Stock Picking with Machine Learning

Dominik Wolff

Deka Investment GmbH, Mainzer Landstrasse 16, 60325 Frankfurt, Germany
and
Technical University Darmstadt, Hochschulstraße 1, 64289 Darmstadt, Germany

Fabian Echterling

Deka Investment GmbH, Mainzer Landstraße 16, 60325 Frankfurt am Main, Germany

First Draft: January 2020

This Version: April 2022

*Corresponding Author: Dominik Wolff, Technical University Darmstadt, Hochschulstraße 1, 64289 Darmstadt, Germany and Deka Investment GmbH, Mainzer Landstraße 16, 60325 Frankfurt am Main, Germany. Views expressed in this paper are those of the authors and do not necessarily reflect those of Deka Investment or its employees, Email: Dominik.Wolff@deka.de.

Stock Picking with Machine Learning

Abstract.

The cross sectional median return ...

We analyze machine learning algorithms for stock selection. Our study builds on weekly data for the historical constituents of the S&P 500 over the period from January 1999 to March 2021 and build on typical equity factors, additional firm fundamentals and technical indicators. A variety of machine learning models are trained on the binary classification task to predict whether a specific stock out- or underperforms the cross sectional median return over the subsequent week. We analyze weekly trading strategies that invest in stocks with the highest predicted outperformance probability. Our empirical results show substantial and significant outperformance of machine learning based stock selection models compared to an equally weighted benchmark. Interestingly, we find more simplistic regularized logistic regression models to perform similarly well compared to more complex machine learning models. The results are robust when applied to the STOXX Europe 600 as alternative asset universe.

Keywords: Investment Decisions, Equity Portfolio Management, Stock Selection, Stock Picking, Machine Learning, Neural Networks, Deep Learning, Long Short-Term Neural Networks (LSTM), Random Forest, Boosting

JEL classification: G11, G17, C58, C63

1. Introduction

Machine learning gained immense importance during the last decade mainly due to three reasons: The availability of computational power, improvements in machine learning algorithms, and the availability of large datasets, which are required to train complex models. While machine learning has increasingly attracted the attention of the asset management industry and the finance literature alike, the use of machine learning methodologies in portfolio management is still very limited.

Where has this claim come from ???

In this study, we combine insights from machine learning and finance research and analyze the potential of different machine learning algorithms for an important portfolio management task: predicting the relative returns of individual stocks. While earlier literature focuses on predicting equity market returns, we analyze the weekly predictability of the relative stock performance. More specifically, we group stocks based on their weekly relative performance and try to forecast outperforming vs. underperforming stocks in a binary classification task.[1]

Given the immense amount of data available and the complex and potential non-linear relations in the data, machine learning models might be very well suited for that task. In contrast to linear models, machine learning models can 'learn' non-linear relationships and even interactions between predictors without specifying the underlying model. Therefore, in the presence of nonlinearities and the availability of large training sets machine learning approaches might improve

[1] We also tested ternary predictions by defining three classes of equal size (outperformer, neutral, and underperformer). The empirical results were very similar to binary predictions.

stock-selection compared to linear models. Since, most machine learning models tend to do better on classification problems rather than on regression problems, we focus on the binary classification of stocks into out- and underperformer rather than on forecasting returns of individual stocks as point estimates.

We empirically analyze deep neural networks (DNN), long short-term neural networks (LSTM), random forest, gradient boosting and regularized logistic regression. All models are trained on stock characteristics to predict whether a specific stock out- or underperforms over the subsequent period. To obtain classes of equal size we use the cross-sectional median return as threshold to distinguish between out- and underperforming stocks.

Our asset universe builds on the historical constituents of the S&P 500 over the period from January 1999 to March 2021. We rely on the insights from the asset pricing literature and include the typical equity factors augmented by additional fundamental data and technical indicators. We analyze the risk-adjusted performance of a trading strategy that weekly picks stocks with the highest predicted probability to outperform. Our empirical results show a substantial and significant outperformance of machine learning based stock selection models compared to a simple equally weighted benchmark. Moreover, we find more simplistic regularized logistic regression models to perform similarly well compared to the more advanced ML models gradient boosting and random forest, DNN, and LSTM.

A number of related studies use machine learning models in an asset-pricing framework to identify relevant factors for explaining the cross-section of stock returns. Coqueret and Guida (2018) use regression trees to identify the most important firm characteristics for explaining stock returns. Chen et al. (2019) build a non-linear asset-pricing model for individual stock returns based on deep neural networks with macroeconomic and firm-specific information. The deep learning asset pricing model outperforms out-of-sample generating lower pricing errors. Similarly, Feng et al. (2018) develop an automated factor search algorithm that searches over firm characteristics with the objective of minimizing pricing errors. Based on group LASSO Freyberger et al. (2018) find that only 13 out of 62 characteristics provide incremental information for the cross-section of expected stock returns. In a similar fashion, Han et al. (2018) apply a forecast combination technique and show that relatively few characteristics affect cross-sectional value-weighted expected returns. Gu et al. (2019) develop an autoencoder asset pricing model that delivers far smaller out-of-sample pricing errors compared to leading factor models.

In addition to the above-mentioned studies that use machine learning for enhancing asset-pricing models, only a few studies use machine learning for stock selection and apply it in an out-of-sample trading strategy. Among those, Gu et al. (2020) employ machine learning for a large set of stock-level data to predict individual stock returns, which they subsequently aggregate to index predictions with promising results. Fischer and Krauss (2018) select stocks with machine learning based on daily historical returns. Chinco et al. (2019) use LASSO for rolling one-minute ahead return predictions based on the entire cross-section of lagged returns. Avramov et al. (2020) analyze stock selection with Neural Networks (NN) and Generative Adversarial Networks (GAN) with realistic investment restrictions. They conclude that realistic investment restrictions dramatically reduce the profitability of machine learning strategies. Choi et al. (2019) apply machine learning methods to predict stock returns in 34 markets around the world based on 12 variables, finding that complex machine learning models outperform linear models and earn higher risk-adjusted returns than the market. Cong et al. (2019) propose a reinforcement learning-based portfolio management model designed to improve over the traditional Markowitz (1952) two-step

What is that???

portfolio-construction approach. Wolff and Neugebauer (2019) analyze tree-based machine learning models and regularized regressions for predicting the equity risk premium.

Almost all of the above-mentioned studies use machine learning models for monthly predictions based on monthly data. In contrast, we analyze shorter-term predictability and focus on weekly data and weekly predictions. Analyzing weekly predictions provides two major advantages: First, the larger number of predictions and trades in an associated trading strategy provides higher statistical evidence due to the larger sample size. Second, machine learning modes require large training sets. Therefore, studies analyzing monthly predictions require very long training sets of at least 10 years. Given the dynamics of financial markets and the changing correlations in financial data over time, it could be suboptimal to train machine learning models on very old data which is not any more relevant for today's financial world due changing market conditions. Since our study builds on weekly data, we are able to reduce the length of the training set to only three years while still having enough observations for training complex machine learning models.

Moreover, in contrast to some of the before-mentioned studies, we focus on the practical relevance of our results and apply machine learning on the liquid S&P 500 constituents, thereby, excluding effects of non-tradable, illiquid stocks or microcaps. Moreover, we employ simple long-only, equally weighted trading strategies ensuring investability for most investors and preventing extreme portfolio positions.

Kk, is it equally weighted by money invested to each stock or by positions?

Our dataset includes well known equity factors that were documented to be important determinants of cross-sectional asset returns in earlier studies (e.g. Fama and French, 2018, Carhart, 1997, Frazzini and Pederson, 2014) but also technical indicators that were shown to contain meaningful information to predict the overall equity market (e.g. Neely et al. 2014).

The remainder of this study is organized as follows. First, we describe the data and the features in Section 2. Section 3 presents the prediction models ranging from linear predictive logistic regression models to non-linear machine learning approaches including tree-based models and neural networks. Section 4 presents the empirical results along with a variety of robustness checks. Section 5 concludes.

2. DATA

Our dataset builds on the historical constituents of the S&P 500 (in total 1,164 stocks) and includes weekly open prices and fundamental data for all stocks. Based on the availability of data, our study covers over 22 years of weekly data ranging from January 1999 to March 2021. Hence, our dataset includes more than 1.3 million stock-week observations. We include typical equity factors as well as additional fundamental data and technical indicators to predict if a specific stock outperforms the market in the subsequent period. We include three months lag for all fundamental data to avoid any forward-looking bias. All data is from Bloomberg.

What is forward-looking bias?

The fundamental data can be grouped in the typical equity factors size, value, quality, profitability, investment, and growth that were documented to be important determinants of cross-sectional asset returns in earlier studies (e.g. Fama and French, 2018, Carhart, 1997, Frazzini and Pederson, 2014). Panel A of Table 1 provides on overview of the fundamental data. Based on open prices we compute a range of technical indicators summarized in Panel B of Table 2.[2] The technical indicators include moving averages and momentum factors of different length, individual stock betas and volatilities, relative strength indices as indicators for short-term reversal, as well as a volume-based signal. We account for nine sector dummies based on Bloomberg Industrial Classification (BIC) codes.

We do not include interactions between variables, since machine learning models such as tree-based models and neural networks are able to model interactions in the data autonomously. We replace missing values in the feature set by carrying forward the last observation available. Any remaining missing values are filled using simple cross-sectional median imputation.

Our dataset builds on weekly observations using data from each Wednesday, in order to avoid start and end of the week effects. If a Wednesdays in the sample is a non-trading day, we use data of the next trading day available. We standardize all features as will be explained in section 3.1.

