduct a quite extensive grid search to find the optimal tuning parameter combinations for each model. The dataset for parameter optimization in both of these studies is randomly selected from the entire dataset. For this reason the actual forecasting results are highly controversial, since the models have already seen some of the test data while optimizing the tuning parameters. The fairly high hit ratios above 70 or even above 80 percent are in line with this observation.

Other studies reporting results in favor of using the SVM over the ANN in forecasting the direction of stock markets are found using data from the Asian markets.

Huang, Nakamori and Wang (2005) forecast the direction of a Japanese stock index using weekly data. Huang et al. (2005) also report impressive hit ratios above 70 percent, but their results are based on a test set consisting of 36 observations only. Kim (2003) use the radial basis function SVM to predict the daily direction of Korean stock exchange using technical analysis indicators as input. SVM is seen to outperform ANN but the difference between these two methods is not statistically significant. The magnitude of the hit ratios reported by Kim (2003) is closer to a true stock market forecasting experiment.

Ballings, den Poel, Hespeels and Gryp (2015) provide a more extensive comparison of different machine learning methods in directional prediction of stock returns. They use yearly data for 5767 listed European companies and forecast if the yearly price change of an individual stock in year 2010 is above 35 percent or not. Using such a high threshold to create the binary response variable leads to highly imbalanced classes. The class imbalance problem is handled by oversampling the majority class.

All the machine learning methods considered in this thesis except for the gradient boosting model (GBM) are included in the model set of Ballings et al. (2015). The family of boosting algorithms is however represented as the adaboost classification algorithm is also studied. Random forest is found to be the top-performer followed by SVM and adaboost based on the median area under the ROC-curve (AUC) among the individual firms. Krauss et al. (2017) also focus on the performance of individual companies and model if the daily return of a particular stock in the S&P 500 index is above the market return or not. Both random forest and GBM is seen to outperform deep neural networks while a combination of all these models yields the most accurate predictions.

Zhong and Enke (2017) run different linear and non-linear dimensionality reduction techniques before applying ANN in predicting the direction of the S&P 500 stock market index. The standard linear principal component analysis (PCA) combined with the ANN yields the most accurate predictions and provides important insights into selecting the optimal predictor set. Especially lagged returns, other stock markets, the largest companies in the S&P 500 index and different exchange rates are considered as influential predictors in the PCA-step.

In a recent study Basak, Kar, Saha, Khaidem and Dey (2019) conduct a directional prediction for ten randomly selected companies included in the S&P 500 index using a variety of different machine learning models. Before constructing the binary

response variable they smooth the return series using exponential smoothing. This exponential smoothing shifts the focus onto detecting a medium to long-term price trend instead of the daily direction of change in market prices.

Different technical analysis indicators derived from the smoothed return series are used as the input for models. The model set considered by Basak et al. (2019) include all the models used in this thesis except for k-nearest neighbors. Random forest yields the best long-term trend forecasts before GBM and ANN. The selection for tuning parameters is fairly novel and the results should be viewed with healthy criticism. For example the worst performing method is the SVM, which is trained using the linear kernel function only.

3 Data and model setup

3.1 Data

The dataset used in the empirical section of this thesis covers daily returns of the S&P 500 stock market index from the beginning of the 1990s to the end of 2018[2]. The goal of the empirical analysis is to study a wide spectrum of different variables that could be used for prediction using the maximum amount of daily data available. With such a high frequency as daily returns the potential predictor variables are mostly based on different types of financial market data.

As was seen in the previous section technical analysis indicators have been the most common choice for the input variables of different machine learning methods (see e.g., Kim, 2003; Basak et al., 2019). In technical analysis different types of indicators are calculated using the historical price or return information from the stock market. Benchmark yields and different interest rate spreads from the corporate and government bond markets have also been extensively studied with both daily and monthly data (see e.g., Zhong and Enke, 2017; Nyberg and Pönkä, 2016). Interest rates express the tightness of the monetary policy set by the Federal Reserve. Different types of interest rate spreads reflect market expectations regarding the upcoming economic activity or the riskiness of the corporate sector for example.

Lagged stock returns and returns from other stock markets are another commonly used alternative (see e.g., Zhong and Enke, 2017). A less studied predictor group is the volatility in different markets. A recent study by Becker & Leschinski (2018) shows that the VIX-index, which is often called the fear factor of stock markets, can also be a viable alternative. As with Zhong and Enke (2017) different exchange rates and commodities indices are also considered as potential features to be used for prediction in this thesis. The appreciation (or depreciation) of the dollar relative to other currencies affects the foreign trade and international flow of funds to the U.S. Variables related to the state of the macroeconomy are found to be important predictors when predicting monthly stock returns (Nyberg, 2011). Unfortunately the majority of the macroeconomic information is not available with daily frequency.

Table 1 summarizes the input variables using seven different categories. A short description and an illustrative example are shown from each category. The full

[2] After deriving and lagging the predictor variables the exact time period is 12.2.1990 - 5.10.2018.

predictor set and the exact transformations for each predictor can be found in the appendix.

