Krauss, Christopher; Do, Xuan Anh; Huck, Nicolas

Working Paper

Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500

FAU Discussion Papers in Economics, No. 03/2016

Provided in Cooperation with:
Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics

Suggested Citation: Krauss, Christopher; Do, Xuan Anh; Huck, Nicolas (2016) : Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500, FAU Discussion Papers in Economics, No. 03/2016, Friedrich-Alexander-Universität Erlangen-Nürnberg, Institute for Economics, Erlangen

This Version is available at:
http://hdl.handle.net/10419/130166

Discussion Papers in Economics

No. 03/2016

Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500

Christopher Krauss
University of Erlangen-Nürnberg

Xuan Anh Do
University of Erlangen-Nürnberg

Nicolas Huck
ICN Business School - CEREFIGE

ISSN 1867-6707

Friedrich-Alexander-Universität Erlangen-Nürnberg
Institute for Economics
http://www.iwqw.wiso.uni-erlangen.de/forschung/iwqw-discussion-paper-series.shtml

Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500

Christopher Krauss[a,1], Xuan Anh Do[b,1], Nicolas Huck[c,1]

[a] *University of Erlangen-Nürnberg, Department of Statistics and Econometrics, Lange Gasse 20, 90403 Nürnberg, Germany*

[b] *University of Erlangen-Nürnberg, Department of Statistics and Econometrics, Lange Gasse 20, 90403 Nürnberg, Germany*

[c] *ICN Business School - CEREFIGE, 13 rue Michel Ney, 54037 Nancy Cedex, France*

Abstract

In recent years, machine learning research has gained momentum: New developments in the field of deep learning allow for multiple levels of abstraction and are starting to supersede well-known and powerful tree-based techniques mainly operating on the original feature space. All these methods can be applied to various fields, including finance. This article implements and analyses the effectiveness of deep neural networks (DNN), gradient-boosted-trees (GBT), random forests (RAF), and a combination (ENS) of these methods in the context of statistical arbitrage. Each model is trained on lagged returns of all stocks in the S&P 500, after elimination of survivor bias. From 1992 to 2015, daily one-day-ahead trading signals are generated based on the probability forecast of a stock to outperform the general market. The highest k probabilities are converted into long and the lowest k probabilities into short positions, thus censoring the less certain middle part of the ranking. Empirical findings are promising. A simple ensemble consisting of one deep neural network, one gradient-boosted tree, and one random forest produces out-of-sample returns exceeding 0.45 percent per day for $k = 10$, prior to transaction costs. Irrespective of the fact that profits are declining in recent years, our findings pose a severe challenge to the semi-strong form of market efficiency.

Keywords: Statistical arbitrage, deep learning, gradient-boosting, random forests, ensemble learning

Email addresses: christopher.krauss@fau.de (Christopher Krauss), anh.do@fau.de (Xuan Anh Do), nicolas.huck@icn-groupe.fr (Nicolas Huck)

[1]The authors have benefited from many helpful discussions with Ingo Klein, Benedikt Mangold, and Johannes Stübinger.

1. Introduction

Statistical arbitrage or *StatArb* in Wall Street sobriquet, is an umbrella term for quantitative trading strategies generally deployed within hedge funds or proprietary trading desks. It encompasses strategies with the following features "(i) trading signals are systematic, or rules-based, as opposed to driven by fundamentals, (ii) the trading book is market-neutral[2], in the sense that it has zero beta with the market, and (iii) the mechanism for generating excess returns is statistical" (Avellaneda and Lee, 2010, p. 761). Following (Lo, 2010, p. 260), this involves "large numbers of securities (hundreds to thousands, depending on the amount of risk capital), very short holding periods (measured in days to seconds), and substantial computational, trading, and information technology (IT) infrastructure". The underlying models are highly proprietary and - for obvious reasons - not accessible to researchers or the general public (Khandani and Lo, 2011). Typical approaches range from plain vanilla pairs trading in the spirit of Gatev et al. (2006) to sophisticated, nonlinear models from the domains of machine learning, physics, mathematics, and others (Pole, 2008). In contrast, classical financial research is primarily focused on identifying capital market anomalies with high explanatory value. As such, standard methodology relies on linear models or (conditional) portfolio sorts. Jacobs (2015) provides a recent overview of 100 capital market anomalies - most of them are based on monthly data and not a single one employs advanced methods from statistical learning. We may thus carefully state that a gap is evolving between academical finance on the one hand, and the financial industry on the other hand. Whereas the former provide explanations for capital market anomalies on a monthly basis, the latter are prone to deploy black-box methods on the short-term for the sake of profitability. This point can be illustrated with The Journal of Finance, one of the leading academic journals in that field. A search for "neural networks" only produces 17 references whereas the journal has published about two thousand articles during the last thirty years. An even more limited number of articles uses neural network techniques in their empirical studies.

With our manuscript, we attempt to start bridging this gap. In particular, we develop a short-term statistical arbitrage strategy for the S&P 500 constituents. For this purpose, we deploy several powerful methods inspired by the latest trends in machine learning. First, we use deep neural networks (DNN) - a type of highly-parametrized neural network composed of multiple hidden layers, thus allowing for feature abstraction. Its popularization has "dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains" (LeCun et al., 2015, p. 436). The classification of handwritten digits is a standard task and test for these methods. With only 5000 training

[2] *StatArb*, like in this article, also includes dollar-neutral portfolios.

samples of 28-by-28 pixels (784 inputs, 256 levels of grey), the error rate of a DNN is lower than one percent.[3]

In 1997, Deep Blue, a chess-playing computer, defeated the reigning world champion Garry Kasparov. In contrast, the Asian game "Go" is much more complex and has long been considered an Everest for artificial intelligence research. As explained in Allis (1994) and in Silver et al. (2016), such games of perfect information may be solved by recursively computing the optimal value function in a search tree. The number of possible moves can be written as b^d, where b is the game's breadth and d its length. Chess involves "only" $35^{80} (\approx 3.3 * 10^{125})$ moves whereas there are $250^{150} (\approx 4.9 * 10^{359})$ for the game of Go. An exhaustive search is unfeasible - the tree must be reduced. During the first months of 2016, AlphaGo, a Go-playing computer based on deep neural networks and Monte Carlo tree search, has successfully defeated the European Go champion of the years 2013, 2014, and 2015. The algorithm is presented in Silver et al. (2016). In March 2016, a refined version of the program has won a five game Go match against Lee Sedol - one of the best human players of all times (The Economist, 2016).

Second, we employ gradient-boosted trees. Boosting is "one of the most powerful learning ideas introduced in the last twenty years" (Hastie et al., 2009, p. 337). Essentially, it is a procedure for combining many weak learners into one strong learner. In our case, we apply boosting to shallow classification trees. Third, we rely on random forests, "a substantial modification of bagging that builds a large collection of de-correlated trees" (Hastie et al., 2009, p. 587). Fourth, we combine the latter three methods to a simple ensemble.

We train these models with lagged returns of all S&P 500 index constituents and forecast the probability for each stock to outperform the general market. For each day from December 1992 until October 2015, all constituents are ranked according to their out-of-sample probability forecast in descending order. The top k stocks are bought and the flop k stocks sold short. For the ensemble of all three models and $k = 10$, we find average raw returns of 0.45 percent per day prior to transaction costs, outperforming deep learning with 0.33 percent, gradient-boosted trees with 0.37 percent, and random forests with 0.43 percent. Due to the high trading frequency, ensemble returns deteriorate to 0.25 percent per day after transaction costs. These results are statistically and economically significant and can only partially be explained by systematic sources of risk. We find particularly strong positive spikes in returns in situations of high market turmoil, e.g., the dot-com bubble or the global financial crisis. Ever since 2001, with increasing popularization of machine learning and rising computing power, we find deteriorating returns, indicating that markets have become more efficient in respect to standard machine learning statistical arbitrage.

[3] See Yann LeCun's website for a ranking.

The remainder of this paper is organized as follows. Section 2 briefly reviews the relevant literature. Section 3 covers the data sample and section 4 the methodology. Section 5 presents the results and discusses key findings in light of the existing literature. Finally, section 6 concludes and provides directions for further research.

2. Literature review

Most relevant for our application are the works of Huck (2009, 2010); Takeuchi and Lee (2013); Moritz and Zimmermann (2014); Dixon et al. (2015), providing initial applications of machine learning techniques to statistical arbitrage.

Huck (2009) develops a statistical arbitrage strategy based on ensembles of Elman neural networks and ELECTRE III, a multi-criteria decision method. His methodology consists of forecasting, outranking, and trading. In the forecasting step, Huck (2009) uses neural networks to generate one week ahead return forecasts $\hat{x}_{i,t+1}|\mathcal{F}_{ij,t}$ for each security i, conditional to the past return information $\mathcal{F}_{ij,t}$ of securities i and j, with $i, j \in \{1, ..., n\}$, where n is the total number of securities in the index. Next, the anticipated spreads between the forecasted returns of securities i and j are collected in an antisymmetric $n \times n$ matrix. In the outranking step, ELECTRE III is used to create an outranking of all stocks based on this input matrix. Given their relative performance, undervalued stocks wind up at the top and overvalued stocks at the bottom of that ranking. In the trading step, the top k stocks of the ranking are bought and the bottom k stocks sold short. After one week, positions are closed and the process is repeated. An empirical application on the S&P 100 constituents from 1992 to 2006 produces weekly excess returns of more than 0.8 percent at 54 percent directional accuracy for $k = 5$. Huck (2010) enhances this approach with multi-step-ahead forecasts.