[2] We rely on open prices to ensure that the models can be implemented in practice. Trading can be executed a couple of minutes after markets open when the models ran. As robustness check, we also used close prices and obtained very similar results.

Timely

TABLE 1: FEATURES

Quarterly, yearly

Panel A: Fundamental Data	
Factor	**Variable**
Size	Market capitalization of equity
Value	Book-to market ratio
Quality	Earnings per share growth Earnings variability (deviation from earnings trend) Financial leverage
Profitability	Return on invested Capital (ROIC) Consensus earnings per share estimates for the subsequent year (EPS) Trailing 12M Net Income/ Market capitalization of equity Trailing 12M Sales/ Enterprise Value Trailing 12M Cash from Operating Activities/ Market capitalization of equity Trailing 12M Free Cash Flow to Equity/ Market capitalization of equity Trailing 12M Free Cash Flow/ Enterprise Value Trailing 12M Dividend Yield Trailing 12M Operating Margin Trailing 12M Profitability Margin
Growth	Asset Growth Trailing 12M Cash from Investing Activity/ Enterprise Value Employee Growth Trailing 12M Sales Growth
Sector	Sector Dummies (based on BIC codes)

Panel B: Technical Indicators	
Factor	**Variable**
Momentum	Momentum 12M Momentum 6M Momentum 1M Relative Share Price Momentum vs. Index (S&P 500)

Moving Averages	log(Price/Moving Average 200D)
	log(Price/Moving Average 100D)
	log(Price/Moving Average 50D)

Risk	Beta 12M Volatility 12M Volatility 6M Volatility 1M
Short Term Reversal	Relative Strength Index 14D Relative Strength Index 9D Relative Strength Index 3D log(Price/Bollinger Upper Band) log(Price/Bollinger Lowed Band) Lagged Return ($Return_{t-1}$, $Return_{t-2}$)
Trading Volume	USD Trading Volume (Stock Price x Trading Volume)

This table reports the features used for the binary prediction whether a specific stock out- or underperforms in the subsequent week. All features are standardized. Accounting data is lagged by three months to avoid any forward-looking bias.

3. Methodology

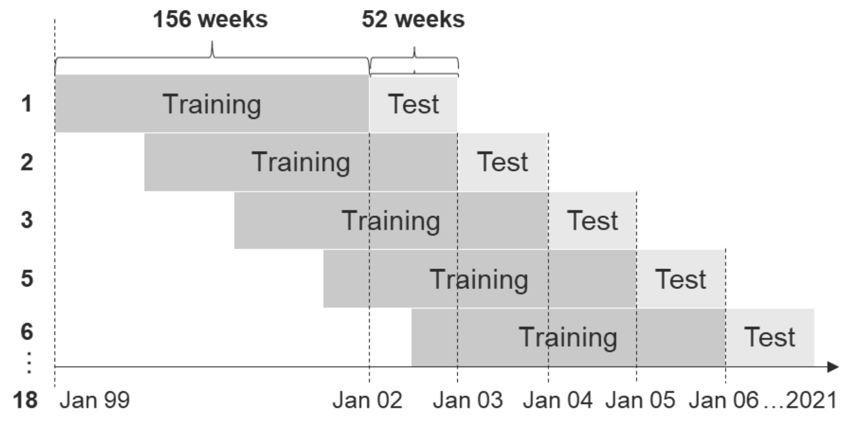
In this section, we provide a brief summary of the machine learning algorithms used in our analysis and the data splitting approach for generating training and test sets.

3.1 Organization of Training and Test Set

To prevent any overfitting, we strictly divide the dataset into training, validation, and test set. While the training set is used for fitting the models, the validation set is used to choose optimal 'tuning parameters' and the test set is used only for model evaluation and to run a trading strategy. A critical aspect is the choice of the training window length. On the one hand, the training window should not be too short in order to obtain enough observations for training complex models; on the other hand, training windows that are too long bear the risk of not sufficiently reflecting structural breaks and changing relationships in the financial market data. Since we combine cross-sectional (500 stocks) and weekly time series data, we can train machine learning models robustly on a relatively short data history of only three years of data (156 weeks). This results in a training set size of 77.500 observations.[3] Since our data set begins in January 1999, we train the initial models using data from 1999 to 2001, and then apply the initial models to the next 52 weeks (one year) of data for the period from January to December 2002.

[3] Three years with 52 weeks each multiplied with 500 stocks in the cross section. For prediction purposes, one observation is lost due to the lag between features and target.

FIGURE 1: ORGANIZATION OF TRAINING AND TEST SETS



This figure illustrates the organization of training and test sets. Each training set includes 3 years of data (156 weeks); 20% of the training set are used as validation set for hyperparamter search. Each test set includes one year of data (52 weeks). The out-of-sample evaluation period covers over 19 years from January 2002 to March 2021.

Iteratively, all models are re-trained on a yearly basis at the beginning of each calendar year based on the previous three years of data and applied on the subsequent year.[4] Similar to Fischer and Krauss (2018) we decided to yearly re-train all models to better account for possible structural breaks and changing relationships in financial market data. Therefore, our out-of-sample evaluation period covers over 19 years from January 2002 to March 2021. To prevent any survivorship bias we rely on the historical constituents of the S&P (in total 1,164 stocks) and build our analysis on stocks, which are included in the S&P 500 at the end of each training period. Figure 1 illustrates the organization of data in training and test sets.

All machine learning models are trained on the label whether a specific stock outperforms

[4] Due to limited computational power, we re-train all machine learning models only once a year.

(1) or underperforms (0) the median cross-sectional stock return in the subsequent week. All features are standardized based on the mean and the standard deviation of each feature in each training set. Next, we provide an overview on the machine learning models and the hyperparameter choice used in our analysis.

3.2 Cross validation and parameter choice

Machine learning algorithms rely on the choice of 'tuning parameters'. Tuning parameters include the penalization parameter in a ridge regression or the number of trees or the tree depth in random forest and boosting. To prevent overfitting we follow the usual approach of 'time series cross-validation'. From a grid of candidate tuning parameters, the tuning parameters that minimize the mean squared forecast error (MSFE) over the validation sets are used in the final model. For PCA, regularized logistic regression and tree-based machine learning models, we implement 5-fold cross-validation and select the 'tuning parameters' via grid search. More specifically, from a grid of candidate tuning parameters we select the 'tuning parameters' that minimize prediction error over 5 validation sets.[5] For neural networks, the tuning parameter include the number of layers, the number of neurons in each layer, the regularization technique (e.g. L1, L2, Dropout and batch normalization), the batch size, the learning rate and many more. This results in a large grid of candidate models with each model being computationally demanding making a full grid search

[5] The list of candidate tuning parameters is available in the appendix.

infeasible with limited computational power. Therefore, for neural networks we select tuning parameters based on the first training set only by individually tuning each parameter while holding all other parameters constant at their default values. While this approach is computationally sparse and a very effective way of tuning models in our setting it may result in a rather conservative performance evaluation of machine learning models since a tuning based on a full grid of hyperparameters and a re-tuning in each calendar year might yield in superior parameter combinations and in a higher model performance. We provide an overview of all model hyperparameters and the associated valued used in grid search in the appendix.

3.3 Regularized Logistic Regressions and Principal Component Regressions

The most simple machine learning model for classification is a linear logistic regression model. Typically, when the number of features is large and the dataset includes partly irrelevant and correlated features regularized regressions or principal component regressions (PCR) are superior compared to ordinary least squares (OLS). How is that the case??? Hoerl and Kennard (1970) introduced the ridge regression estimator as an alternative to the ordinary least squares (OLS) estimator in the presence of multi-collinearity. Ridge regression is a regularization technique, which performs linear regression with a penalty term on the coefficient estimates. Instead of just minimizing the residual sum of squares, an additional penalty on the coefficients is included. The minimization problem is given by:

$$\min_{\beta} \sum_{x,y} log(1 + \exp(-\beta' x \cdot y)) + \lambda \beta' \beta \quad (1)$$

where λ is a regularization parameter which controls for the level of shrinkage, x is the vector of lagged features and y is the prediction target, in our case the information whether a stock outperforms (1) or underperforms (0) in the subsequent week. In ridge regression, coefficient estimates are shrunk towards zero, but coefficients are usually never exactly zero. Hence, ridge regression cannot perform variable selection. In contrast, LASSO regression (Tibshirani, 1996) shrinks some regression coefficients to exactly zero and thereby performs variable selection. LASSO regression is very similar to ridge regression, with the difference that LASSO uses the absolute beta coefficients as penalty term, rather than the squared coefficients.

$$\min_{\beta} \sum\nolimits_{x,y} log(1 + \exp(-\beta' x \cdot y)) + \lambda|\beta| \quad (2)$$

If predictors are not on the same scale penalization in ridge and LASSO is unequal for different regressors. As for all models, we center and scale all features based on the full training set. The regularization parameter λ is determined directly from the data using 5-fold cross-validation, so that the MSFE is minimized in the validation set. For higher values of λ more shrinkage is employed and more coefficients are set to zero. While ridge and LASSO regression are very similar the coefficient estimates for both models can be very different. While LASSO provides the advantage of a higher interpretability, both models usually perform similarly well. As a shortcoming of LASSO, it usually selects one variable out of a group of correlated variables and ignores the others.

Why is this a shortcoming

Zou and Hastie (2005) combine the ideas of LASSO and ridge regression in the 'elastic net' combining the absolute and squared penalties. Zou and Hastie (2005) show that the elastic net often outperforms LASSO for real world data as well as in simulations, while being similarly sparse. In addition to ridge and LASSO regression we employ the elastic net approach and use 5-fold cross-validation for estimating the regularization parameters.[6]

What is principal component regression ...