Table 1: Predictor groups

Group	Description	Example
Stock market	S&P 500 price information, Returns from other stock markets	Lagged returns, Returns from DAX or FTSE
Interest rates	Government and corporate benchmark yields	3-month T-bill, Term spread
Exchange rates	The appreciation of dollar relative to other currencies	Dollar/British Pound, major currencies index
Commodities	Information from the commodities market	Copper, Oil, Gold, Silver
Volatility	Volatility in the stock and bond markets	VIX-index, MOVE-index
Technical analysis	Indicators derived from price or return information	Relative strength index
Macro	Information regarding the macroeconomy	ADS-index

The total amount of different predictors studied in this thesis is 37. Following the approach of Krauss et al. (2017) various lag lengths of the predictors are also considered. Lag lengths beyond ten trading days are found to be uninformative in the preliminary analysis using the model selection capability of the gradient boosting machine[3]. By considering the lagged predictors from the previous ten trading days the full predictor set consists of 370 different inputs (37×10). Only data points (days) with information available for each predictor are considered. For this reason

[3]Results available upon request.

predictor variables utilizing market data outside the U.S. is kept minimal because of the individual holiday periods in each country. After leaving out the data points with missing values the final dataset includes 6686 daily observations.

From the machine learning methods considered in this thesis only the tree-based classifiers gradient boosting machine and random forest are capable of handling such a large predictor set. Both of these methods are able to perform model selection simultaneously with estimation as new split points are introduced for the tree-based base learners. Other methods end up easily overfitting the data with a large predictor set and therefore a reduced dataset is needed. A smaller predictor set is built using a combination of prior knowledge and the results from the tree-based methods.

Both GBM and random forest select the VIX-index as the single most influential predictor[4]. For the random forest model each of the ten most influential inputs are different lag lengths of the VIX-index, whereas GBM includes six lags of VIX in the top-10. Different technical analysis indicators are also considered as important predictors by both models. The best performing technical analysis indicators are the stochastic oscillator (StochK), moving average convergence divergence (MACD) and Williams %R (Rperc)[5]. The ranking of these indicators are slightly different for the two methods.

In addition to these the spread between the daily high and low stock prices is selected by both models. GBM also ranks the corporate interest rate spread among the ten most influential predictors. The results from the principal component analysis by Zhong and Enke (2017) are in favor of using the lagged stock returns and international stock returns. Lagged stock returns from the S&P 500 and the returns from the German stock index are thereby also included in the reduced dataset.

The reduced dataset includes six inputs which are the VIX-index, MACD-indicator, spread between daily high and low prices, corporate interest rate spread, lagged stock returns and returns from the DAX-index. The data from the previous three trading days are used for each predictor. The total amount of inputs in the reduced dataset is thereby 18 (6×3). Several other choices for both the composition of predictors and lag lengths were also considered.

[4]For more information about the relative influence measure see Breiman (2001) and Friedman (2001).

[5]See the appendix for further details about the indicators.

3.2 Tuning parameter optimization

Each of the machine learning methods considered in this thesis involve free parameters that affect the final output of the model. Often the parameters can affect the results quite dramatically as is the case with neural networks for example (Zhang, Patuwo & Hu, 1998). These parameters are usually called tuning parameters since it is up to the end user to finetune the optimal parameters for the particular learning task.

Table 2 summarizes the tuning parameters of the machine learning methods studied in this thesis. A brief description and the notation used for the tuning parameter in the methodological part of this thesis is shown in Table 2. The last column illustrates the considered parameter values. It should be noted that for each method a wider grid search has been conducted in order to find a suitable range for each parameter. Only this smaller interval is depicted in Table 2.

Table 2: Tuning parameters for each method

Method	**Description**	**Notation**	**Values**
k-NN	Number of neighbors	k	$1, 11, 21, ..., 461$
GBM	Number of iterations	M	$1, ..., 1000$
	Tree depth	D	$1, 2, 3$
	Fraction of training points	η_{row}	$0.5, 0.7, 0.9$
	Fraction of predictors	η_{col}	$0.7, 0.9$
RF	Number of trees	B	$100, 300, 500$
	Number of predictors	m	$10, 50, 100, 200, 300$
	Observations in each node	n_{min}	$10, 50, 100, 200, ..., 600$
ANN	Number of hidden units	M	$3, 5, 8, 10, 12, 15, 20$
	Weight decay	λ	$0.1, 0.2, 0.4, 0.6, 0.8, 1$
SVM	Cost parameter	C	$0.01, 0.1, 0.2, ...9$
	Radial kernel parameter	γ	$0.005, 0.01$
	Polynomial kernel, scale	s	$0.005, 0.01$
	Polynomial kernel, degree	d	$2, 3$

Because the number of different machine learning methods considered in this thesis is quite large some simplifications have been done in order to keep the parameter search feasible. Additional finetuning would be available for several methods. There are for example alternative distance measures for the k-NN method and different learning algorithms for the ANN-model. Following the approach of Hastie et al. (2009) the learning rate in the gradient boosting machine algorithm is set as small as possible. The learning rate is held fixed at a value of 0.001. For computational reasons the parameter search is restricted to the parameter values presented in Table 2.

The performance of each model specification is evaluated using the validation accuracy produced by the K-fold cross-validation procedure. In K-fold cross-validation the training sample is split into K independent folds. Each of these K folds is used as a hold-out test set once, while the remaining $K-1$ folds are used for estimating the model. This process is repeated for each fold and the validation accuracy is the average accuracy produced by the K independent folds

$$CV_{Acc} = \frac{1}{K}\sum_{k=1}^{K} Acc_k,$$

where K is the amount of folds and Acc_k is the obtained accuracy when the data points of fold k are used as an independent test set.