Takeuchi and Lee (2013) develop an enhanced momentum strategy on the U.S. CRSP stock universe from 1965 until 2009. Specifically, deep neural networks are employed as classifiers to calculate the probability for each stock to outperform the cross-sectional median return of all stocks in the holding month $t + 1$. The feature space is created as follows: For every month t, the authors construct standardized cumulative return time series for the 12 months $t - 2$ through $t - 13$ and the past 20 days approximately corresponding to month t. Together with a dummy variable denoting if the holding period of month $t + 1$ corresponds to January, a total of 33 predictors are created. These are fed into a restricted Boltzmann machine (RBM) to perform feature abstraction from 33 input features to a four-dimensional code. This code is then processed in a standard three-layer feedforward neural network, ultimately returning a probability forecast, indicating if stock s outperforms its cross-sectional median in the holding month $t + 1$. All stocks are ranked according to this probability forecast. The top decile of the ranking is bought and the bottom decile sold

short, producing annualized returns of 45.93 percent in the out-of-sample testing period from 1990 until 2009. Dixon et al. (2015) run a similar strategy in a high-frequency setting with five-minute binned return data. They reach substantial classification accuracy of 73 percent, albeit without considering microstructural effects - which is quintessential in light of high-frequency data.

Moritz and Zimmermann (2014) deploy random forests on U.S. CRSP data from 1968 to 2012 to develop a trading strategy relying on "deep conditional portfolio sorts". Specifically, they use decile ranks based on all past one-month returns in the 24 months prior to portfolio formation at time t as predictor variables. A random forest is trained to predict returns for each stock s in the 12 months after portfolio formation. The top decile is bought and the bottom decile sold short, resulting in average risk-adjusted excess returns of 2 percent per month in a four-factor model similar to Carhart (1997). Including 86 additional features stemming from firm characteristics boosts this figure to a stunning 2.28 percent per month. Highest explanatory power can be attributed to most recent returns, irrespective of the inclusion of additional firm characteristics. In spite of high turnover, excess returns do not disappear after accounting for transaction costs.

Krauss (2015) provides a recent review of more than 90 statistical arbitrage pairs trading strategies, focusing on relative mispricings between two and more securities. Atsalakis and Valavanis (2009) survey over 100 articles employing machine learning techniques for stock market forecasting. Sermpinis et al. (2013) provide further references in this respect.

Given the available literature, our contribution is threefold. First, to our knowledge, this study is unique in deploying three state-of-the-art machine learning techniques and their simple ensemble on a large and liquid stock universe. We are thus able to compare the performance of deep learning to the tree-based methods, to the ensemble and, as a benchmark, to a "simple" feedforward network - thereby deriving relevant insights for academics and practitioners alike. Second, we provide a holistic performance evaluation, following current standards in the financial literature. It reveals that ensemble returns only partially load on systematic sources of risk, are robust in the light of transaction costs, and deteriorate over time - presumably driven by the increasing popularization of machine learning and the advancements in computing power. However, strong positive returns can still be observed in recent years at times of high market turmoil. Third, we focus on a daily investment horizon instead of monthly frequencies, allowing for much more training data and for profitably exploiting short-term dependencies. All of the above contribute towards bridging the gap between academic and professional finance, making this study relevant for both parties alike.

3. Data and software

3.1. Data

For the empirical application, we opt for the S&P 500. As in Krauss and Stübinger (2015), our choice is motivated by computational feasibility, market efficiency, and liquidity. The S&P 500 consists of the leading 500 companies in the U.S. stock market, accounting for approximately 80 percent of available market capitalization (S&P Dow Jones Indices, 2015). This highly liquid subset serves as a true acid test for any trading strategy, given high investor scrutiny and intense analyst coverage. We proceed along the lines of Krauss and Stübinger (2015) for eliminating survivor bias. First, we obtain all month end constituent lists for the S&P 500 from Thomson Reuters Datastream from December 1989 to September 2015. We consolidate these lists into one binary matrix, indicating whether the stock is a constituent of the index in the subsequent month or not. Second, for all stocks having ever been a constituent of the index, we download the daily total return indices from January 1990 until October 2015. Return indices reflect cum-dividend prices and account for all further corporate actions and stock splits, making it the most suitable metric for return calculations. Previously reported concerns about Datastream quality by Ince and Porter (2006) are mainly focused on small size deciles. Also, Datastream seems to have reacted in the meantime, see Leippold and Lohre (2012). Hence, besides eliminating holidays, we apply no further sanitization measures.

3.2. Software

Preprocessing and data handling are conducted in `R`, a programming language for statistical computing (R Core Team, 2014). For time series subsetting, we rely on the packages `xts` by Ryan and Ulrich (2014) and `TTR` by Ulrich (2013). For performance evaluation, we employ several routines in the package `PerformanceAnalytics` by Peterson and Carl (2014). Deep neural networks, gradient-boosted trees, and random forests are implemented via H2O, a Java-based platform for fast, scalable, open source machine learning, currently deployed in more than 2000 corporations (Candel et al., 2016). Part of the communication between `R` and H2O is implemented with Windows PowerShell.

4. Methodology

Our methodology consists of four steps. First, we split our entire data in non-overlapping training and trading sets. Training sets are required for in-sample training of the specific models and trading sets for their out-of-sample application. Second, for each of these training-trading sets, we generate the feature space necessary for making predictions. Third, we train DNNs, GBTs, and RAFs on each of the training sets. Fourth, we use these models

and a simple ensemble to make out-of-sample predictions on the corresponding trading sets. Stocks are ranked according to these predictions and traded accordingly. This section follows the four step logic outlined above.

4.1. Generation of training and trading sets

In our application to daily data, we set the length of the in-sample training window to 750 days (approximately three years) and the length of the subsequent out-of-sample trading window to 250 days (approximately one year). This choice is motivated by having a sufficient number of training examples available for estimating the models presented in subsection 4.3. We move the training-trading set forward by 250 days in a sliding-window approach, resulting in 23 non-overlapping batches to loop over our entire data set from 1990 until 2015. Let n denote the number of stocks in the S&P 500 at the end of the training period, having full historical data available, i.e., no missing prices in the prior 750 days. Typically, n is close to 500. As, such, for daily data, a training set consists of approximately $500 \cdot 750 = 375000$ and a trading set of approximately 125000 examples.

4.2. Feature generation

For each training-trading set, we generate the feature space (input) and the response variable (output) as follows:

Input: Let $P^s = (P_t^s)_{t \in T}$ denote the price process of stock s, with $s \in \{1, \ldots, n\}$. Then, we define the simple return $R_{t,m}^s$ for each stock s over m periods as

$$R_{t,m}^s = \frac{P_t^s}{P_{t-m}^s} - 1. \tag{1}$$

In our application to daily data, we consider $m \in \{\{1, \ldots, 20\} \cup \{40, 60, \ldots, 240\}\}$. In other words, we follow Takeuchi and Lee (2013) and first focus on the returns of the first 20 days, approximately corresponding to one trading month. Then, we switch to a lower resolution and consider the multi-period returns corresponding to the subsequent 11 months. In total, we thus count 31 features, corresponding to one trading year with approximately 240 days.

Output: We construct a binary response variable $Y_{t+1,1}^s \in \{0, 1\}$ for each stock s. The response $Y_{t+1,1}^s$ is equal to one (class 1), if the one-period return $R_{t+1,1}^s$ of stock s is larger than the corresponding cross-sectional median return computed over all stocks and zero otherwise (class 0). We construct a classification instead of a regression problem, as the literature suggests that the former performs better than the latter in predicting financial market data (Leung et al., 2000; Enke and Thawornwong, 2005). However, please note that we forecast a probability $\mathcal{P}_{t+1}^s$ for each stock s to outperform the cross-sectional median in period $t+1$, which we then post-process.

By approximation, our training sets consist of 375000×32 matrices and our trading sets of 125000×32 matrices.

4.3. Model training

4.3.1. Deep neural networks

This brief description of DNNs follows Candel et al. (2016); Dixon et al. (2015). A deep neural network consists of an input layer, one or more hidden layers, and an output layer, forming the topology of the net. The input layer matches the feature space, so that there are as many input neurons as predictors. The output layer is either a classification or regression layer to match the output space. All layers are composed of neurons, the basic units of such a model. In the classical feedforward architecture, each neuron in the previous layer l is fully connected with all neurons in the subsequent layer $l+1$ via directed edges, each representing a certain weight. Also, each neuron in a non-output layer of the net has a bias unit, serving as its activation threshold. As such, each neuron receives a weighted combination α of the n_l outputs of the neurons in the previous layer l as input,

$$\alpha = \sum_{i=1}^{n_l} w_i x_i + b, \tag{2}$$

with w_i denoting the weight of the output x_i and b the bias. The weighted combination α of (2) is transformed via some activation function f, so that the output signal $f(\alpha)$ is relayed to the neurons in layer $l+1$. Following Goodfellow et al. (2013), we use the maxout activation function $f : \mathbb{R}^2 \to \mathbb{R}$,

$$f(\alpha_1, \alpha_2) = \max(\alpha_1, \alpha_2), \tag{3}$$

receiving inputs from two separate channels with its own weights and biases. Our choice is motivated by the fact that maxout "activation works particularly well with dropout" (Candel et al., 2016, p. 12) - a modern regularization technique.