In contrast to regularized regression, the idea of principal component regression is to reduce the feature set to a set of uncorrelated latent factors (principal components) that capture the common information (co-movements). Then the regression runs on these first principal components:

$$\min_{\beta} \sum_{x,y} log(1 + \exp(-\beta' F \ \cdot y)), \tag{3}$$

where F is a matrix containing the first k principal components. Principal component regression reduces model complexity and filters noise in the predictive variables, reducing the risk of overfitting (Ludvigson and Ng, 2007; Neely et al. 2014). We compute principal components based on standardized predictor variables. A critical issue is the selection of the optimal number of principal components to include in the predictive regression. We determine the optimal number of principal components to include in the PCR forecast directly from the data based in 5-fold cross validation. More specifically, we split the training set in 5 randomly selected subsets from which k-1 are used for estimation the PCR with the first N principal components and use the remaining subset for measuring the mean squared forecast error (MSFE) for each PCR model. [7] The number of principal components that minimizes MSFE over the k validation sets are used in the final model.

[6] For ridge, LASSO and the elastic net we perform a grid search to find the best parameters based on 5-fold cross-validation. The grid includes 500 values for λ on a logarithmic scale ranging from 0.0001 to 10,000, for the combination parameter α we use 20 values between 0 and 1.
[7] To accelerate the computation we only include up to N principal components, with N being the elbow point.

3.4 Random Forest and Boosting

Not familiar with this topic. Go over it later

Tree-based models such as random forests and boosting are based on the idea of splitting the target variable into binary groups with the highest possible intra-group homogeneity. Breiman et al. (1984) propose the classification and regression trees (CART) algorithm choosing the splits in a top-down fashion: At the beginning of the tree, the feature that allows the highest gain in homogeneity of the target variable is chosen for splitting. The tree grows by sequentially adding variables for splitting and obtaining increasingly homogenous groups. The tree grows until a pre-defined tree depth or until group-homogeneity does not improve over a pre-defined threshold. On a stand-alone basis, decision trees generally provide a low predictive power, since they tend to overfit the data. Breiman (1986) proposes a bootstrap aggregation approach ('bagging') which averages predictions from a collection of trees where each tree is generated from random subsamples of the training data. Bagging alleviates the overfitting problem and usually substantially enhances out-of-sample predictions. As a drawback, the interpretability of decision trees is lost, but the importance of features for the final prediction can be extracted based on the frequency of features appearing in each tree in the forest. Bagging itself is extended by the random forest approach (Friedman et al., 2001) where only a random subset of features is included in each tree. Random Forest usually improves over bagging since it enhances the variability of trees and the dominance of single predictors is alleviated. Typically, for classification problems the number of randomly allowed predictors is the square root of the number of all features in the dataset. We leave the number of predictors at this default value., We choose the 'tuning parameters' 'number of trees', 'maximum tree depth' (number of variables included in each tree) and 'node size' (minimum number of observations in each node) based on a grid of candidate parameters provided in the appendix with 5-fold cross validation.

While random forest rely on the majority vote of multiple trees trained on random sets of features and random subsamples of the training set, Boosting algorithms, in contrast, try to minimize the prediction error by iteratively adjusting the tree in such a way that each tree aims at correcting the errors of the previous trees. This is achieved by re-weighting observations adaptively, putting higher weight on observations that were previously misclassified. Boosting was implemented by Freund and Schapire (2003) in the 'AdaBoost' algorithm which we include in our analysis. We employ the 'AdaBoost' algorithm with 1000 iterations and choose the 'maximum tree depth', 'node size' and the additional Boosting parameters based on a grid of candidate parameters provided in the appendix with 5-fold cross validation.

3.5 Feedforward Neural Networks

Feedforward Neural Networks consist of neurons, which are organized in layers. A shallow neural network consists of one input layer, which takes the feature values as an input, typically one or two hidden layers, and one output layer.[8] The input features are fed into the input layer whereas each hidden layer takes the output from the previous layer as input. Each neuron in the neural network works similar to a logistic regression: Each neuron processes a vector of input data X, computes the activation *a* as the sum of weighted inputs with *w* being the weighting vector, adds a constant *b* (bias term) and feeds the activation *a* into a non-linear activation function. The weights *w* and the bias *b* are initialized randomly leading to stochastic and unreasonable outputs

[8] In contrast to shallow neural networks, deep neural networks have usually three or more hidden layers. Cybenko (1989) shows that a single hidden layer network with a finite number of neurons is capable of approximating any continuous function (universal approximation theorem). However, deep neural networks usually approximate the same function with less neurons compared to neural networks with only one hidden layer, thereby working more efficient.

in the first iteration. The last layer - the output layer - produces the output of the neural network where the prediction error made by the network is measured with the loss function *L*. Weights *w* and biases *b* are then updated in the backpropagation step based on stochastic gradient descent (SGD). Basically, the gradients of the loss function with respect to weights *w* and biases *b* are used to shift weights and biases towards more optimal values, thereby reducing prediction error (loss) made by the network. The network is trained over many iterations, presenting random subsamples (batches) of the data in each iteration. The number of iterations required until the complete training set is passed through the network is called 'epoch'. The training of the network stops after a predefined number of epochs or when the loss does not decrease over a couple of epochs (early stopping).

Overfitting is a common problem in neural networks. That means that neural networks adopt too well to the training set and do not perform well out of sample. A couple of strategies were proposed to reduce overfitting of neural networks. The most prominent among these are dropout, batch normalization and (L1, L2) regularization. Dropout was proposed by Srivastava et al. (2014) and means that a predefined number of neurons are randomly dropped in each iteration. L1 and L2 regularization in neural networks is similar to regularized LASSO and ridge regression (Tibshirani, R., 1996) described above. With regularization, the loss function does not only include the prediction error of the network but also a penalty term on the weights (L1 regularization: absolute weights, L2 regularization: squared weights) in the network, leading to sparse weights.

Based on the first training set (first three years of data) in which we test different model architectures, we implement a feed forward neural network with three hidden layers having 20, 10, and 5 neurons per layer. We train the network with 100 epochs and early stopping of 10, meaning that the training stops if the network does not improve over 10 episodes.

In line with Avramov et al. (2020) we employ batch regularization and L1-regularization in each hidden layer with regularization parameter of 0.0001. We rely on the RMSprop optimizer with a learning rate of 0.001.[9] The RMSprop optimizer was proposed by Tieleman and Hinton (2012) and is one of the most popular optimization algorithms in deep learning.

3.6 Recurrent Neural Networks and LSTM

Feedforward neural networks discussed above consider all input features as independent observations and, therefore, temporal effects - usually present in time series data - can only be addressed by feeding in past observations as separate features. Unlike feedforward networks, recurrent neural networks (RNN) have cyclic connections making them powerful for modeling time-dependencies in the data, by processing both current observations and the output of the previous time step. However, while RNNs are able to model short-term time dependencies they are not very well in accounting for longer-term dependencies due to the vanishing (or exploding) gradient problem (Bengio et al., 1994; Sak et al, 2014). An established approach for considering short as well as long-term dependencies are long short-term memory (LSTM) neural networks, a special form of recurrent neural networks developed by Hochreiter and Schmidhuber (1997). LSTM networks

[9] For sparsity, due to limited computational power, we choose the architecture of the DNN, as well as the learning rate and regularization parameter based on cross-validation for the first training set only (i.e. the first three years of data). This provides a conservative evaluation of the performance of DNNs, since for recent data another architecture might be superior.

are applied in many fields ranging from speech recognition, machine translation, text and handwriting detection to time series prediction for financial data (see also Jozefowicz et al., 2015).

LSTM networks do not only rely on current inputs, but also on short-term and long-term memory derived from previous inputs. A LSTM network consists of an input layer, one or several hidden layers and an output layer. The input and output layers consist of ‘regular’ neurons. In the input layer, the number of neurons equals the number of input features. The hidden layer consists of LSTM cells. Each LSTM cell controls memory with three ‘control switches’: an input gate, that decides which information is added to the cell memory, a forget gate, that decides which information is removed from the memory and an output gate, that defines which information from the cell memory is used as output. The control switches use sigmoid activation functions to efficiently learn how to weigh current observations against long-term versus short-term memory and hyperbolic tangent activation functions are used for processing the data. This architecture makes the LSTM cell robust when dealing with long-term dependencies, but also for capturing non-stationarity (Chen et al., 2019). One may argue that the architecture of a LSTM cell is rather heuristic and many alternative architectures with alternative activation functions and gates are possible. However, Jozefowicz et al. (2015) empirically analyze the performance of LSTM networks compared to 10,000 different RNN architectures, finding that LSTM networks work well in many applications outperforming competing models.

Based on the first training set (first three years of data) in which we test different LSTM architectures, we use a LSTM network with one hidden layer containing 30 cells and a softmax activation function in the output layer for the binary output. In line with the DNN, we train the network with 100 episodes and early stopping of 10. As for DNN we use the RMSprop optimizer

(Tieleman and Hinton, 2012) with a learning rate of 0.001.

3.7 Combined Forecast

Combining forecasts across different prediction models may result in forecasts superior to all individual forecasts (Bates and Granger, 1969). If individual forecasts are not perfectly correlated, the combined forecasts are less volatile and usually provide lower fluctuation in prediction accuracy over time (Hendry and Clements, 2004; Timmerman, 2006). The simplest form of combining individual forecasts is simply averaging over several models. Several studies showed that a more sophisticated weighting scheme of individual forecasts, such as minimizing the historical mean squared forecast error (MSFE) usually does not perform better than the simple average (Clemen, 1989; Stock and Watson, 2004).[10] We compute an ensemble model that combines all models by simply averaging over all forecasts.