10-fold cross-validation is used to estimate the optimal tuning parameters for each method. Table 3 shows the parameter optimization results and depicts the optimal tuning parameter combination for each method.

Table 3: Final model specifications

Method	**Parameters**
k-NN	$k = 331$
GBM	$M = 931, D = 2, \eta_{\text{row}} = 0.9, \eta_{\text{col}} = 0.7$
RF	$B = 500, m = 50, n_{\min} = 300$
ANN	$M = 5, \lambda = 0.8$
SVM	$d = 3, s = 0.005, C = 6.9$

Relatively restricted models are chosen in the cross-validation procedure as can be seen in Table 3. This is not very surprising as the noisy stock market data combined with a complex model can easily lead to overlearning the training data. The limitations in allowed flexibility can be seen for each method. For example, quite a large number of nearest neighbors are used while classifying each data point and the tree-based methods GBM and RF seem to favor quite shallow trees. A fairly small amount of neurons are used in the hidden layer of an ANN-model combined with quite a heavy penalization through the weight decay parameter. The polynomial kernel of degree 3 is used for the support vector machine and the cost parameter is rather small, which results in a wider margin and also supports the finding of restricted models.

The low amount of hidden neurons reported in Table 3 for the ANN is of similar magnitude as the parameter value selected using trial-and-error in Zhong and Enke (2017). The results from a tuning parameter optimization procedure in Kara et al. (2011) also favor the use of a polynomial kernel over the radial basis function for the SVM. The polynomial kernel of degree three is the optimal choice when forecasting the direction of the Turkish stock market as well (Kara et al., 2011). Based on prior knowledge Krauss et al. (2017) end up with the same tree depth for the GBM-model as reported in Table 3.

4 Empirical results

4.1 Estimation results

The complete data sample covering the time period from 12.2.1990 to 5.10.2018 is split into two parts. The training set contains data before the year 2007 and is used for training and validating the models. The test set covering the rest of the data is used as an independent test set to evaluate how well each method performs on a completely unseen dataset. Therefore roughly 58 percent of the complete dataset is used for training and the remaining 42 percent for testing. The test set thus contains 3899 daily observations.

Figure 5 visualizes the daily returns of the S&P 500 index. The horizontal dashed lines show the upper and lower quartiles of the return, which are used as the thresholds in equation (3) to create the multinomial response variable. The vertical gray dashed line illustrates the split into training and testing datasets.

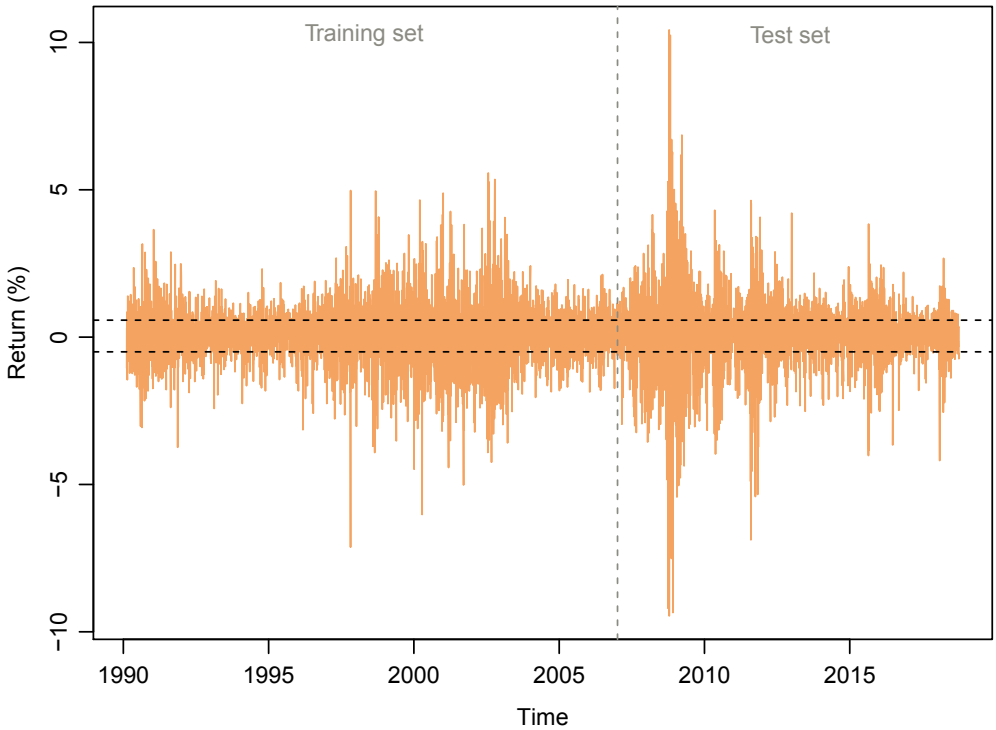

Figure 5: Daily returns of the S&P 500

In order to keep the test set completely independent, the upper and lower quartiles of the return are calculated using only the training data[6]. The benchmark accuracy for the training and validation is one half as the majority class contains fifty percent of the observations. In the test set there are slightly more observations coming from the majority class and the strategy of always predicting the majority class yields the accuracy of 0.526. This should be used as the benchmark accuracy when evaluating the prediction results for the test set.