For the entire network, let W be the collection $W = \bigcup_{l=1}^{L-1} W_l$, with W_l denoting the weight matrix that connects layers l and $l+1$ for a network of L layers. Analogously, let B be the collection $B = \bigcup_{l=1}^{L-1} b_l$, with b_l denoting the column vector of biases for layer l. The collections W and B fully determine the output of the entire DNN. Learning is implemented by adapting these weights in order to minimize the error on the training data. In particular, the objective is to minimize some loss function $\mathcal{L}(W, B|j)$ for each training example j. Since we are dealing with a classification problem, the loss function is cross-entropy,

$$\mathcal{L}(W, B|j) = -\sum_{y \in \mathcal{O}} \left(ln\left(o_y^{(j)}\right) t_y^{(j)} + ln\left(1 - o_y^{(j)}\right)\left(1 - t_y^{(j)}\right)\right), \tag{4}$$

with y representing the output units and $\mathcal{O}$ the output layer. This loss function is minimized by stochastic gradient descent, with the gradient of the loss function $\nabla\mathcal{L}(W, B|j)$ being calculated via backpropagation. In the course of this optimization, we take advantage

of two advanced methods via H2O. First, we use dropout - a modern form of regularization introduced by Srivastava et al. (2014). Thereby, each neuron suppresses its activation with a certain dropout probability during forward propagation for a given training example. As such, instead of one architecture, effectively 2^N architectures are trained, with N denoting the number of training examples. The resulting network thus represents an ensemble of an exponentially large number of averaged models. This regularization method helps to avoid overfitting and improves generalization abilities. Second, we use an advanced optimization routine in H2O called ADADELTA (Candel et al., 2016; Zeiler, 2012), combining the advantages of momentum learning and rate annealing. The former aids in avoiding local minima and the latter helps in preventing "optimum skipping" in the optimization landscape (Zeiler, 2012).

"The design of an ANN is more of an art than a science" (Zhang et al., 1998, p. 42), and tuning parameters are often determined via computationally highly intensive hyperparemeter optimization routines and cross-validation. Instead, we opt for a pragmatic approach and fix the tuning parameters based on the literature. First, let us describe the topology of the net with the following code: I-H1-H2-H3-O. I denotes the number of input neurons, H1, H2, and H3 the number of hidden neurons in hidden layers 1, 2, 3, and O the number of output neurons. In this respect, we choose a 31-31-10-5-2 architecture. The input layer matches the input space with 31 features. Overfitting is a major issue: researchers have provided empirical rules to restrict the number of hidden nodes. Of course, none of theses heuristics works well for each and every problem. A popular rule to set the number of neurons in the first hidden layer of a feedforward network is to use as many neurons as there are inputs. We follow this recommendation in our application. Via the second and third hidden layer, we introduce a bottleneck, enforcing a reduction in dimensionality in line with Takeuchi and Lee (2013); Dixon et al. (2015). The output layer matches the binary output space. This configuration requires the estimation of 2746 parameters, so we have more than 136 training examples per parameter, yielding robust results.[4] Second, we perform regularization. In particular, we use a hidden dropout ratio of 0.5, which "seems to be close to optimal for a wide range of networks and tasks" (Srivastava et al., 2014, p. 1930) and an input dropout ratio of 0.1, again in line with the suggestions in Srivastava et al. (2014). Also, we perform slight L1 regularization with shrinkage parameter $\lambda_{DNN} = 0.00001$. Third, we train with

[4]Due to the two separate channels of the maxout activation function, we have the following number of weights: I-H1: 62×31; H1-H2: 20×31; H2-H3: 10×10; H3-O:2×5. The number of biases can be calculated with the same logic: I: 62×1; I-H1: 20×1; H1-H2: 10×1; H2-H3: 2×1. Summing up all products leads to 2746 parameters. The output layer has no biases and a softmax activation function. Given 375000 training examples, we thus have 136 training examples per parameter.

400 epochs, i.e., we pass 400 times over the training set, as in Huck (2009). For the sake of reproducibility, we set the seed to one, run all calculations on a single core to suppress hardware-based stochastics, and leave all potential further tuning parameters at their H2O default values.

At this stage, we would like to point out that our network is still relatively small with only 31 inputs and 2746 parameters. Deep learning allows for large-scale models with thousands of features and millions of parameters, offering significant potential for further studies. However, for starting to bridge the gap between academic and professional finance, our model is sufficient, computationally not too costly, and exhibits state-of-the-art features, i.e., dropout regularization, maxout activation, and ADADELTA optimization.

4.3.2. Gradient-boosted trees

Boosting is introduced with the seminal paper of Schapire (1990), describing a method for "converting a weak learning algorithm into one that achieves arbitrarily high accuracy" (Schapire, 1990, p. 197). This method is formalized in the algorithm "AdaBoost" of Freund and Schapire (1997), originally applied to classification problems. Boosting works by sequentially applying weak learners to repeatedly re-weighted versions of the training data (Hastie et al., 2009). After each boosting iteration, misclassified examples have their weights increased, and correctly classified examples their weights decreased. Hence, each successive classifier focuses on examples that have been hard to classify in the previous steps. After a number of iterations M_{GBT}, the predictions of the series of weak classifiers are combined by a weighted majority vote into a final prediction. Stochastic gradient boosting is a variation introduced by Friedman (2002), where we sample - without replacement - a subset of the training data upon each iteration to fit the base learner. We use a slightly different approach and select m_{GBT} features at random from the p features upon every split. This subsampling procedure increases computational efficiency, generally improves performance, and decorrelates the trees. We use H2O's implementation of AdaBoost, deploying shallow decision trees as weak learners. For further details, see Click et al. (2016). We have four parameters to set: The number of trees or boosting iterations M_{GBT}, the depth of the tree J_{GBT}, the learning rate λ_{GBT}, and the subset of features to use at each split, i.e., m_{GBT}. Boosting may potentially overfit, if M_{GBT} is too large, so we fix the number of iterations to 100 - a very conservative value compared to examples provided in the standard literature, as in Hastie et al. (2009). Boosting relies on weak learners, i.e., shallow trees, which generally result in the highest performance (Click et al., 2016). As stumps with only one split allow for no variable interaction effects, we settle for a value of $J_{GBT} = 3$, allowing for two-way interactions. Learning rate and number of trees are in an inverse relationship given constant error rates. Hastie et al. (2009) suggest learning rates smaller than 0.1. Taking into account

the low number of trees, we settle for the upper end of the spectrum and fix λ_{GBT} at 0.1. For m_{GBT}, we use 15, i.e., half of the available feature space - a share motivated by Friedman (2002). All other tuning parameters are at their default values and the seed is fixed to one.

4.3.3. Random forests

In the case of boosting, we successively fit shallow decision trees, each taking into account the classification error of the previous trees to build a strong ensemble of weak learners. In contrast, random forests consist of many deep but decorrelated trees built on different samples of the data. They have been introduced by Breiman (2001) and feature high popularity as they are simpler to deploy than boosting. The algorithm to grow a random forest is relatively simple. For each of the B_{RAF} trees in the random forest, we first draw a random subset from the original training data. Then, we grow a modified decision tree to this sample, whereby we select m_{RAF} features at random from the p features upon every split. We grow the tree to the maximum depth of J_{RAF}. The final output is an ensemble of B_{RAF} random forest trees, so that classification can be performed via majority vote. Subsampling substantially reduces variance of (low bias) trees and the random feature selection decorrelates them. We have three tuning parameters, i.e., the number of trees B_{RAF}, their maximum depth J_{RAF}, and the number of features to randomly select m_{RAF}. Random forests are not prone to overfit, so we can choose a high B_{RAF} of 1000 trees. We fix the maximum depth J_{RAF} at 20, a default value in machine learning allowing for substantial higher order interactions (H2O, 2016). Regarding the feature subsampling, we typically choose $m_{RAF} = \lfloor \sqrt{p} \rfloor$ (James et al., 2014). Again, the seed is set to one and all further parameters at their default values.

4.3.4. Equal-weighted ensemble

In addition to DNNs, GBTs, and RAFs, we also use a simple ensemble of the latter. In particular, let $\hat{\mathcal{P}}^{s,ML}_{t+1,1}$ denote the probability forecast of a learning algorithm ML that stock s outperforms its cross-sectional median in period $t+1$, with $ML \in \{DNN, GBT, RAF\}$. We define the ensemble prediction as

$$\hat{\mathcal{P}}^{s,ENS}_{t+1,1} = \frac{1}{3}\left(\hat{\mathcal{P}}^{s,DNN}_{t+1,1} + \hat{\mathcal{P}}^{s,GBT}_{t+1,1} + \hat{\mathcal{P}}^{s,RAF}_{t+1,1}\right), \tag{5}$$

i.e., the simple equal-weighted average of the DNN, GBT, and RAF predictions. According to Dietterich (2000), there are three reasons why ensembles can successfully be employed in machine learning. First, we have a statistical advantage. Each base learner searches the hypothesis space $\mathbb{H}$ to identify the optimal hypothesis. However, in light of limited data compared to the size of $\mathbb{H}$, we usually find several hypotheses in $\mathbb{H}$ that give similar accuracy on the training data. By averaging these hypotheses, we reduce the risk of selecting the wrong

classifier. Second, we have a computational advantage. All of our models perform a local search in hyperparameter space that is prone to get stuck in local optima, i.e., the tree-based methods with the greedy splitting rules and the neural networks with stochastic gradient descent. Averaging across several of these models thus may result in a better approximation of the true, but unknown function. Third, we have a representational advantage. Often, the true, but unknown function is not element of $\mathbb{H}$. Allowing for combinations of several hypotheses from $\mathbb{H}$ considerably increases the solution space of representable functions. The latter may then also include the unknown function. In the econometric field, the combination of forecasts is a major issue (Genre et al., 2013) and simple averaging constitutes a relevant and efficient approach in many cases.