4. Empirical Results

In this section, we present our empirical results. In section 4.1 we analyze the classification accuracies and the distribution and correlation of predictions across different models. In section 4.2., we analyze investment strategies based on ML models. Then, we analyze the optimal portfolio size for the ML models (4.3), the ML portfolio performance during different economic cycles (4.4) and the exposures of the ML portfolios to common risk factors (4.5). In section 4.6, we analyze feature importance metrics to shed light on how machine learning models arrive at their decisions.

[10] This stylized fact is termed the 'forecast combining puzzle' because in theory it should be possible to improve upon simple combination forecasts.

In section 4.7, we re-run all models on the Stoxx 600 Index constituents as robustness check.

4.1 Analysis of Predictions

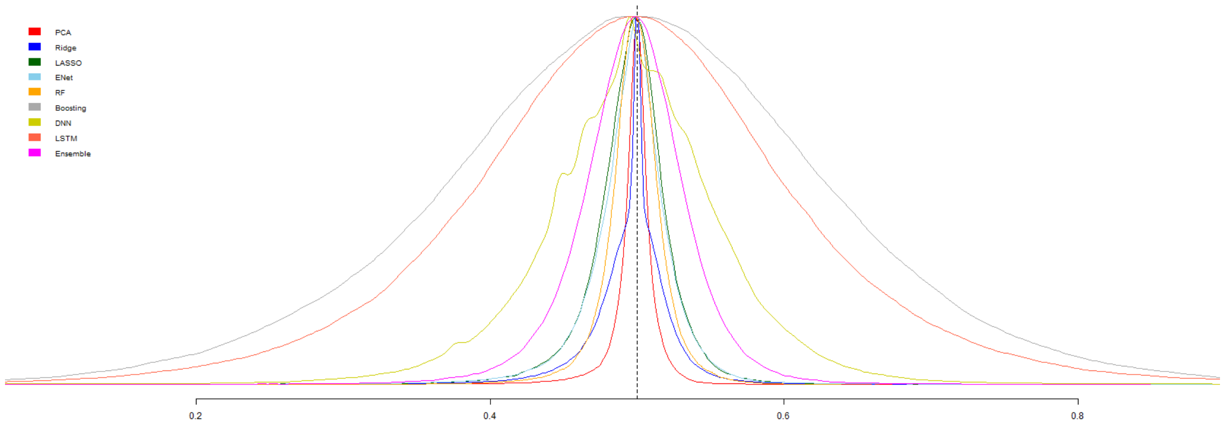
We start with the analysis of prediction accuracy and the associated sensitivity and specificity measures for the different models and present the results in table 2. The overall classification accuracy measures are not very impressive ranging from 50.4% for the PCA model to 50.8% for the ensemble model. However, the classification accuracies for all models are highly statistically significant larger than the no information rate (50%). The associated Mcnemar p-values are close to zero, indicating high statistical significance.

Table 2: Model Accuracies

	Accuracy (Full Sample)	Sensitivity	Specificity	Mcnemar P-Value (Sig.Level)	Acc. 10% Higest-Lowest	Acc. 5% Higest-Lowest	Acc. 1% Higest-Lowest
PCA	50.43%***	52.21%	48.65%	***	51.11%***	51.45%***	51.58%***
Ridge	50.63%***	54.41%	46.86%	***	51.93%***	51.80%***	52.06%***
LASSO	50.53%***	55.45%	45.62%	***	51.89%***	51.72%***	51.92%***
ENet	50.55%***	55.36%	45.75%	***	51.87%***	51.72%***	51.03%***
RF	50.69%***	53.04%	48.35%	***	52.00%***	52.15%***	51.79%***
Boosting	50.61%***	50.74%	50.48%	***	52.36%***	52.68%***	53.77%***
DNN	50.59%***	52.63%	48.56%	***	51.37%***	51.51%***	51.18%***
LSTM	50.64%***	51.98%	49.30%	***	51.83%***	52.24%***	53.48%***
Ensemble	50.81%***	53.29%	48.33%	***	52.56%***	52.84%***	53.37%***

This table reports accuracy measures for the binary predictions whether a specific stock out- or underperforms in the subsequent week and the associated sensitivity and specificity measures for the different models. '***', '**', '*' indicate significance at the 0.1%, 1% and 5% significance level, respectively for the null hypothesis that the accuracy is below or equal to 50%.

FIGURE 2: DISTRIBUTION OF POOLED PREDICTIONS



The figure shows the distribution of predictions for the different models pooling time-series and cross-sectional predictions. The figure highlights the different standard deviations of different models.

The sensitivity measures presented in table 2 are larger than 50% for all models, stating that all models identify outperforming stocks with higher accuracy than the no information rate (guessing). In contrast, the specificity measures are below 50% for all models except for gradient boosting, indicating that ML models do worse in correctly identifying underperforming stocks, than in correctly identifying outperforming stocks. As mentioned above, we define outperforming (underperforming) stocks as stocks performing better (worse) than the median return during a specific week, ensuring equal class size.

Figure 2 plots the distribution of predictions for each model pooling time-series and cross-sectional predictions. The figure illustrates that predictions are approximately symmetrically distributed with center at around 0.5. Interestingly, the standard deviation of predictions differs substantially for different models. The outperformance probability predictions capture additional information than is not reflected in the simple accuracy measure. The accuracy measure classifies all stocks with an outperformance probability predictions larger than 50% as outperformers. However, a stock with a higher outperformance probability predictions e.g. of 80% is more likely to outperform compared to a stock with an outperformance probability prediction of only 51%.

Therefore, we compute accuracies for the stocks with the 10% (5%, 1%) largest and lowest outperformance probability predictions and present the results in the last three columns of table 2. The results illustrate that higher outperformance probability predictions correspond to higher prediction accuracies. More specifically, the accuracy monotonously increases if the stock picking is applied more aggressively (in most of the cases). This motivates a trading strategy of investing only in N stocks with the largest predicted outperformance probabilities.

Next, we analyze the correlation of predictions across different models pooling predictions over time and for different stocks. The results presented in table 3 show that predictions of related models are highly correlated, while predictions of different model families are relatively uncorrelated. For instance, predictions of the regularized regression models ridge, LASSO and the elastic net are highly correlated with correlation coefficients of 0.95 or larger. Predictions of the tree-based models random forest and gradient boosting generate moderately correlated predictions with a correlation coefficient of 0.49. The PCA predictions are moderately correlated to the other regression based models ridge, LASSO and the elastic net. The neural networks DNN and LSTM are relatively uncorrelated to the other models with correlations being below 0.45. Also, the predictions of tree-based models are relatively uncorrelated to predictions of regularized regressions and to predictions of neural networks. Overall, predictions of different models are relatively uncorrelated. This motivates an ensemble strategy that averages predictions across different models for diversifying over different models.

TABLE 3: CORRELATIONS OF POOLED PREDICTIONS ACROSS DIFFERENT MODELS

	PCA	Ridge	LASSO	ENet	RF	Boosting	DNN	LSTM
PCA	1***							
Ridge	0.52***	1***						
LASSO	0.53***	0.95***	1***					
ENet	0.52***	0.95***	0.98***	1***				
RF	0.45***	0.55***	0.56***	0.55***	1***			
Boosting	0.12***	0.25***	0.26***	0.25***	0.49***	1***		
DNN	0.22***	0.4***	0.42***	0.41***	0.43***	0.3***	1***	
LSTM	0.10***	0.25***	0.26***	0.26***	0.37***	0.37***	0.45***	1***

The table displays the correlations of predictions of different models pooling time-series and cross-sectional predictions. The figure highlights that predictions of related models are highly correlated, while predictions of different model families are relatively uncorrelated. '***', '**', '*' indicate significance at the 0.1%, 1% and 5% significance level, respectively for the null hypothesis that a specific correlation coefficient equals zero.

4.2 PERFORMANCE OF MACHINE LEARNING BASED TRADING STRATEGIES

Next, we analyze a trading strategy that picks stocks with the highest outperformance probability prediction. Each week the strategy invests in the stocks (equally weighted) with the highest predictions. Table 4 reports the strategy performance for different machine learning models for portfolios sizes of 50 (Panel A), 100 (Panel B) and 200 stocks (Panel C). As benchmarks we include the value weighted S&P 500 index and compute an equally weighted portfolio including all constituents of the S&P 500. Figure 3 plots the performance of machine learning strategies (portfolio size 50) and the equally weighted benchmark (dashed line) over time.

Table 4 shows that all machine learning models generate substantially higher returns than the S&P 500 index and the equally weighted benchmark. Interestingly, tree-based machine learning models (RF and Boosting) and neural networks (DNN and LSTM) achieve similar returns and sharpe ratios as regularized logistic regression models (Ridge, LASSO and ENet) for portfolio size 100 and 200. The PCA model shows the lowest and the ensemble model the highest returns for all portfolio sizes (50, 100 and 200 stocks). The table shows that returns decline with increasing portfolio size, which is in line with table 2 stating that for smaller portfolio sizes only stocks with the highest outperformance probabilities are selected and accuracies being higher for more confident predictions.

Table 4 also shows that volatility declines with increasing portfolio size. This is because the level of diversification increases and idiosyncratic risk declines with portfolio size. Consequently, the sharpe ratio optimal portfolio size is a trade-off of outperformance probability and diversification. We analyze the optimal portfolio size in the next section. Comparing the risk-adjusted performance of different machine learning models, we find that the ensemble model achieves the highest sharpe ratios of 0.84 (portfolio size 50), 0.79 (portfolio size 100), and 0.73 (portfolio size 200), substantially outperforming the equally weighted benchmark portfolio which yields a sharpe ratio of 0.57 and the S&P 500 index (0.40).