The estimation results for each machine learning method are presented in Table 4. The first column shows the considered method while the next three columns give the classification accuracies for the training, validation and test sets.

Table 4: Estimation results

Method	**Train**	**Validation**	**Test**
k-NN	0.5204	0.5194	0.5457
GBM	0.5560	0.5294	0.5597
RF	0.6194	0.5304	0.5536
ANN	0.5247	0.5224	0.5558
SVM	0.5463	0.5224	0.5504

The results indicate significant return forecastability as the classification accuracies for the training and validation sets are well above the benchmark of one half for each method. In terms of the training and validation accuracies the results are in favor of using the tree-based methods gradient boosting and random forest, which can utilize the full predictor set. The relatively high training accuracy for the random forest model stands out from the rest, however the validation accuracy is only slightly higher than for GBM. ANN and SVM have a similar validation performance. The simplicity of the nearest neighbor algorithm has led to the lowest training and validation accuracies.

The classification results for the test set indicate how well the models generalize to new data. The observation of return predictability seems to hold even when testing on an unseen dataset as the accuracies for each method are well above the test set

[6]The upper and lower quartiles for the training set are $\{-0.4989\%, 0.5755\%\}$ whereas the quartiles for the entire dataset are $\{-0.4656\%, 0.5723\%\}$.

benchmark performance. The ranking between the machine learning methods based on the classification accuracies for the test set is slightly different from the ranking obtained using the accuracies for the training set. While GBM still outperforms the other machine learning methods random forest reaches only the third highest test accuracy as neural networks show better generalization ability. The fairly large deviance between the training and testing results for the random forest raises questions of potential overfitting. A similar observation can be made when comparing the generalization capabilities of ANN and SVM. The SVM model has better training accuracy but results in slightly lower generalization performance.

4.2 Economic influence

Leitch and Tanner (1991) argue that a model performing well from a statistical point of view does not necessarily imply economic profitability, especially when trading costs are taken into account. In order to evaluate the ability to gain economic profits a real-life trading simulation is conducted. Our trading simulation is similar to those in Pesaran and Timmermann (1995) and Leung et al. (2000) for example. The classification patterns are turned into a trading strategy, which depends on the current ($\hat{R}_{t+1}$) and previous forecasted class ($\hat{R}_t$), as can be seen in Table 5.

Table 5: Trading strategy

		$\hat{\mathbf{R}}_t$		
		1	2	3
$\hat{\mathbf{R}}_{t+1}$	1	Stay out	Sell	Sell
	2	Buy	Hold	Hold
	3	Buy	Hold	Hold

Table 5 shows how the multinomial response variable enables a richer set of possible trading strategies compared to the more commonly studied binary response case. In a traditional market timing setup as presented in Pesaran and Timmermann (1995) an asset allocation decision is made between investing in stocks or in bonds. With the daily trading frequency the returns from investing in the bond market are fairly low and we only consider the options of staying fully invested in stocks or not.

The asset allocation decision between stocks and bonds could easily be incorporated to the trading strategy in Table 5. The complexity of the asset allocation strategy based on the multinomial response could be increased even further. Depending on the predicted class one could allocate 0,100 or say 70 percent of the wealth in stocks and the remaining in the bond market. These more complex strategies are left for further research.

In the benchmark buy-and-hold strategy the entire initial wealth is invested in stocks at the beginning of the period and sold at the end of the period. To get a fair comparison with the buy-and-hold strategy additional wealth can not be invested in stocks. This limits the ability to benefit from large positive returns. In reality the investor would certainly like to increase the amount of wealth invested in stocks if large positive stock returns are expected.

Instead of just staying out from the market it is also possible to profit from the large negative price changes through shortselling. Allowing for shortselling as in Becker and Leschinski (2018) can be seen problematic for several reasons. The risks involved in shortselling are large as the possible losses are basically unlimited. The potential restrictions placed on shortselling by the Securities and Exchange comission (SEC) during high market turmoil can also be seen problematic. One such event took place during the financial crisis as shortselling restrictions were imposed on financial companies (Becker & Leschinski, 2018). Although such restrictions are less of an issue when considering investments in a major stock index the main analysis on economic profitability is conducted without shortselling.

The trading cost is assumed to be a fixed percentage rate, which has been a common choice in the previous literature (see e.g., Pesaran & Timmermann, 1995; Fiévet & Sornette, 2018). A trading cost of 0.1 percent is used in this study. This could be regarded as relatively high since an active individual investor can achieve much lower costs when trading the stocks of a private company instead of the index. Becker and Leschinski (2018) show that the average bid-ask spread on individual stocks in the U.S during the period of 2004-2017 is around 0.05 percent. This is also the level of transaction costs used by Fiévet and Sornette (2018) for example. Naturally the average bid-ask spreads are much lower for the more liquid exchange traded funds (ETF) following the S&P 500 (Hsu, Hsu & Kuan, 2010). Over the past year the bid-ask spread of the worlds largest ETF has ranged between 0.003 and 0.005 percent.[7]

[7] See https://www.etf.com/SPY for more information on the oldest ETF following the S&P 500.