4.4. Forecasting, ranking, and trading

For each period $t+1$, we forecast the probability $\hat{\mathcal{P}}_{t+1,1}^{s,ML}$ for each stock s to outperform its cross-sectional median, with $ML \in \{DNN, GBT, RAF, ENS\}$ and $s \in \{1, \ldots, n\}$. Sorting all stocks over the cross-section in descending order, separately by each of these four forecasts, results in four rankings - corresponding to the DNN, GBT, RAF, and ENS forecasts. At the top, we find the most undervalued stocks according to the respective learning algorithm and at the bottom the most overvalued stocks with the lowest probability to outperform the cross-sectional median in period $t+1$. In consequence, we go long the top k stocks of each ranking, and short the bottom k stocks, with $k \in \{1, \ldots, \lfloor n/2 \rfloor\}$. By censoring the middle part of the ranking as in Huck (2009, 2010), we exclude the stocks with highest directional uncertainty from trading.

5. Results

5.1. General results

At first, we analyze the performance of the portfolios consisting of the top k stocks, with $k \in \{10, 50, 100, 150, 200\}$. They are compared in terms of returns per day prior to transaction costs, standard deviation, and daily directional accuracy at the portfolio level. Figure 1 depicts the results. For $k = 10$ - a quite diversified portfolio with 10 long and 10 short positions - we observe that the ensemble produces returns of 0.45 percent per day, followed by the random forest with 0.43 percent, the gradient-boosted trees with 0.37 percent, and the deep neural networks with 0.33 percent per day. Directional accuracy follows a similar pattern. These results are line with Huck (2009). Increasing k, i.e., including stocks with higher uncertainty, leads to decreasing returns and directional accuracy. The latter indicator, irrespective of k or the forecasting method, is always greater than 50 percent - an important benchmark for a dollar neutral strategy. Increasing the number of assets leads to decreasing standard deviations - in line with classical portfolio theory.

In summary, the ensemble outperforms all base models in terms of directional accuracy regardless of the level of k - despite its simplicity. Following Hansen and Salamon (1990); Dietterich (2000), there are two necessary and sufficient conditions for an ensemble to achieve higher accuracy than its base learners: First, the base learners need to be diverse, and second, they need to be accurate. Diversity means that the errors of the models exhibit low correlation. Even though we do not explicitly test for this, we combine three vastly different model types - deep neural networks, boosted shallow trees, and decorrelated trees of high depth. As such, it is fair to assume that the base learners are somewhat diverse. In respect to the second condition, we may say that all base learners are accurate, as they achieve more than 50 percent directional accuracy. The combination of three accurate yet diverse base learners leads to superior results in our application.

When focusing on the base learners, we see that random forests achieve higher returns and directional accuracy than gradient-boosted trees. We presume that this outperformance is driven by the high number of decorrelated, deep trees. Rigorous random feature selection makes random forests basically immune to overfitting and very robust to the noisy feature space we are facing. Deep trees allow for high interaction depth between explanatory variables, thus reducing bias. Conversely, both tree-based models perform better than the deep neural network - the most recent advancement in machine learning. Tree-based methods are relatively easy to train. Especially random forests can almost be deployed in a "standard configuration" without the need for extensive hyperparameter tuning. In contrast, neural networks are notoriously difficult to train. It may well be that there are configurations in hyperparameter space to further improve the performance of the DNN, but in a baseline setting without extensive tuning, its performance is inferior to RAF and GBM.

In the following subsections, we focus on the portfolio formed with $k = 10$. We depict results prior to and after incorporating transaction costs of 0.05 percent per share per half-turn, following Avellaneda and Lee (2010). This pragmatic estimate is deemed viable in light of our high-liquidity stock universe and a high-turnover statistical arbitrage strategy.

First, we evaluate strategy performance in terms of return distribution, value at risk, risk-return characteristics, and exposure to common sources of systematic risk. The majority of selected performance metrics is detailed in Bacon (2008). Second, we evaluate returns over time. Third, we run further analyses, i.e., we assess variable importances, split the portfolio by industries, and perform a sensitivity analysis to show robustness to hyperparameter selection.

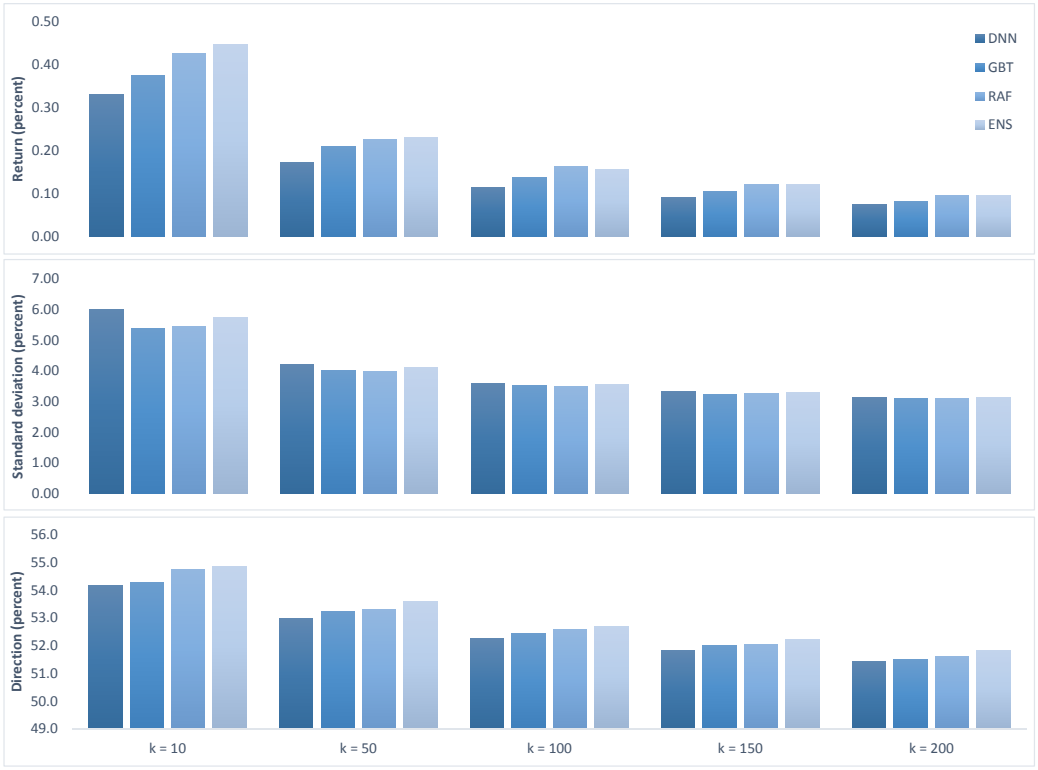

Figure 1: Daily performance metrics for long-short portfolios of different sizes: Mean return, standard deviation, and directional accuracy from December 1992 until October 2015.

5.2. Strategy performance

Table 1 reports daily return characteristics for the $k = 10$ portfolio from December 1992 until October 2015. We observe statistically and economically significant returns - even when factoring in transaction costs. In contrast to the general market, all strategy variants exhibit positive skewness - a positive property for potential investors. In line with the theory, returns are strongly leptokurtic, driven by large outliers - see the minimum and maximum statistics. Return contribution of the long-leg ranges between 65 and 70 percent prior to transaction costs, indicating that the strategies profit from the long and the short investments.

Following the RiskMetrics approach of Mina and Xiao (2001), we analyze the tail risk of the strategies. Historical one percent value at risk (VaR 1%) fluctuates between -5.9 and -6.9 percent - about twice the level of the general market. In this respect, RAFs exhibit the lowest risk and DNNs the highest. Compared to other strategies, such as classical pairs trading, the tail risk is substantial. For example, Gatev et al. (2006) find daily VaR 1% at 1.24 percent for the $k = 5$ and at 0.65 percent for the $k = 20$ portfolio, which is significantly