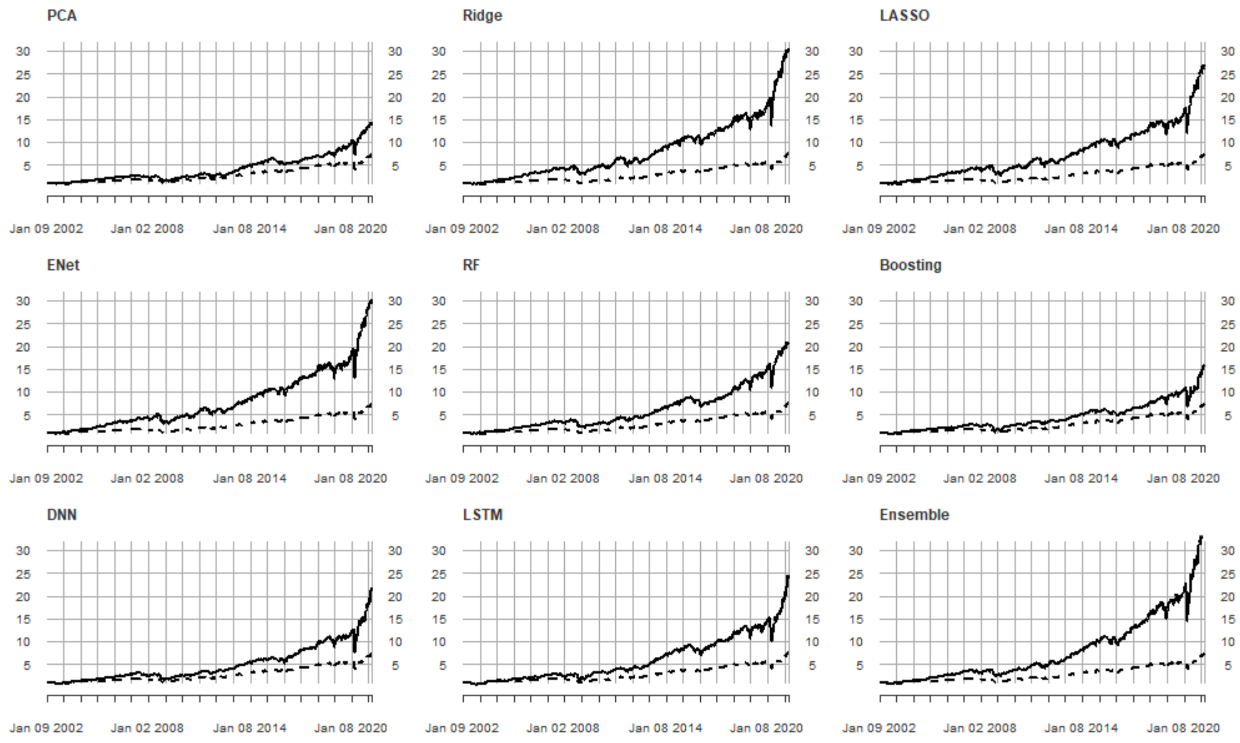
All machine learning based stock selection models substantially enhance portfolio returns. While the equally weighted benchmark earned 11.1% p.a (the value weighted S&P 500 index yielded 6.4% p.a.) the ensemble model earned 20.8% p.a. and the ridge regression 19.4% p.a. (portfolio size 50). Compared to the equally weighted benchmark the machine learning strategies are more risky with volatilities ranging from 21.1% (PCA) to 25.9% (DNN) compared to 19.3% benchmark volatility.

TABLE 4: PERFORMANCE OF DIFFERENT MACHINE LEARNING MODELS FOR STOCK PICKING

Panel A: Portfolio Size 50	**Return p.a.**	**Vola p.a.**	**Sharpe p.a.**	**Max. Drawdown**	**CAPM Alpha p.a.**	**FF-6 Alpha p.a.**	**Turn-over p.a.**	**BTC vs. 1/N**
Market (S&P 500)	6.60%	16.70%	0.40	-55.40%	0.00	0.00		
Benchmark (1/N)	11.1%	19.3%	0.57	-54.8%	0.04*	0.05*	0.8	0.00%
PCA	14.8%	21.1%	0.70	-48.2%	0.08**	0.08*	17.4	0.11%
Ridge	19.4%	25.1%	0.77	-45.2%	0.11***	0.14***	16.4	0.25%
LASSO	18.6%	25.5%	0.73	-46.5%	0.10**	0.13**	19.4	0.19%
ENet	19.3%	25.5%	0.76	-46.3%	0.11***	0.14***	19.9	0.21%
RF	16.9%	22.7%	0.75	-50.6%	0.09***	0.11**	18.7	0.16%
Boosting	15.5%	24.4%	0.63	-55.1%	0.08**	0.10**	32.8	0.07%
DNN	17.2%	25.9%	0.67	-52.4%	0.09**	0.12**	28.0	0.11%
LSTM	18.1%	25.6%	0.71	-50.1%	0.10**	0.13***	31.1	0.11%
Ensemble	20.8%	24.8%	0.84	-44.8%	0.12***	0.15***	28.8	0.17%
Panel B: Portfolio Size 100	**Return p.a.**	**Vola p.a.**	**Sharpe p.a.**	**Max. Drawdown**	**CAPM Alpha p.a.**	**FF-6 Alpha p.a.**	**Turn-over p.a.**	**BTC vs. 1/N**
Market (S&P 500)	6.60%	16.70%	0.40	-55.40%	0.00	0.00		
Benchmark (1/N)	11.1%	19.3%	0.57	-54.8%	0.04*	0.05*	0.8	0.00%
PCA	12.6%	19.6%	0.64	-52.3%	0.06**	0.06*	14.4	0.05%
Ridge	15.9%	22.1%	0.72	-43.3%	0.08***	0.10**	14.1	0.17%
LASSO	15.6%	22.1%	0.70	-42.5%	0.08***	0.10**	17.1	0.13%
ENet	15.1%	22.2%	0.68	-42.5%	0.07**	0.09**	17.4	0.12%
RF	15.6%	20.9%	0.75	-44.7%	0.08***	0.09**	14.6	0.15%
Boosting	15.9%	22.4%	0.71	-47.7%	0.08***	0.10**	28.4	0.08%
DNN	16.0%	22.4%	0.71	-45.3%	0.08***	0.10**	24.0	0.10%
LSTM	16.3%	22.8%	0.72	-45.6%	0.08***	0.11***	26.7	0.10%
Ensemble	17.7%	22.5%	0.79	-43.3%	0.10***	0.12***	24.4	0.14%
Panel C: Portfolio Size 200	**Return p.a.**	**Vola p.a.**	**Sharpe p.a.**	**Max. Drawdown**	**CAPM Alpha p.a.**	**FF-6 Alpha p.a.**	**Turn-over p.a.**	**BTC vs. 1/N**
Market (S&P 500)	6.60%	16.70%	0.40	-55.40%	0.00	0.00		
Benchmark (1/N)	11.1%	19.3%	0.57	-54.8%	0.04*	0.05*	0.8	0.00%
PCA	12.1%	18.5%	0.65	-51.9%	0.05**	0.05*	9.9	0.05%
Ridge	13.3%	20.2%	0.66	-47.5%	0.06**	0.07*	10.3	0.11%
LASSO	13.1%	20.1%	0.65	-47.2%	0.06**	0.07*	12.4	0.08%
ENet	13.0%	20.1%	0.65	-46.4%	0.06**	0.07*	12.7	0.07%
RF	14.1%	19.6%	0.72	-46.1%	0.07***	0.08**	10.1	0.15%
Boosting	15.0%	20.5%	0.73	-47.0%	0.07***	0.09**	21.1	0.09%
DNN	13.7%	20.2%	0.68	-47.3%	0.06***	0.08**	17.8	0.07%
LSTM	14.9%	20.5%	0.73	-46.0%	0.07***	0.09**	19.6	0.10%
Ensemble	15.0%	20.5%	0.73	-44.7%	0.07***	0.09**	17.4	0.11%

The table shows the performance of a trading strategy investing in the 50, (100, 200) stocks (equally weighted) with the largest predictions based on the machine learning model specified in the first column for the full sample from 2002 to 2021. '***', '**', '*' indicate significance at the 0.1%, 1% and 5% significance level, respectively for the null hypothesis that the alpha of a trading strategy equals zero.

FIGURE 3: PERFORMANCE OF MACHINE LEARNING-BASED TRADING STRATEGY VS. 1/N BENCHMARK (DASHED)



The figure displays the performance of the trading strategies investing in the 50 stocks (equally weighted) with the largest predictions (based on the different machine learning models) for the full sample from 2002 to 2021 compared to a benchmark portfolio that equally invests in all stocks (1/N benchmark).

However, the maximum drawdown is smaller for all machine learning-based stock picking strategies compared to both benchmarks. Most importantly, table 4 shows that all machine learning strategies generate positive, statistically significant and economically relevant risk-adjusted returns with Jensen alphas ranging from 8% p.a. (PCA) to 12% p.a. (ensemble model). In addition, when controlling for the six Fama-French (2018) factors positive and statistically significant alphas remain ranging from 8% p.a. (PCA) to 12% p.a. (ensemble model). This result shows that the performance of the machine learning models cannot be explained by exposures to known risk factors.