The results from the trading simulation are presented in Table 6. The first column shows the considered machine learning method. The second column illustrates the final wealth level after following the trading strategy presented in Table 5. The initial wealth of 100 is grown to a final wealth of 195.86 using the benchmark buy-and-hold strategy.

Table 6: Results from a trading simulation

Method	**Trading**
k-NN	202.95
GBM	351.54
RF	225.15
ANN	244.51
SVM	210.22

All the considered methods beat the buy-and-hold benchmark. The ranking between the machine learning methods remain the same as based on the test set accuracies. The worst performing nearest neighbor algorithm produces a final wealth that is only slightly higher than the benchmark. The wealth levels of SVM and random forest are also fairly modest compared to the benchmark. The best performing methods based on the test set accuracy outperform the benchmark quite substantially. The final wealth produced by the neural network model is 25 percent higher than the benchmark. The final wealth based on the predictions of the gradient boosting machine yields a final wealth 80 percent higher than the benchmark.

If we allow for shortselling to be used as a tool to profit from the correctly predicted negative returns the final wealth gained using the predictions from a GBM model is 181 percent higher than the benchmark. In shortselling the assets sold short are borrowed from a broker and then returned when closing the short position. It should however be noted that because of these borrowing costs the actual trading costs involved with shortselling can be higher than the considered level of 0.1 percent. The results based on shortselling can therefore be overestimated. We take the approach of Becker and Leschinski (2018) and ignore these additional costs related to shortselling.

In order to see how the wealth is accumulated throughout the test period Figure

6 shows the wealth patterns for the benchmark buy-and-hold strategy and for each of the machine learning methods considered.

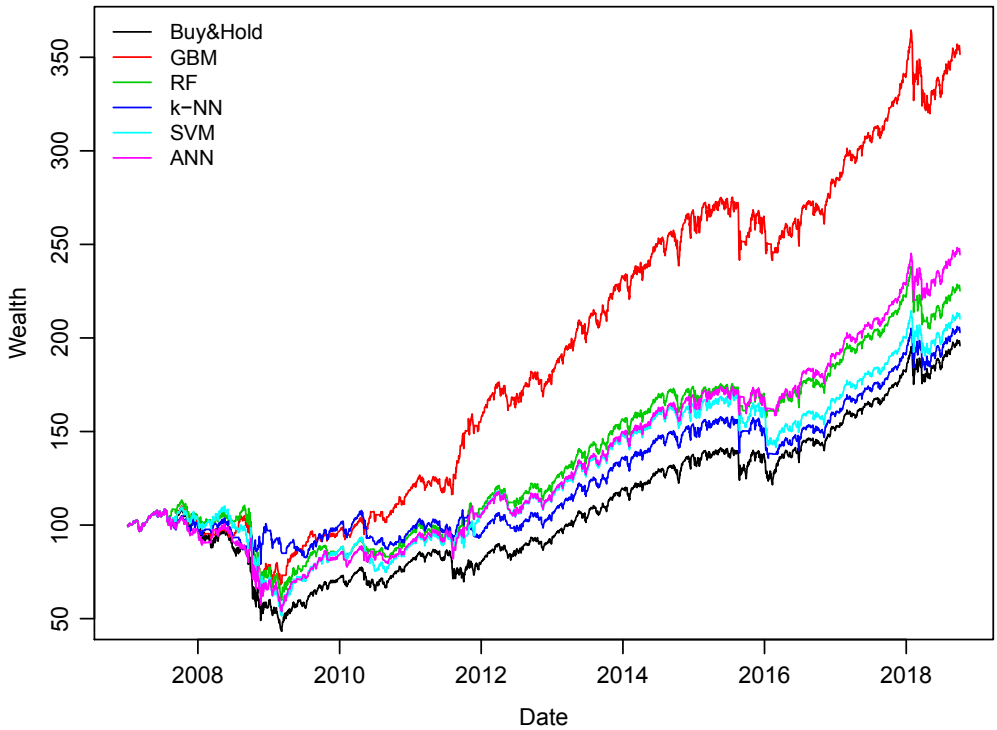

Figure 6: Trading simulation results for each method

There are certain periods of time when the wealth produced by the machine learning methods deviate from the benchmark. All the machine learning methods are able to avoid some of the heavy losses involved in the financial crisis. Some do so better than others as k-NN can avoid majority of the losses while SVM is quite close to the benchmark. Another such period is the debt crisis in the Euro zone, which took place between the years 2010 and 2012. In the year 2010 the wealth level of the GBM model starts to deviate from the rest of the field. A similar observation can be made in the end of the year 2011.

To have a closer look at the daily returns during these events Figure 7 illustrates the correctly predicted large positive and negative returns for the GBM model.