	Before transaction costs				After transaction costs				
	DNN	GBT	RAF	ENS	DNN	GBT	RAF	ENS	MKT
Mean return	0.0033	0.0037	0.0043	0.0045	0.0013	0.0017	0.0023	0.0025	0.0004
Mean return (long)	0.0022	0.0025	0.0030	0.0029	0.0012	0.0015	0.0020	0.0019	-
Mean return (short)	-0.0011	-0.0013	-0.0013	-0.0015	-0.0001	-0.0003	-0.0003	-0.0005	-
Standard error (NW)	0.0004	0.0003	0.0003	0.0003	0.0004	0.0003	0.0003	0.0003	0.0001
t-Statistic (NW)	8.6159	12.6786	14.9327	13.3962	3.3835	5.8952	7.9104	7.3781	2.8305
Minimum	-0.1916	-0.1487	-0.1622	-0.1681	-0.1936	-0.1507	-0.1642	-0.1701	-0.0895
Quartile 1	-0.0079	-0.0062	-0.0053	-0.0063	-0.0099	-0.0082	-0.0073	-0.0083	-0.0046
Median	0.0025	0.0032	0.0033	0.0034	0.0005	0.0012	0.0013	0.0014	0.0008
Quartile 3	0.0133	0.0133	0.0127	0.0138	0.0113	0.0113	0.0107	0.0118	0.0058
Maximum	0.5475	0.2003	0.3754	0.4470	0.5455	0.1983	0.3734	0.4450	0.1135
Standard deviation	0.0269	0.0217	0.0208	0.0239	0.0269	0.0217	0.0208	0.0239	0.0117
Skewness	2.8547	0.2434	1.7283	2.6837	2.8547	0.2434	1.7283	2.6837	-0.1263
Kurtosis	50.5137	8.2267	29.8174	43.2698	50.5137	8.2267	29.8174	43.2698	7.9791
Historical VaR 1%	-0.0672	-0.0580	-0.0508	-0.0570	-0.0692	-0.0600	-0.0528	-0.0590	-0.0320
Historical CVaR 1%	-0.0929	-0.0831	-0.0721	-0.0786	-0.0949	-0.0851	-0.0741	-0.0806	-0.0461
Historical VaR 5%	-0.0322	-0.0262	-0.0229	-0.0267	-0.0342	-0.0282	-0.0249	-0.0287	-0.0179
Historical CVaR 5%	-0.0544	-0.0462	-0.0406	-0.0449	-0.0564	-0.0482	-0.0426	-0.0469	-0.0277
Maximum drawdown	0.5815	0.4391	0.3454	0.4017	0.9544	0.8425	0.6689	0.7367	0.5467
Calmar ratio	1.8884	3.2219	5.1033	4.6277	0.2813	0.5466	1.0037	0.9903	0.1692
Share with return > 0	0.5713	0.5939	0.6028	0.5883	0.5174	0.5351	0.5423	0.5367	0.5426

Table 1: Daily return characteristics of $k = 10$ portfolio, prior to and after transaction costs for DNN, GBT, RAF, ENS compared to general market (MKT) from December 1992 until October 2015. NW denotes Newey-West standard errors with with one-lag correction.

lower compared to our strategies. Distance-based pairs trading is an equilibrium strategy, prone to construct pairs exhibiting low volatility. However, lower risk comes at a price - classical pairs trading only achieves annualized excess returns of 11 percent, so lower returns go along with lower tail risk. We observe a very similar picture for the conditional value at risk (CVaR). Also, maximum drawdown of all strategies exceeds the market with 55 percent by far. RAFs perform best with 67 percent, whereas DNNs produce a drawdown of 95 percent. The ensemble settles at 74 percent. At first sight, these values are tremendously high, but the Calmar ratio conveys a more moderate picture. It scales annualized return by the absolute value of maximum drawdown, resulting in a value of 99 percent for the ensemble and 17 percent for the market. In other words, it takes approximately one average annual return, or one year of time, to recover from the maximum drawdown in case of the ENS strategy and approximately six annual returns, or six years of of time in case of an investment in MKT.

Table 2 reports annualized risk-return metrics. After transaction costs, annualized returns amount to 73 percent for the ensemble, compared to 67 percent for random forests, 46 percent for gradient-boosted trees, and 27 percent for deep neural networks. All strategies largely outperform the general market with average returns of 9 percent p.a.

Standard deviation ranges between 33 percent (RAF) and 43 percent (DNN) - roughly

	Before transaction costs				After transaction costs				
	DNN	GBT	RAF	ENS	DNN	GBT	RAF	ENS	MKT
Mean return	1.0981	1.4149	1.7627	1.8588	0.2685	0.4605	0.6714	0.7296	0.0925
Mean excess return	1.0446	1.3534	1.6924	1.7861	0.2361	0.4232	0.6287	0.6855	0.0646
Standard deviation	0.4277	0.3438	0.3308	0.3793	0.4277	0.3438	0.3308	0.3793	0.1852
Downside deviation	0.2474	0.2113	0.1847	0.2049	0.2615	0.2250	0.1981	0.2188	0.1307
Sharpe ratio	2.4426	3.9364	5.1166	4.7091	0.5521	1.2310	1.9008	1.8073	0.3486
Sortino ratio	4.4384	6.6956	9.5462	9.0702	1.0268	2.0466	3.3883	3.3337	0.7077

Table 2: Annualized returns and risk measures of $k = 10$ portfolio, prior to and after transaction costs for DNN, GBT, RAF, ENS compared to general market (MKT) from December 1992 until October 2015.

twice the level compared to the general market with 19 percent. The Sharpe ratio is defined as excess return per unit of risk, measured in standard deviations. The ensemble achieves excess returns, which are more than ten times larger than those of the general market, at approximately two times the standard deviation. Hence, the Sharpe ratio of the ensemble with 1.81 is roughly five times higher than that of the general market. It also compares favorably to other statistical arbitrage strategies. Classical pairs trading results in a Sharpe ratio of 0.59 for the top 20 pairs from 1962 until 2002 (Gatev et al., 2006), generalized pairs trading in a Sharpe ratio of 1.44 from 1997 until 2007 (Avellaneda and Lee, 2010), and deep conditional portfolio sorts in a Sharpe ratio of 2.96 from 1968 until 2012 - albeit prior to transaction costs and on a larger and less liquid stock universe (Moritz and Zimmermann, 2014). Huck (2009) achieves a Sharpe ratio of approximately 1.5 with Elman neural networks and ELECTRE III from 1992 until 2006 - also prior to transaction costs. The Sortino ratio scales the returns by their downside deviation. Its advantage lies in the lower partial moment metric, only measuring downside deviations as actual risk (compared to favorable upward deviations). We see that downside deviation ranges between 0.20 (RAF) and 0.26 (DNN), with a value of 0.22 for the ensemble - around 1.7 times the level of the general market. Hence, downside deviations are less expressed for the machine learning strategies compared to those of the general market, leading to favorable Sortino ratios. Across both risk-return metrics, RAF performs best, followed by ENS, GBT, and DNN.

In table 3, the exposure of returns to common sources of systematic risk is evaluated for the ensemble strategy. For simplicity's sake, we focus this analysis on the ensemble strategy for the $k = 10$ portfolio, as it exhibits the highest returns. We perform four regressions: First, we use the Fama-French three-factor model (FF3), following Fama and French (1996). The latter captures exposure to the general market, small minus big capitalization stocks (SMB), and high minus low book-to-market stocks (HML). Second, we enhance this model by a momentum and a short-term reversal factor, as in Gatev et al. (2006). We call this variant Fama-French 3+2-factor model (FF3+2). Third, we use the recently developed Fama-French five-factor model, following Fama and French (2015). It originates from the

	FF3	FF3+2	FF5	FF VIX
(Intercept)	0.0022***	0.0014***	0.0024***	0.0010**
	(0.0003)	(0.0003)	(0.0003)	(0.0003)
Market	0.3271***	0.1759***	0.2172***	0.1903***
	(0.0269)	(0.0278)	(0.0300)	(0.0279)
SMB	−0.0036	−0.0458		−0.0362
	(0.0524)	(0.0493)		(0.0492)
HML	−0.0290	0.2983***		0.3126***
	(0.0515)	(0.0515)		(0.0515)
Momentum		0.3885***		0.3972***
		(0.0355)		(0.0355)
Reversal		0.9474***		0.9387***
		(0.0361)		(0.0361)
SMB5			−0.0689	
			(0.0561)	
HML5			0.2348***	
			(0.0603)	
RMW5			−0.3308***	
			(0.0794)	
CMA5			−0.6639***	
			(0.0911)	
VIX				0.0047***
				(0.0010)
R^2	0.0259	0.1402	0.0381	0.1436
Adj. R^2	0.0254	0.1395	0.0373	0.1427
Num. obs.	5750	5750	5750	5750
RMSE	0.0236	0.0222	0.0234	0.0221

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

Table 3: Ensemble strategy with $k = 10$: Exposure to systematic sources of risk after transaction costs for DNN, GBT, RAF, ENS from December 1992 until October 2015. Standard errors are depicted in parentheses.

three-factor model (FF5), enhanced by two additional factors, i.e., portfolios of stocks with robust minus weak profitability (RMW) and with conservative minus aggressive (CMA) investment behavior. All data related to these factor models are downloaded from Kenneth French's website.[5] In the fourth regression (FF VIX), we enhance the FF3+2 model with the VIX index, the "investor fear gauge" (Whaley, 2000; Fernandes et al., 2014). Specifically, we add the VIX as a dummy variable equaling one if the VIX is greater than 30 - the 90 percent quantile. This threshold signals highly volatile periods corresponding to approximately 10 percent of all trading days. In this regression, the intercept can no longer be interpreted as excess return, given that the dummy variable is not investable.

According to the FF3, the ENS strategy results in a statistically and economically significant daily alpha of 0.22 percent. The returns positively load on the market with a coefficient of 0.33, which is possible, given that the strategy design is dollar neutral, not market neutral. In contrast, SMB and HML factors are not statistically significant. The FF3+2 model

[5]We thank Kenneth R. French for providing all relevant data for these models on his website.

has much higher explanatory power with an adjusted R^2 of 0.14. Daily alpha decreases to 0.14 percent - albeit still statistically and economically significant. We see that additional explanatory content is provided by the momentum and the short-term reversal factor. Surprisingly, in both cases, the strategy exhibits strong and statistically significant positive factor loadings. As such, we may carefully conclude that the machine learning algorithms extract momentum as well as short-term reversal patterns from the data, thus explaining the factor loadings. The FF5 model exhibits the highest alpha of 0.24 percent. We observe positive, significant loadings on the HML factor and negative, significant loadings on the RMW and the CMA factors. Investment behavior seems to exhibit a slight tilt towards glamour stocks with weaker profitability and aggressive investment behavior. The last regression exhibits statistically significant loadings on the VIX, indicating that the ensemble strategy performs better at times of high market turmoil.