Table 4 also presents the portfolio turnover for each strategy. The turnover ranges from 9.9

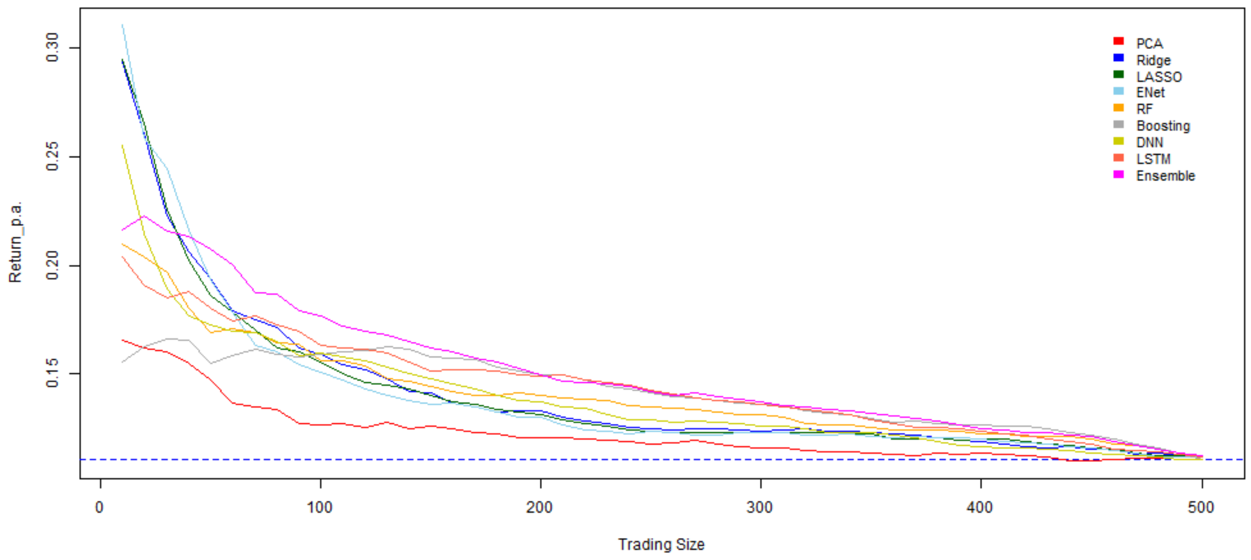
(PCA) to 31.1 (LSTM) stating that the portfolio value is traded 9.9 to 31.1 times on average each year (one-sided turnover), depending on the model and the portfolio size. In line with Avramov et al. (2020), we find that machine learning strategies rely on a large turnover. In the last column of table 4 we present the break-even transaction costs (BTC). The BTC states the level of variable transaction costs until which the strategy is beneficial over the benchmark 1/N strategy. The BTC range from 5 (PCA) to 21 (ENet) basis points. For the ensemble model the BTC are 17 basis points highlighting that transaction costs have to be low in order to exploit the machine learning models.

To summarize our results so far, we find that machine learning models are able to identify outperforming stocks but rely on a high level of turnover. These findings are in line with Avramov et al. (2020). Therefore, capitalizing on machine learning models requires efficient trading with low transaction costs, preferably below 5 basis points of the trading volume.

4.3 ANALYSIS OF THE OPTIMAL PORTFOLIO SIZE

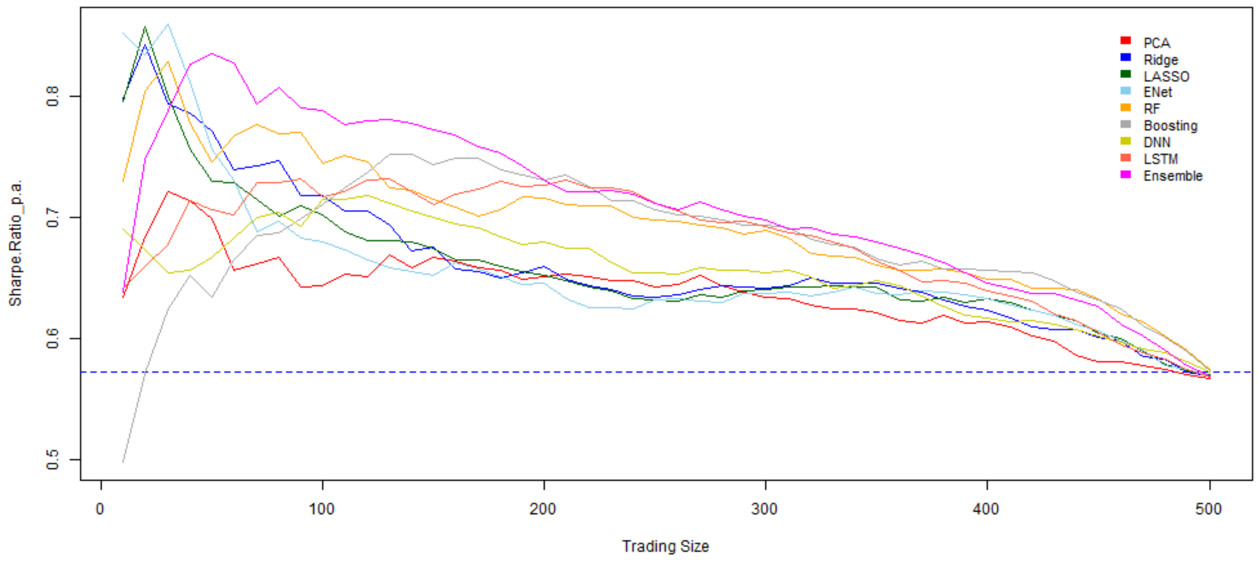
In the previous section, we analyzed arbitrarily chosen portfolio sizes of 50, 100 or 200 stocks, respectively. In this section, we explore the optimal portfolio size for implementing the machine learning strategies. As seen in the previous section, if portfolio size declines only stocks with the highest outperformance potential are selected yielding in higher returns. Figure 4 plots the average annualized returns of the machine learning strategies depending on the portfolio size. The figure confirms that portfolio returns decline with portfolio size. This corroborates the finding reported in table 2 that accuracy improves with higher predictions. While smaller portfolio sizes provide higher expected returns, they are subject to higher idiosyncratic risk due to lower level of diversification. Hence, a sharpe ratio optimal portfolio size is a trade-off of outperformance probability and diversification. Figure 5 plots the sharpe ratio for varying portfolio sizes.

FIGURE 4: AVERAGE ANN. RETURNS OF MACHINE LEARNING STRATEGIES FOR DIFFERENT PORTFOLIO SIZES



The figure illustrates the annualized return of the trading strategies for different portfolio sizes N. The trading strategies weekly invest in the N stocks (equally weighted) with the largest predictions.

FIGURE 5: SHARPE RATIOS OF MACHINE LEARNING STRATEGIES FOR DIFFERENT PORTFOLIO SIZES



The figure illustrates the sharpe ratios of the trading strategies for different portfolio sizes N. The trading strategies weekly invest in the N stocks (equally weighted) with the largest predictions.

The figure shows that the sharpe ratio optimal portfolios size depends on the machine learning model, but is around 50 for most models. Therefore, we continue our analysis with a portfolio size of 50.

4.4 Performance in different economic cycles

Next, we analyze the performance of machine learning strategies over time. To analyze whether the outperformance of machine learning models is stable over time and whether it is present during sub-periods, we split the full evaluation period into four sub-periods based on the NBER recession indicator. Table 5 presents the sub-period showing that all machine learning strategies outperform the S&P 500 index (market) during all sub-periods. Moreover, all machine learning models achieve a higher or at least the same return compared to the equally weighted benchmark during all sub-periods.

Table 5: Sub Period Returns: NBER Recession Dummies

	2002-01/2007-11 Expansion	**2007-12/2009-06 Recession**	**2009-07/2020-02 Expansion**	**2020-03/2021-03 Recession**
S&P 500	0.9%	-6.6%	2.7%	5.0%
Benchm.(1/N)	2.3%	-4.9%	3.3%	6.4%
PCA	**3.4%**	**-4.3%**	**3.8%**	**7.9%**
Ridge	**5.8%**	**-1.9%**	**3.5%**	**11.4%**
LASSO	**5.7%**	**-1.7%**	3.3%	**11.0%**
ENet	**5.8%**	**-1.7%**	**3.4%**	**11.4%**
RF	**4.7%**	**-3.3%**	**3.9%**	6.4%
Boosting	**3.6%**	**-1.2%**	3.3%	**10.0%**
DNN	**3.5%**	**-2.3%**	**3.8%**	**13.5%**
LSTM	**3.5%**	**0.4%**	**3.8%**	**12.7%**
Ensemble	**4.5%**	**0.1%**	**4.3%**	**13.0%**

The table shows the performance of a trading strategy investing in the 50 stocks (equally weighted) with the largest predictions based on the machine learning model specified in the first column for different sub periods. Bold values indicate a higher performance than the 1/N benchmark portfolio.

4.5 Factor attribution

An essential question is whether the performance of the machine learning strategies is attributable to common factors and whether factor exposures of the models are stable over time. Therefore, we run full sample and rolling factor regressions of machine learning strategy returns on a common factor set. We rely on the Fama and French (2018) six factor model (FF-6) augmented by the betting-against-beta (BaB) factor proposed by Frazzini and Pederson (2013).[11] Hence, our factor regressions control for the market risk factor ("MKT"), the size factor ("SMB"), the value factor ("HML"), the quality factor ("RMW"), the investment factor ("CMA"), the momentum factor ("MOM") as well as the betting-against-beta factor ("BaB").