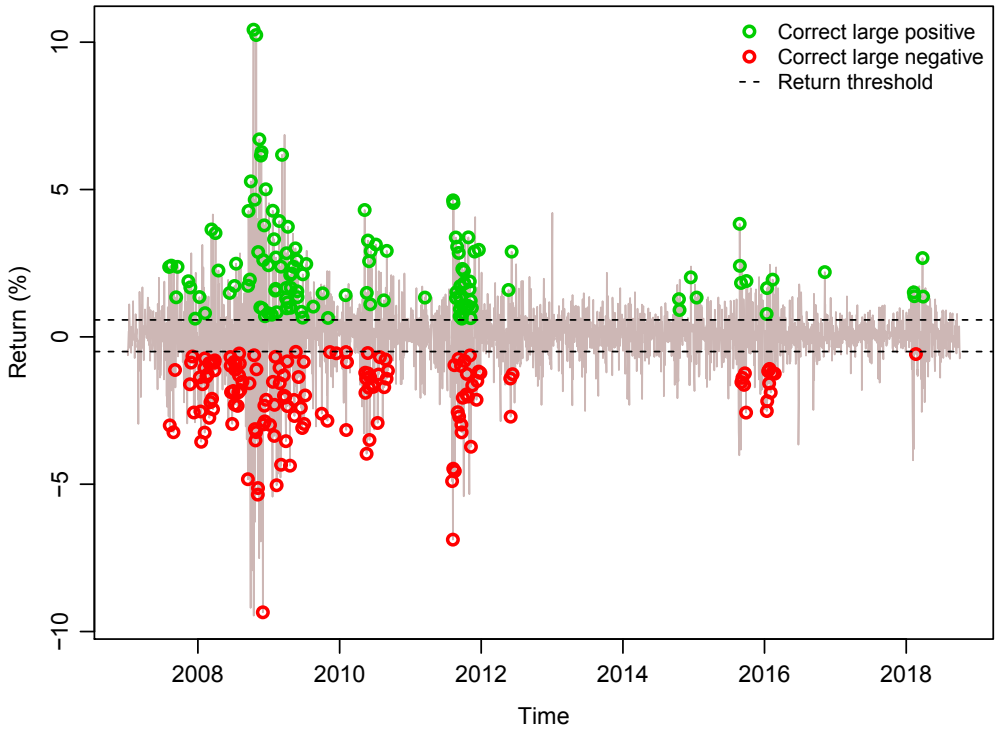

Figure 7: Correctly predicted large absolute returns for the GBM-model

Several interesting observations can be noted from Figure 7. First of all the predictability of large positive and negative stock returns seem to cluster together. The financial crisis and the European debt crisis are periods of higher predictability. This is in line with the results of Krauss et al. (2017) and Fiévet & Sornette (2018). These were also the periods, where the wealth levels of the machine learning methods started to deviate from the benchmark, as was seen in Figure 6. Some predictability is also observed at the beginning of year 2016 when the low oil prices caused worries in the market. All of these events involve high volatility.

Figure 7 also illustrates how the individual correct predictions for the large positive and negative returns can be of very different magnitude. There are correct predictions with return level right above or below the fixed thresholds depicted using the black dashed lines in Figure 7. On the other hand there are positive and negative daily returns close to ten percent. Naturally in terms of the profits obtained from the trading simulation the further apart the correctly predicted observation is from the fixed threshold levels the better. And vice versa for the incorrectly classified

observations.

Regarding the classification accuracy all the correctly predicted observations are considered equally important and thus receive the same weight. This could be a possible explanation why the final wealth levels reported in Table 6 and further illustrated in Figure 6 deviate quite substantially between the different methods. The trading strategy and the return levels could be more closely incorporated to the actual model estimation process. The level of returns or the preferences regarding different correctly and incorrectly predicted classes could be taken into account using caseweights for example. These alternative approaches are left for further research.

It is also interesting to see that despite the daily trading frequency the trading strategy presented in Table 5 could be considered relatively passive. Higher trading activity is observed only during short periods of time involving high volatility. Becker and Leschinski (2018) argue that the assumed fixed trading cost can overestimate the gained profits as the actual bid-ask spreads tend to rise during high market turmoil. The highest bid-ask spread observed by Becker and Leschinski (2018) is around 0.2 percent during a short period of time in the financial crisis. As a sanity check the final wealth gained using the predictions from the GBM-model and the trading cost of 0.2 percent throughout the whole period is still 40 percent higher than the final wealth with the benchmark strategy. The maximum cost level yielding the same final wealth as with the buy-and-hold strategy is 0.33 percent. This is significantly higher compared to the 0.21 percent break even cost reported by Fiévet and Sornette (2018).

5 Conclusions

This thesis introduces a new multinomial classification approach to forecasting daily stock returns of the S&P 500 stock market index. The multinomial approach puts more emphasis on predicting large absolute stock returns instead of the noisy variation around zero. The multinomial approach also provides a larger set of possible trading strategies compared to the more commonly used binary response variable. The classification ability of five different machine learning methods are compared both from the viewpoints of classification accuracy and from the ability to generate economic profits in a real-life trading simulation.

The empirical results show how the gradient boosting model is the top-performer among the machine learning methods based on the classification accuracies for both the validation and test sets. The model selection capability of the gradient boosting model also provides important information about the useful predictor variables. The volatility in the stock market as measured by the VIX-index turns out to be the best single predictor. Several technical analysis indicators are also useful when predicting multinomial stock returns.