Overall, we conclude that the ensemble strategy produces statistically and economically significant daily alphas between 0.14 and 0.24 percent - depending on the employed factor model. Returns partly load on common sources of systematic risk, but not in a uniform manner, thus suggesting an investment behavior that partially incorporates several return-based capital market anomalies.

5.3. Sub-period analysis

Four sub-periods are introduced to provide more detail about the performance and the risk profile of the four strategies over time. Details are provided in figure 2 and table 4.

The first sub-period ranges from 12/92 to 03/01 and corresponds to a period of strong and consistent outperformance, prior to the invention and propagation of the machine learning algorithms employed in this paper. As such, it is no surprise that annualized returns after transaction costs exceed 200 percent at Sharpe ratios of 6.7 for the ensemble strategy. Essentially, we are using powerful techniques that had not been (publicly) available at this point in the past to find and profitably exploit structure in financial time series. It remains unclear if variants of these techniques had already been deployed within the hedge fund industry during that period.

The second sub-period ranges from 04/01 to 08/08 and corresponds to a period of moderation. Annualized returns for the ensemble strategy decline to 22 percent and the Sharpe ratio drops to 0.51 - both after transaction costs. Random forests outperform the ensemble in this period with mean returns of 35 percent p.a. and a Sharpe ratio close to 1. This effect is driven by the weak performance of deep learning, negatively affecting the ensemble. Primarily, the decline may be driven by the advancement of machine learning techniques and an associated increase in market efficiency. Random forests - the strongest base model - have been invented in 2001 by Breiman (2001), and have been popularized in the years

after. Similarly, stochastic gradient boosting - the workhorse behind GBT - originates from the contribution of Friedman (2002), with an earlier version available in Friedman (1999). These high-performing methods are able to detect structure in financial market data at times prior to their invention (sub-period 1), and performance deteriorates in the years that follow (sub-period 2). This effect is exacerbated by the availability of cheaper computing power. Following Moore (1965), the number of transistors on integrated circuits doubles every 12-24 months. Effectively, this exponential increase in computing power allows for efficient computations of powerful machine learning algorithms on a single personal computer ever since the year 2000.

The third sub-period ranges from 09/08 to 12/09 and corresponds to the global financial crisis. We see that all our long-short strategies perform particularly well at such times of high market turmoil. This is in line with Do and Faff (2010); Bogomolov (2013); Huck (2015), showing that liquidity-providing pairs trading strategies fare well in light of the financial crisis. Specifically, for the ensemble strategy, we observe annualized returns of more than 400 percent after transaction costs at a Sharpe ratio of 4.5. The latter is lower than during the first sub-period, given the high volatility during the global financial crisis.

The fourth sub-period ranges from 01/01 to 10/15 and corresponds to a period of deterioration. All base learners and the ensemble produce negative annualized returns ranging between -14 and -25 percent p.a. after transaction costs. We surmise that increasing public availability of powerful machine learning algorithms and decreasing entry barriers in terms of technological investment has led to profits being arbitraged away in recent years. However, we presume that within the hedge fund industry, proprietary machine learning algorithms at high technological scale may still result in outperformance, given the success of our strategies at earlier times. This assumption is corroborated by the fact that all our models still produce positive mean returns prior to transaction costs. As such, some structure is detected, but the effects are not large enough to construct a profitable strategy.

Motivated by the strong returns during the global financial crisis, we analyze yearly and monthly outliers for the ensemble strategy. All returns provided in this paragraph are after transaction costs. Several periods come to our attention. First, we observe particularly strong annual returns of 334 percent in 1999, i.e., the year leading up to the dot-com bubble. These are outstripped in 2000 with annual returns of 545 percent, i.e., the time when the bubble finally busted and technology stocks lost billions in market capitalization. Second, the largest outlier is the year 2008, when annual returns of 681 percent fall together with the outbreak of the global financial crisis. In particular, returns in October 2008, the month following the bust of Lehman Brothers, exceed 100 percent - by far the strongest period from December 1992 until October 2015. Third, October 2011 results in a plus of 35 percent, and

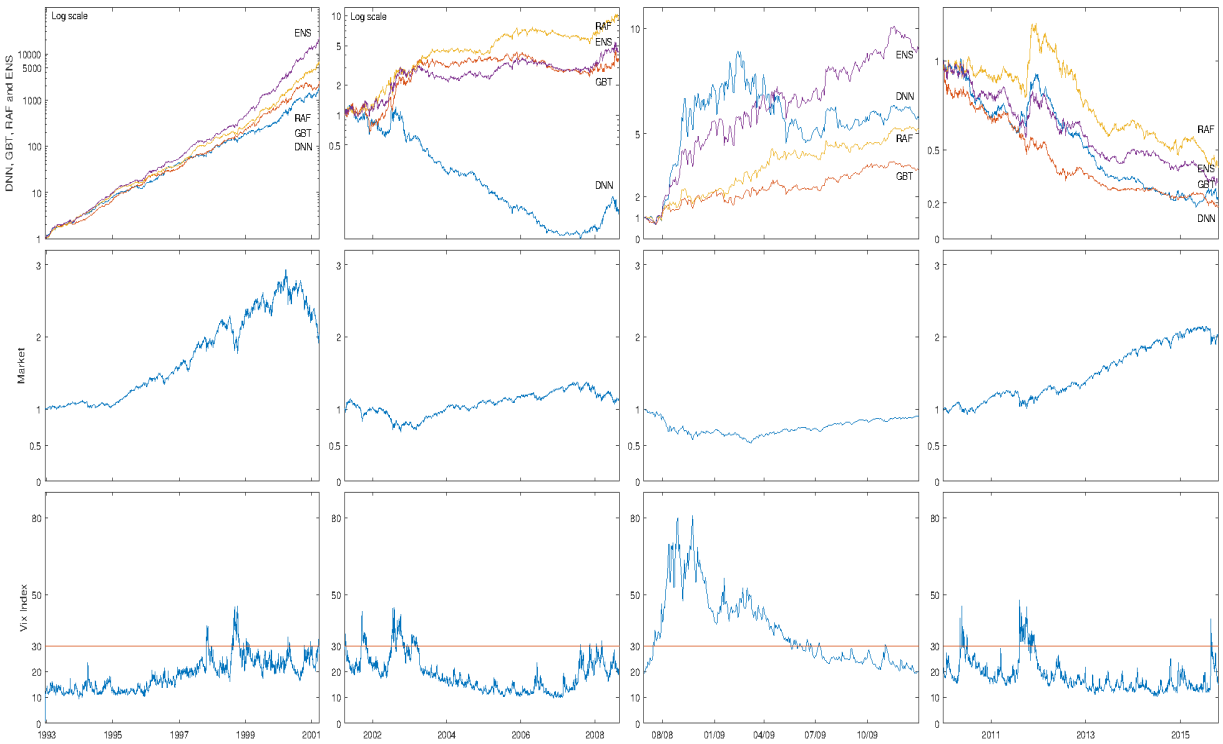

Figure 2: Sub-periods profile of $k = 10$ portfolio, after transaction costs for DNN, GBT, RAF, ENS compared to general market (MKT) and the VIX index from December 1992 until October 2015.

coincides with the peak of the European debt crisis, i.e., when a haircut on Greek bonds was agreed. Given these events, we may carefully surmise that we effectively capture relative mispricings between securities at times of high market turmoil, i.e., when investor attention is diverted from individual stocks to the general market by events of high significance (Jacobs and Weber, 2015). On such occasions, mispricings may occur and can be profitably exploited. The literature on the phenomenon called asymmetric/extreme correlation (Longin and Solnik, 2001) is also linked.

5.4. Further analyses

5.4.1. Variable importances

With all three machine learning algorithms, we can extract variable importance via H2O, i.e., the relative predictive strength of each feature (H2O, 2016). For deep learning, the method of Gedeon (1997) is employed, i.e., the weight matrices connecting the inputs with the first two hidden layers is analyzed to extract variable importance. For the tree-based methods, variable importance is determined by computing the relative influence of each variable, i.e., by assessing whether a particular variable is used during splitting when growing trees, and by how much the loss function improves as a result on average across all trees (H2O, 2016). We extract variable importances for all models across all 23 training sets.