Table 6 presents the results of the full sample factor regressions. The table shows that all machine learning strategies have a significant exposure to the market factor, which is natural given that they are long-only strategies. Moreover, all strategies except for PCA have significant positive exposures to the value factor ("HML") and the betting-against-beta factor ("BaB"), and a significant negative exposure the quality factor ("RMW"). Interestingly, the momentum factor ("MOM") is only significant for boosting and the neural nets, highlighting the differences in the different models. The size factor is insignificant for almost all models, stating that size does not play a major role in stock selection of the machine learning models.

[11] Fama-French factor data are from Kenneth French Website: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html, Betting-against-beta (BaB) factor data are from the AQR Website: https://www.aqr.com/Insights/Datasets/Betting-Against-Beta-Equity-Factors-Daily

TABLE 6: FULL SAMPLE FACTOR REGRESSIONS

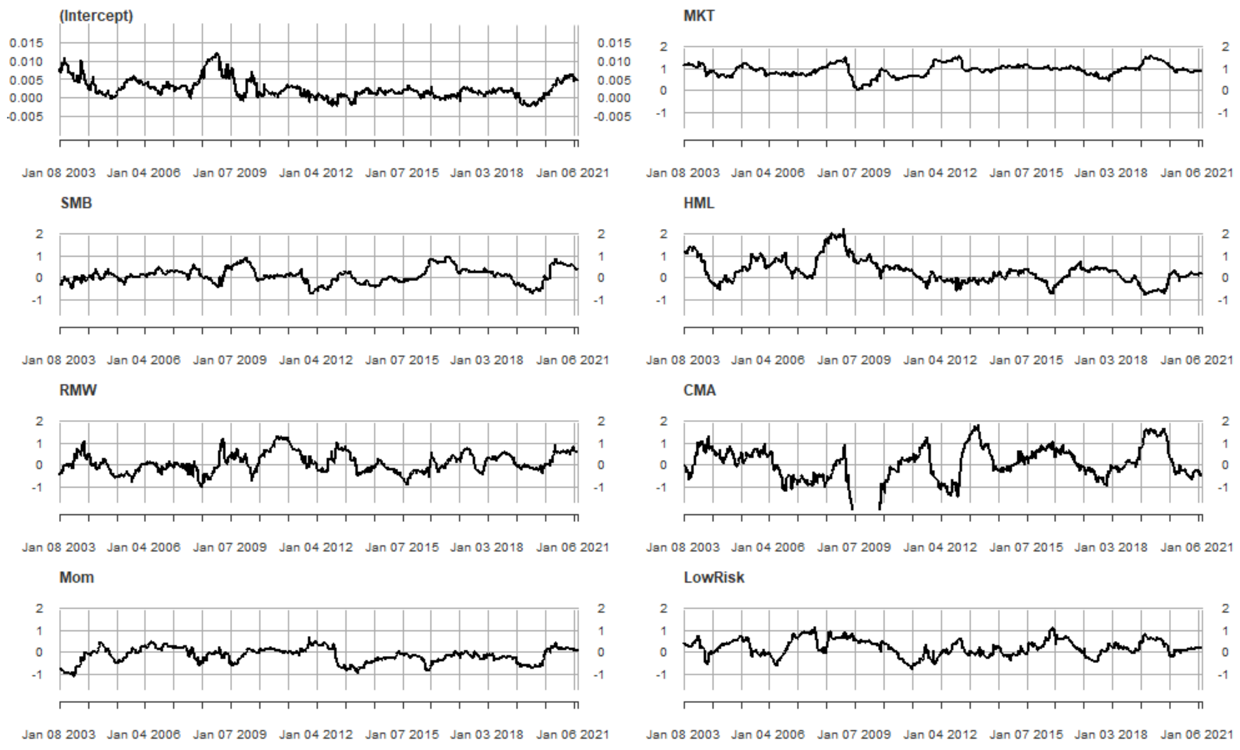
		Alpha	MKT	SMB	HML	RMW	CMA	Mom	BaB	Adj.R2
PCA	Coeff	0.002	0.996	0.029	0.443	-0.554	-0.186	-0.395	0.381	0.592
	t value	1.6	29.8***	-1.1	1.8	-1.8	-2.3*	-1.1	7.3***	
Ridge	Coeff	0.002	0.967	0.039	0.412	-0.344	-0.583	-0.076	0.435	0.539
	t value	2.8**	25.8***	-0.2	5.4***	-3.5***	-4.0***	-2.3*	5.6***	
LASSO	Coeff	0.001	0.974	0.034	0.430	-0.370	-0.564	-0.078	0.442	0.540
	t value	2.5*	25.5***	-0.5	5.0***	-4.0***	-3.8***	-1.9	5.6***	
ENet	Coeff	0.002	0.975	0.029	0.433	-0.344	-0.585	-0.067	0.423	0.545
	t value	2.7**	25.4***	-0.3	5.3***	-4.0***	-4***	-1.8	5.6***	
RF	Coeff	0.002	0.921	0.018	0.249	-0.387	-0.225	-0.258	0.330	0.584
	t value	2.5*	30.1***	-0.7	4.8***	-2.8**	-2.7**	-1.4	5.8***	
Boosting	Coeff	0.002	0.918	-0.004	0.191	-0.410	-0.052	-0.322	0.331	0.584
	t value	2.3*	27.5***	1.8	5.2***	-2.4*	-0.7	-8.9***	5.1***	
DNN	Coeff	0.002	0.897	0.002	0.231	-0.572	-0.169	-0.323	0.306	0.589
	t value	2.5*	25.3***	2.2*	5.6***	-4.6***	-2.6**	-6.0***	5.8***	
LSTM	Coeff	0.002	0.909	0.051	0.210	-0.486	-0.163	-0.338	0.285	0.610
	t value	3.1**	26.2***	0.1	3.4**	-5.9***	0.1	-9.0***	4.3***	
Ensemble	Coeff	0.002	0.930	0.015	0.276	-0.467	-0.285	-0.213	0.313	0.568
	t value	3.4**	26.9***	0.2	4.6***	-4.2***	-2.4*	-4.7***	5***	

The table displays coefficient estimates and the respective t-values for factor regressions when regressing the performance of a ML based trading strategy on common equity factors including market (“MKT”), size (“SMB”), value (“HML”), quality (“RMW”), investment (“CMA”), momentum (“MOM”) and the betting-against-beta factor (“BaB”). The trading strategy invests in 50 stocks (equally weighted) with the largest predictions based on the machine learning model specified in the first column. ‘***’, ‘**’, ‘*’ indicate significance at the 0.1%, 1% and 5% significance level, respectively for the null hypothesis that the respective coefficient equals zero.

Most importantly, table 6 shows that after controlling for the common risk factors a positive and statistically significant alpha remains of around 10-20 basis points per week for all models. Hence, while the outperformance of the machine learning strategies is partly attributable to known risk factors, after controlling for these factors, a statistically and economically significant alpha remains for all models.

Finally, we analyze rolling factor exposures of the ensemble model to derive insights whether the strategy’s factor exposures are stable or fluctuating over time. Figure 7 presents 52-weeks (1 calendar year) rolling factor exposures of the ensemble model. The Figure shows that factor exposures substantially fluctuate over time indicating that the machine learning strategy does not follow a certain factor strategy but rather performs implicit factor timing.

FIGURE 7: ROLLING FACTOR EXPOSURES OF THE ENSEMBLE MODEL



The table displays 52-weeks rolling coefficient estimates when regressing the performance of the ensemble strategy on common equity factors including market ("MKT"), size ("SMB"), value ("HML"), quality ("RMW"), investment ("CMA"), momentum ("MOM") and the betting-against-beta factor ("BaB"). The trading strategy invests in 50 stocks (equally weighted) with the largest predictions based on the ensemble machine learning model.