The validity of the efficient market hypothesis (EMH) is typically tested in a real-life trading simulation. The ability to generate economic profits beyond the passive buy-and-hold strategy when transaction costs are taken into account is seen as a violation of the EMH. All the machine learning methods considered in this thesis are able to beat the benchmark buy-and-hold strategy after accounting for the transaction cost of 0.1 percent. The best performing gradient boosting model produces returns 80 percent higher than the buy-and-hold strategy. The predictability is highest during the market turmoil of the financial crisis and the European debt crisis, which is in line with recent literature.

The current research can be extended in several directions. Now the two fixed return thresholds used to create the multinomial response variable were based on the upper and lower quartiles of the return series. This choice was based on creating well balanced classes, which simplify the comparison between different machine learning methods. Several other choices are also possible and are left for further research. Setting the return thresholds further away from zero may result in increased statistical predictability but the amount of predictions for the large absolute returns decrease. Thereby the trading strategy would become increasingly passive and hence there might be no additional economic value despite statistically superior predictions over the ones presented in this thesis.

There are also various alternative trading strategies that could be used to assess the economic profits generated by different methods. One could for example benefit more from the predictions indicating large positive or negative returns by shortselling or by increasing the wealth invested. Alternatively modern financial products such as the bull and bear certificates could be used to exploit the correctly predicted large absolute returns. Furthermore, the linkage between forecastability based on statistical evaluation criteria and the economic profitability of the trading strategy should be more closely examined. The trading strategy could even be incorporated to the actual model estimation process.

Bibliography

Ang, A. & Chen, J. (2002). Asymmetric correlations of equity portfolios. *Journal of Financial Economics*, 63(3), 443–494.

Atsalakis, G. S. & Valavanis, K. P. (2009). Surveying stock market forecasting techniques - part II: Soft computing methods. *Expert Systems with Applications*, 36(3, Part 2), 5932 – 5941.

Ballings, M., den Poel, D. V., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, 42(20), 7046 – 7056.

Basak, S., Kar, S., Saha, S., Khaidem, L., & Dey, S. R. (2019). Predicting the direction of stock market prices using tree-based classifiers. *The North American Journal of Economics and Finance*, 47, 552 – 567.

Becker, J. & Leschinski, C. (2018). *Directional Predictability of Daily Stock Returns*. Hannover Economic Papers (HEP) dp-624, Leibniz Universität Hannover, Wirtschaftswissenschaftliche Fakultät.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag.

Boonpeng, S. & Jeatrakul, P. (2016). Decision support system for investing in stock market by using oaa-neural network. *2016 Eighth International Conference on Advanced Computational Intelligence (ICACI)*, (pp. 1–6).

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.

Cavalcante, R. C., Brasileiro, R. C., Souza, V. L., Nobrega, J. P., & Oliveira, A. L. (2016). Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, 55, 194 – 211.

Christoffersen, P. F. & Diebold, F. X. (2006). Financial asset returns, direction-of-change forecasting, and volatility dynamics. *Management Science*, 52(8), 1273–1287.

Chung, J. & Hong, Y. (2007). Model-free evaluation of directional predictability in foreign exchange markets. *Journal of Applied Econometrics*, 22(5), 855–889.

Cujean, J. & Hasler, M. (2017). Why does return predictability concentrate in bad times? *The Journal of Finance*, 72(6), 2717–2758.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2), 383–417.

Fiévet, L. & Sornette, D. (2018). Decision trees unearth return sign predictability in the s&p 500. *Quantitative Finance*, (pp. 1–18).

Fix, E. & Hodges, J. L. (1951). Discriminatory analysis, nonparametric discrimination: Consistency properties. *US Air Force School of Aviation Medicine*, Technical Report 4(3), 477+.

Freund, Y. & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96 (pp. 148–156). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28, 337–407.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5), 1189–1232.

Granger, C. W. & Ding, Z. (1996). Varieties of long memory models. *Journal of Econometrics*, 73(1), 61 – 77.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction.* Springer, 2 edition.

Henkel, S., Martin, J. S., & Nardari, F. (2011). Time-varying short-horizon predictability. *Journal of Financial Economics*, 99(3), 560–580.

Hong, Y., Tu, J., & Zhou, G. (2007). Asymmetries in stock returns: Statistical tests and economic evaluation. *The Review of Financial Studies*, 20(5), 1547–1581.

Hsu, C.-W. & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2), 415–425.

Hsu, P.-H., Hsu, Y.-C., & Kuan, C.-M. (2010). Testing the predictive ability of technical analysis using a new stepwise test without data snooping bias. *Journal of Empirical Finance*, 17(3), 471 – 484.

Huang, W., Nakamori, Y., & Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10), 2513 – 2522.

Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert Systems with Applications*, 38(5), 5311 – 5319.

Kendall, M. G. (1953). The analysis of economic time series, part I: Prices. *Journal of the Royal Statistical Society*, 96.

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1), 307 – 319.

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500. *European Journal of Operational Research*, 259(2), 689 – 702.

Leitch, G. & Tanner, J. E. (1991). Economic forecast evaluation: Profits versus the conventional error measures. *The American Economic Review*, 81(3), 580–590.

Leung, M. T., Daouk, H., & Chen, A.-S. (2000). Forecasting stock indices: a comparison of classification and level estimation models. *International Journal of Forecasting*, 16(2), 173 – 190.