	Before transaction costs				After transaction costs				
	DNN	GBT	RAF	ENS	DNN	GBT	RAF	ENS	MKT
Period 12/92-03/01									
Mean return	3.0789	3.2647	3.7490	4.5055	1.4695	1.5823	1.8762	2.3353	0.1384
Mean excess return	2.8935	3.0709	3.5333	4.2556	1.3570	1.4647	1.7453	2.1836	0.0864
Standard deviation	0.3636	0.3190	0.2900	0.3263	0.3636	0.3190	0.2900	0.3263	0.1581
Sharpe ratio	7.9573	9.6274	12.1857	13.0410	3.7319	4.5919	6.0191	6.6914	0.5466
Maximum drawdown	0.2223	0.2888	0.1332	0.2069	0.2473	0.3519	0.1593	0.2168	0.3080
Calmar ratio	13.8483	11.3029	28.1521	21.7726	5.9429	4.4962	11.7756	10.7723	0.4493
Period 04/01-08/08									
Mean return	0.2107	0.9737	1.2374	1.0228	-0.2688	0.1932	0.3530	0.2229	0.0436
Mean excess return	0.1788	0.9218	1.1786	0.9696	-0.2881	0.1617	0.3173	0.1907	0.0161
Standard deviation	0.4037	0.3578	0.3294	0.3722	0.4037	0.3578	0.3294	0.3722	0.1698
Sharpe ratio	0.4430	2.5762	3.5777	2.6054	-0.7136	0.4521	0.9633	0.5124	0.0945
Maximum drawdown	0.5815	0.4391	0.3200	0.2689	0.9544	0.4889	0.3353	0.3293	0.3870
Calmar ratio	0.3624	2.2173	3.8670	3.8034	-0.2816	0.3952	1.0526	0.6770	0.1126
Period 09/08-12/09									
Mean return	5.0550	2.9330	4.5282	7.3371	2.6658	1.3802	2.3476	4.0519	-0.0704
Mean excess return	5.0388	2.9224	4.5133	7.3148	2.6560	1.3737	2.3386	4.0384	-0.0729
Standard deviation	1.0625	0.6932	0.7578	0.9050	1.0625	0.6932	0.7578	0.9050	0.3932
Sharpe ratio	4.7426	4.2159	5.9559	8.0824	2.4998	1.9818	3.0861	4.4621	-0.1854
Maximum drawdown	0.4276	0.3349	0.3201	0.3139	0.4920	0.3525	0.3288	0.3307	0.4636
Calmar ratio	11.8216	8.7581	14.1448	23.3773	5.4182	3.9149	7.1401	12.2532	-0.1518
Period 01/10-10/15									
Mean return	0.2781	0.2359	0.4180	0.3580	-0.2279	-0.2535	-0.1433	-0.1796	0.1334
Mean excess return	0.2776	0.2354	0.4174	0.3574	-0.2282	-0.2538	-0.1436	-0.1799	0.1330
Standard deviation	0.2371	0.2041	0.1882	0.2044	0.2371	0.2041	0.1882	0.2044	0.1641
Sharpe ratio	1.1710	1.1536	2.2182	1.7485	-0.9626	-1.2434	-0.7631	-0.8802	0.8105
Maximum drawdown	0.2351	0.2554	0.1741	0.1836	0.8245	0.8222	0.6689	0.7034	0.2032
Calmar ratio	1.1832	0.9238	2.4002	1.9494	-0.2764	-0.3083	-0.2142	-0.2553	0.6567

Table 4: Annualized risk-return characteristics per sub-period for DNN, GBT, RAF, ENS.

With the most important variable normalized to an index value of 100, we depict average relative variable importance across all training sets in figure 3 for all models. $R(m)$ refers to the multi-period return calculated over m periods, as in equation (1) - the indices are dropped for simiplicity's sake.

For the tree-based methods, features can - very roughly - be split in three groups. The returns corresponding to the past 4 to 5 days have the highest relative importance, ending up at the top of the ranking. In case of the gradient-boosted trees, we observe an "elbow", with relative importance sharply dropping after $R(4)$. The next group includes the multi-period returns roughly corresponding to the monthly resolution with $m \in \{20, 40, \ldots, 240\}$, intermingled with some medium-term returns with $m \in \{5, \ldots, 10\}$. The third group of returns with $m \in \{11, \ldots, 20\}$ can generally be found towards the bottom of the tree-based rankings. The spread between top and bottom is more expressed for the gradient-boosted trees, since we consider a lower tree depth. In case of the DNN, the ranking is similar at

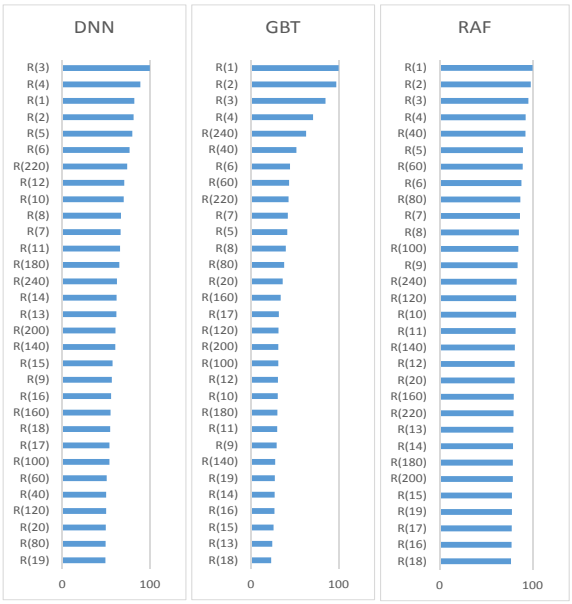

Figure 3: Variable importance extracted from DNN, GBT, RAF from December 1992 until October 2015. Most important variable normalized to 100.

first sight. At the top, we also find the short-term features, ranging from $R(1)$ until $R(6)$. Then, one long-term feature with $R(220)$ follows; the rest of the ranking seems arbitrary.

Subsuming all this information, the following logic seems to apply: First and foremost, the short-term returns corresponding to the past five trading days have the highest explanatory power (group 1), as confirmed by all models. Then, the returns corresponding to the past 12 months follow, with very distant months generally having higher importance than more recent months (group 2). At the bottom of the ranking, we find returns between $R(10)$ and $R(20)$ in case of the tree-based models (group 3).

Interestingly enough, standard capital market anomalies as in Jacobs (2015) are mainly based on returns corresponding to group 2. Our results suggest that there may be significant potential for short-term strategies based on the returns of the past five days.

Industry	Share in data base	Share long	Share short
Basic Materials	6.59	8.06	8.33
Consumer Goods	11.65	10.90	11.00
Consumer Services	14.91	16.36	16.58
Financials	17.07	11.26	9.89
Health Care	8.49	7.85	7.57
Industrials	16.08	11.70	12.04
Oil & Gas	6.50	8.33	8.14
Technology	10.48	19.82	20.78
Telecommunications	2.53	2.17	2.32
Utilities	5.69	3.55	3.33

Table 5: Ensemble strategy with $k = 10$ from December 1992 until October 2015: Breakdown of S&P 500 constituents by industry versus breakdown of ENS long and short portfolio holdings by industry, in percent.

5.4.2. Industry breakdown

For all constituents of the S&P 500 from 1990 until 2015, we obtain the Global Industry Classification Standard (GICS) code. In the leftmost column of table 5, we see the relative weight[6] of each industry in our entire data base. Contrasted to this figure is the investment behavior of the ensemble strategy. Apparently, financials are underweighted - they account for 17.1 percent of all shares in the data base, yet only 11.3 percent of the long positions and 9.9 percent of the short positions consist of stocks from the financial sector. Similarly, industrials and utilities are also underrepresented in the investment portfolios. This gap is filled by investments in technology companies, accounting for 10.5 percent of the data base versus 19.8 percent of all long and 20.8 percent of all short investments. Clearly, there seems to be a bias towards higher beta stocks (technology) at the expense of lower beta stocks (industrials, utilities, financials prior to the global financial crisis). Generally, within each industry group, the share of long and short investments is well-balanced.

5.4.3. Robustness checks

Given the high returns of our strategy and the silent reproach of data snooping whenever machine learning algorithms are deployed, we perform a series of robustness checks. Specifically, we run a sensitivity analysis on our input parameters. In table 6, we compare our baseline configuration with a low parametrization (alternative 1) and a high parametrization (alternative 2). The low parametrization exhibits 15 instead of 31 neurons in the first hidden layer for the DNN, and only half the number of trees for GBT and RAF. Conversely, the high parametrization features 62 hidden neurons in the first layer and twice the number of trees. All other parameters remain the same. Alternative 3 only applies to the DNN. Here, we benchmark the deep architecture with a dimensionality reduction over two additional hidden layers versus a 31-31-2 single-layer neural network with tanh as standard activation

[6]Note: Relative weights are calculated as number of stocks per industry divided by total number of stocks.

Model	Baseline configuration	Alternative 1	Alternative 2	Alternative 3
A. Design				
(D)NN: Architecture	31-31-10-5-2	31-15-10-5-2	31-62-10-5-2	31-31-2*
GBT: Number of trees	100	50	200	-
RAF: Number of trees	1000	500	2000	-
B. Mean return per day				
(D)NN	0.0033	0.0030	0.0031	0.0015
GBT	0.0037	0.0035	0.0038	-
RAF	0.0043	0.0042	0.0044	-

Table 6: Panel A: Design of baseline configuration versus lower parametrization (alternative 1) and higher parametrization (alternative 2). *DNN is also compared with a standard neural network with one hidden layer with 31 hidden neurons, no dropout regularization and tanh activation function (alternative 3). Panel B: Mean return per day for the $k = 10$ portfolio from December 1992 until October 2015 before transaction costs.

function and no dropout regularization. Panel A summarizes the model design.

Panel B of table 6 reports the returns per day for the $k = 10$ portfolio. We observe that the DNN produces consistently high returns around 0.30 percent per day - irrespective of the number of hidden neurons in the first layer (as long as the choice is made in a sensible way). However, when we eliminate the second and third hidden layer as well as the dropout regularization and choose tanh as standard activation function, returns diminish to 0.15 percent per day. The advantages of deep learning, i.e., the feature abstraction over several layers, dropout regularization, and maxout activation become evident. Aternative 3 is the simplest model considered in this article: The performance, including transaction costs, is positive from December 1992 to March 2001 and during the first months of the global financial crisis of 2008.

The tree-based methods are easier to configure and thus more robust. We observe a slight improvement when doubling the number of trees for GBT (0.38 versus 0.37 percent) and for RAF (0.44 versus 0.43 percent). Conversely, dividing the number of trees by two leads to slightly diminishing returns for GBT (0.35 versus 0.37 percent) and for RAF (0.42 versus 0.43 percent). This is an indication that we have not yet reached the point of overfitting.