Linton, O. & Whang, Y.-J. (2007). The quantilogram: With an application to evaluating directional predictability. *Journal of Econometrics*, 141(1), 250–282.

Longin, F. & Solnik, B. (2001). Extreme correlation of international equity markets. *The Journal of Finance*, 56(2), 649–676.

Maheu, J. M. & McCurdy, T. H. (2004). News arrival, jump dynamics, and volatility components for individual stock returns. *The Journal of Finance*, 59(2), 755–793.

Merton, R. (1981). On market timing and investment performance. i. an equilibrium theory of value for market forecasts. *The Journal of Business*, 54, 363–406.

Neely, C. J., Rapach, D. E., Tu, J., & Zhou, G. (2014). Forecasting the equity risk premium: The role of technical indicators. *Management Science*, 60(7), 1772–1791.

Nyberg, H. (2011). Forecasting the direction of the us stock market with dynamic binary probit models. *International Journal of Forecasting*, 27, 561–578.

Nyberg, H. & Pönkä, H. (2016). International sign predictability of stock returns: The role of the United States. *Economic Modelling*, 58(C), 323–338.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1), 259 – 268.

Pesaran, M. H. & Timmermann, A. (1995). Predictability of stock returns: Robustness and economic significance. *The Journal of Finance*, 50(4), 1201–1228.

Rojas, R. (1996). *Neural Networks: A Systematic Introduction.* New York, NY, USA: Springer-Verlag New York, Inc.

Skabar, A. (2013). Direction-of-change financial time series forecasting using a similarity-based classification model. *Journal of Forecasting*, 32(5), 409–422.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory.* Berlin, Heidelberg: Springer-Verlag.

Welch, I. & Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies*, 21(4), 1455–1508.

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, 14(1), 35 – 62.

Zhong, X. & Enke, D. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, 67, 126 – 139.

Appendix: Full predictor set

Table 7: Full predictor set

Category	Predictor	Transformation
Stock market	Lagged return	$r_{sp500,t} = \log(\frac{P_{sp500,t}}{P_{sp500,t-1}})$
	DAX	$r_{dax,t} = \log(\frac{P_{dax,t}}{P_{dax,t-1}})$
	FTSE	$r_{ftse,t} = \log(\frac{P_{ftse,t}}{P_{ftse,t-1}})$
	High-Low	$hilow_t = P_{high,t} - P_{low,t}$
	Trade volume	$tvol_t = \log(\frac{Vol_t}{Vol_{t-1}})$
	Market capitalization	$mcap_t = \log(\frac{Cap_t}{Cap_{t-1}})$
	Lagged response	-
	Squared return	$r_t^2 = r_{sp500,t}^2$
	Skew	$skew_t = r_{sp500,t}^3$
	Kurtosis	$kurt_t = r_{sp500,t}^4$
Interest rates	Fed funds rate	$ff_t = fedf_t - fedf_{t-1}$
	3-mth Tbill	$3mth_t = 3m_t - 3m_{t-1}$
	10-yr bond	$10yr_t = 10y_t - 10y_{t-1}$
	30-yr bond	$30yr_t = 30y_t - 30y_{t-1}$
	Moody's Aaa	$Aaa_t = yAaa_t - yAaa_{t-1}$
	Moody's Baa	$Baa_t = yBaa_t - yBaa_{t-1}$
	Term spread	$ts_t = 10y_t - 3m_t$
	Long spread	$ls_t = 30y_t - 10y_t$
	Moody's spread	$corp_t = yBaa_t - yAaa_t$
	Corporate vs Government	$Aaa10y_t = yAaa_t - 10y_t$
	TED-spread	-

*Table is continued on the next page.

Table 7: Full predictor set (continued)

Category	Predictor	Transformation
Exchange rates	Major currencies indx	$curr_t = \log(\frac{Major_t}{Major_{t-1}})$
	USD / GBP	$usdgbp_t = \log(\frac{\$to£_t}{\$to£_{t-1}})$
Commodities	Copper	$copper = \log(\frac{P_{cop,t}}{P_{cop,t-1}})$
	Oil	$oil = \log(\frac{P_{oil,t}}{P_{oil,t-1}})$
	Gold	$gold = \log(\frac{P_{gold,t}}{P_{gold,t-1}})$
	Silver	$slver = \log(\frac{P_{slver,t}}{P_{slver,t-1}})$
Volatility	VIX-index	-
	MOVE-index	-
Macro	ADS-index	-
Technical analysis	Moving average	$ma_t = n^{-1} \sum_{i=0}^{n-1} P_{sp500,t-i}$
	Momentum	$mom_t = P_{sp500,t} - P_{sp500,t-(n-1)}$
	Stochastic %K	$k_t = \frac{P_t - L_n}{H_n - L_n} \times 100$
	Stochastic %D	$d_t = n^{-1} \sum_{i=0}^{n-1} k_{t-1}$
	Relative strength index	$rsi_t = \frac{up}{up+down} \times 100$
	LW %R	$rperc_t = \frac{H_n - P_t}{H_n - L_n} \times 100$
	MACD	see Kara et al. (2011)

*For simplicity n is assumed to be 10 for all the indicators. H_n is the highest high in the previous n-1 days whereas L_n is the lowest low. up is the sum of positive price changes and low the sum of negative price changes in the previous n-1 days.