6. Conclusion

We have developed a statistical arbitrage strategy based on deep neural networks, gradient-boosted trees, random forests, and their simple ensemble and deployed it on the S&P 500 constituents from December 1992 until October 2015. We make the following contributions to the available literature:

The first contribution focuses on the different machine learning approaches: We find that random forests outperform gradient-boosted trees and deep neural networks in our applica-

tion, making it the method of choice in light of our noisy feature space. However, careful hyperparameter optimization may still yield advantageous results for the tuning-intensive deep neural networks. The latter is subject for further research. Combining the predictions of our base learners - deep neural networks, boosted shallow trees, and decorrelated trees of high depth - into a simple ensemble outperforms each individual model. In case a practitioner is oblivious on which model to apply and if cheap computing power is available, an adequate compromise may be to construct an ensemble of diverse yet accurate base learners. Per se, it is possible to include more base learners into the ensemble and to apply advanced ensemble integration methods, such as stacking or super learning. The latter is subject for further research.

The second contribution is rooted in the performance evaluation of our trading strategies: For the ensemble strategy with $k = 10$, we find statistically and economically significant returns of 0.25 percent per day, or 73 percent on an annualized basis, after transaction costs. These returns translate to an annualized Sharpe ratio of 1.81, compared to 0.35 for the general market. However, there is still potential for reducing return standard deviations and tail risk, for example with a CVaR-portfolio optimization. The latter is subject for further research. We find that the ensemble returns partially load on systematic sources of risk, but daily alpha after transaction costs still ranges between 0.14 and 0.24 percent per day - depending on the employed factor model. Most importantly, returns can partially be explained by momentum and short-term reversal effects, indicating that the machine learning techniques partially extract both patterns from lagged return data. Ever since 2001, we see that returns start to deteriorate. We surmise that this effect is caused by the increasing popularization of the machine learning techniques that we deploy and the rise in computing power. Both drivers lead to excess returns being arbitraged away. However, when facing severe market turmoil, as during the financial or the European debt crisis, we still recognize potential for successful relative-value arbitrage.

The third contribution is based on the daily trading frequency. An analysis of variable importance suggests that most recent returns corresponding to the prior five trading days have the highest explanatory value. Then, returns corresponding to the past 12 trading months follow. Interestingly enough, most capital market anomalies focus on monthly data. Our analysis indicates that profitable patterns may yet be discovered based on more recent returns and daily data. The latter is subject for further research.

Overall, we hope having contributed to bridging the gap between academic (highly transparent models focusing on monthly data) and professional finance (highly performing black-box models focusing on profitability criteria). The consistently high returns prior to the year 2001 may potentially be interpreted as a successful machine learning powered statistical arbitrage strategy up to the early days of the new millennium. Clearly, the latter is speculative,

as no outsider has transparency about the exact nature of the models deployed at proprietary trading desks or within hedge funds. However, some publicly available information indicates that our returns are not unrealistic: For example, Renaissance Technologies, one of the most successful quantitative hedge funds, report average annualized returns of 71.8 percent from 1994 until mid-2014 before fees and a spike of 98.2 percent in 2008 (Rubin and Collins, 2015). Clearly, we do not claim that our algorithms are in any way affiliated to this firm, but they exhibit similarly strong performance and strong spikes at times of high market turmoil.

Allis, L. (1994). *Searching for solutions in games and artificial intelligence*. PhD thesis, University Limburg, Maastricht, The Netherlands.

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

Avellaneda, M. and Lee, J.-H. (2010). Statistical arbitrage in the US equities market. *Quantitative Finance*, 10(7):761–782.

Bacon, C. R. (2008). *Practical portfolio performance: Measurement and attribution*. Wiley finance. Wiley, Chichester, England and Hoboken, NJ, 2nd edition.

Bogomolov, T. (2013). Pairs trading based on statistical variability of the spread process. *Quantitative Finance*, 13(9):1411–1430.

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

Candel, A., LeDell, E., Parmar, V., and Arora, A. (2016). Deep learning with H2O.

Carhart, M. M. (1997). On persistence in mutual fund performance. *The Journal of Finance*, 52(1):57.

Click, C., Malohlava, M., Candel, A., Roark, H., and Parmar, V. (2016). Gradient boosted models with H2O.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer.

Dixon, M., Klabjan, D., and Bang, J. H. (2015). Implementing deep neural networks for financial market prediction on the Intel Xeon Phi. In *Proceedings of the 8th Workshop on High Performance Computational Finance*, pages 1–6.

Do, B. and Faff, R. (2010). Does simple pairs trading still work? *Financial Analysts Journal*, 66(4):83–95.

Enke, D. and Thawornwong, S. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications*, 29(4):927–940.

Fama, E. F. and French, K. R. (1996). Multifactor explanations of asset pricing anomalies. *The Journal of Finance*, 51(1):55.

Fama, E. F. and French, K. R. (2015). A five-factor asset pricing model. *Journal of Financial Economics*, 116(1):1–22.

Fernandes, M., Medeiros, M. C., and Scharth, M. (2014). Modeling and predicting the cboe market volatility index. *Journal of Banking & Finance*, 40:1 – 10.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Friedman, J. H. (1999). Stochastic gradient boosting. *Working paper, Stanford University*.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378.

Gatev, E., Goetzmann, W. N., and Rouwenhorst, K. G. (2006). Pairs trading: Performance of a relative-value arbitrage rule. *Review of Financial Studies*, 19(3):797–827.

Gedeon, T. D. (1997). Data mining of inputs: Analysing magnitude and functional measures. *International Journal of Neural Systems*, 8(02):209–218.

Genre, V., Kenny, G., Meyler, A., and Timmermann, A. (2013). Combining expert forecasts: Can anything beat the simple average?. *International Journal of Forecasting*, 29(1):108 – 121.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.

H2O (2016). H2O Documentation.

Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):993–1001.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer series in statistics. Springer, New York, 2nd edition.

Huck, N. (2009). Pairs selection and outranking: An application to the S&P 100 index. *European Journal of Operational Research*, 196(2):819–825.

Huck, N. (2010). Pairs trading and outranking: The multi-step-ahead forecasting case. *European Journal of Operational Research*, 207(3):1702–1716.

Huck, N. (2015). Pairs trading: Does volatility timing matter? *Applied Economics*, 47(57):6239–6256.

Ince, O. S. and Porter, B. R. (2006). Individual equity return data from Thomson Datastream: Handle with care! *Journal of Financial Research*, 29(4):463–479.

Jacobs, H. (2015). What explains the dynamics of 100 anomalies? *Journal of Banking & Finance*, 57:65–85.

Jacobs, H. and Weber, M. (2015). On the determinants of pairs trading profitability. *Journal of Financial Markets*, 23:75–97.

James, G. M., Witten, D., Hastie, T., and Tibshirani, R. (2014). *An introduction to statistical learning: With applications in R*. Springer texts in statistics. Springer, New York, corrected edition.

Khandani, A. E. and Lo, A. W. (2011). What happened to the quants in August 2007? Evidence from factors and transactions data. *Journal of Financial Markets*, 14(1):1–46.

Krauss, C. (2015). Statistical arbitrage pairs trading strategies: Review and outlook. *IWQW Discussion Paper Series, University of Erlangen-Nürnberg*.

Krauss, C. and Stübinger, J. (2015). Nonlinear dependence modeling with bivariate copulas: Statistical arbitrage pairs trading on the S&P 100. *IWQW Discussion Paper Series, University of Erlangen-Nürnberg*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Leippold, M. and Lohre, H. (2012). International price and earnings momentum. *The European Journal of Finance*, 18(6):535–573.

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

Lo, A. W. (2010). *Hedge Funds: An analytic perspective*. Advances in financial engineering. Princeton University Press, Princeton, rev. and expanded edition.

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

Mina, J. and Xiao, J. Y. (2001). Return to RiskMetrics: the evolution of a standard. *RiskMetrics Group*.

Moore, G. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117.

Moritz, B. and Zimmermann, T. (2014). Deep conditional portfolio sorts: The relation between past and future stock returns. *Working paper*.

Peterson, B. G. and Carl, P. (2014). PerformanceAnalytics: Econometric tools for performance and risk analysis. *R package*.

Pole, A. (2008). *Statistical arbitrage: Algorithmic trading insights and techniques*. John Wiley & Sons, Hoboken, N.J.

R Core Team (2014). R: A Language and Environment for Statistical Computing.

Rubin, R. and Collins, M. (2015). How an exclusive hedge fund turbocharged its retirement plan. *BloombergBusiness*.

Ryan, J. A. and Ulrich, J. M. (2014). R package xts: Extensible time series. *R package*.

Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, 5(2):197–227.

Sermpinis, G., Theofilatos, K., Karathanasopoulos, A., Georgopoulos, E. F., and Dunis, C. (2013). Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization. *European Journal of Operational Research*, 225(3):528–540.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

S&P Dow Jones Indices (2015). Equity S&P 500.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Takeuchi, L. and Lee, Y.-Y. A. (2013). Applying deep learning to enhance momentum trading strategies in stocks. *Working paper, Stanford University*.

The Economist (2016). A game-changing result.

Ulrich, J. (2013). R package TTR: Technical trading rules. *R package*.

Whaley, R. E. (2000). The investor fear gauge. *Journal of Portfolio Management*, 26(3):12 – 17.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

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