4.6 FEATURE IMPORTANCE

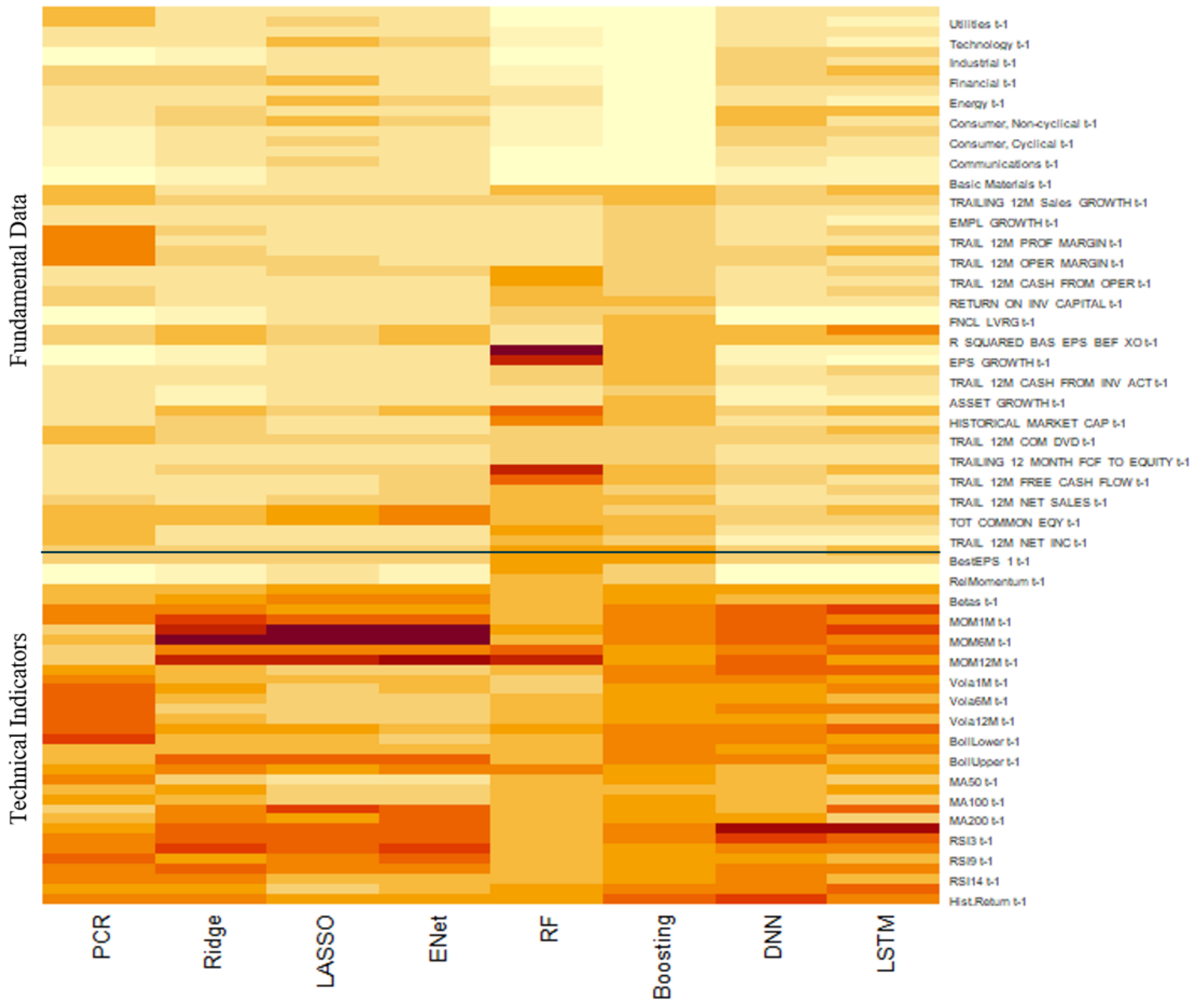
Next, we analyze feature importance to shed light on how machine learning models arrive at decisions. More specifically, we analyze the relative importance of different stock features for a model's decision to classify a stock either as an under- or outperformer. For our relatively short prediction horizon of only one week, one may argue that technical features are more important than the underlying fundamental firm characteristics. We address this question twofold: First, we compute feature importance measures based on Shapley values. Second, we compute the performance of our ML strategies when trained only with fundamental data or only with technical indicators.

A recent approach to determine feature importance for machine learning models was developed by Lundberg and Lee (2017). Their approach builds on the theoretical concept of Shapley values from game theory (Shapley, 1953) which quantifies the contribution of each player in a game to the game's outcome. Analogously, the SHAP (SHapley Additive exPlanations) value of a feature reflects the contribution of the feature to the prediction of a model. Figure 8 illustrates the feature importance (SHAP values) for the different features for each machine learning model. Darker coloring corresponds to a higher feature importance. The upper half of figure 8 presents the fundamental stock data summarized in Panel A of table 1. The lower half of figure 8 illustrates technical indicators presented in Panel B of table 1. Figure 8 shows darker coloring for the technical indicators for all models visualizing that for the relatively short prediction horizon of one week, technical indicators play a larger role than fundamental data in classifying a stock as under- or outperformer. Particularly, momentum (1 month, 6 months and 12 months) plays an important role in the logistic regression models, whereas the relative strength index (3 days) (reversal) seems to be the most important feature for neural nets (DNN and LSTM). In contrast, by construction tree-based machine learning modes random forest and boosting account more evenly for fundamental data and technical indicators (only a random subset of features is allowed when growing each tree, forcing more features to be included in trees). For boosting particularly the earnings per share growth and the cash flow to price ratio play an important role for classifying stocks in under- or outperformer.

Next, we explore the relative importance of fundamental and technical features for the performance of the ML based trading strategies. Table 7 Panel A presents the performance of machine learning models when only relying on fundamental data and sector dummies. Table 7 Panel B provides the same performance measures when ML models are trained only with technical indicators. For comparison, Panel C provides the original results for the full feature set. Table 7 provides an interesting insight: regression-based and tree-based models provide a superior performance with fundamental data compared to technical indicators.

In contrast, for neural networks (DNN and LSTM) we find an opposed picture: these models provide a superior performance for technical indicators and adding fundamental data does not improve model performance. This might be due to the high dynamic in technical indicators and the strength of LSTMs to process long and short-term dependencies. Hence, a promising strategy might be to rely on a LSTM, employing further technical indicators and a longer history of data, feeding not only the last two observations into the model but a longer sequence. We leave this idea to further research.

FIGURE 8: FEATURE IMPORTANCE



This figure illustrates the feature importance (SHAP values) for the different features for each machine learning model. Darker coloring corresponds to a higher feature importance.-

TABLE 7: PERFORMANCE FOR DIFFERENT FEATURE GROUPS

Panel A: Fundamental Data	**Return p.a.**	**Vola p.a.**	**Sharpe p.a.**	**Max. Drawdown**	**CAPM Alpha p.a.**	**FF-6 Alpha p.a.**	**Turn-over p.a.**	**BTC vs. 1/N**
Market (S&P 500)	6.60%	16.70%	0.40	-55.40%	0.00	0.00		
Benchmark (1/N)	11.1%	19.3%	0.57	-54.8%	0.04*	0.05*	0.8	0.00%
PCA	15.5%	21.3%	0.73	-48.6%	0.08***	0.1**	6.03	0.36%
Ridge	17.2%	23.0%	0.75	-54.6%	0.09***	0.11**	8.65	0.35%
LASSO	19.8%	24.6%	0.81	-48.1%	0.11***	0.14***	12.58	0.35%
ENet	18.2%	24.4%	0.75	-54.3%	0.1***	0.12**	11.03	0.32%
RF	15.9%	22.2%	0.72	-52.3%	0.08**	0.1**	10.91	0.22%
Boosting	16.3%	22.6%	0.72	-45.7%	0.08**	0.12***	21.59	0.12%
DNN	11.5%	24.3%	0.47	-57.6%	0.04	0.07*	22.04	0.01%
LSTM	13.3%	24.0%	0.56	-51.2%	0.05*	0.08*	23.75	0.05%
Ensemble	16.3%	23.2%	0.70	-47.1%	0.08**	0.11**	20.64	0.13%
Panel B: Technical Indicators	**Return p.a.**	**Vola p.a.**	**Sharpe p.a.**	**Max. Drawdown**	**CAPM Alpha p.a.**	**FF-6 Alpha p.a.**	**Turn-over p.a.**	**BTC vs. 1/N**
Market (S&P 500)	6.60%	16.70%	0.40	-55.40%	0.00	0.00		
Benchmark (1/N)	11.1%	19.3%	0.57	-54.8%	0.04*	0.05*	0.8	0.00%
PCA	11.2%	22.1%	0.51	-63.4%	0.05	0.06	21.0	0.00%
Ridge	15.5%	24.3%	0.64	-43.9%	0.08*	0.11**	24.1	0.09%
LASSO	13.2%	24.8%	0.53	-44.8%	0.06	0.08*	27.2	0.04%
ENet	13.8%	25.2%	0.55	-44.1%	0.07	0.09*	28.3	0.05%
RF	12.9%	23.7%	0.55	-47.8%	0.06	0.08*	27.9	0.03%
Boosting	15.6%	22.1%	0.71	-49.7%	0.08**	0.10**	39.2	0.06%
DNN	21.4%	27.4%	0.78	-40.7%	0.13***	0.18***	36.5	0.14%
LSTM	18.1%	24.4%	0.74	-54.6%	0.10***	0.13***	37.2	0.09%
Ensemble	19.3%	24.2%	0.80	-36.4%	0.11***	0.15***	36.7	0.11%
Panel C: Fundamental & Technical	**Return p.a.**	**Vola p.a.**	**Sharpe p.a.**	**Max. Drawdown**	**CAPM Alpha p.a.**	**FF-6 Alpha p.a.**	**Turn-over p.a.**	**BTC vs. 1/N**
Market (S&P 500)	6.60%	16.70%	0.40	-55.40%	0.00	0.00		
Benchmark (1/N)	11.1%	19.3%	0.57	-54.8%	0.04*	0.05*	0.8	0.00%
PCA	14.8%	21.1%	0.70	-48.2%	0.08**	0.08*	17.4	0.11%
Ridge	19.4%	25.1%	0.77	-45.2%	0.11***	0.14***	16.4	0.25%
LASSO	18.6%	25.5%	0.73	-46.5%	0.10**	0.13**	19.4	0.19%
ENet	19.3%	25.5%	0.76	-46.3%	0.11***	0.14***	19.9	0.21%
RF	16.9%	22.7%	0.75	-50.6%	0.09***	0.11**	18.7	0.16%
Boosting	15.5%	24.4%	0.63	-55.1%	0.08**	0.10**	32.8	0.07%
DNN	17.2%	25.9%	0.67	-52.4%	0.09**	0.12**	28.0	0.11%
LSTM	18.1%	25.6%	0.71	-50.1%	0.10**	0.13***	31.1	0.11%
Ensemble	20.8%	24.8%	0.84	-44.8%	0.12***	0.15***	28.8	0.17%

The table shows the performance of the different ML trading strategies based on either fundamental data or technical indicators or both (base case). '***', '**', '*' indicate significance at the 0.1%, 1% and 5% significance level, respectively for the null hypothesis that the alpha of a trading strategy equals zero.

4.7 Alternative Asset Universes

To check the robustness of our results, we apply the machine learning models on the constituents of the STOXX Europe 600 as an alternative asset universe. The STOXX Europe 600 represents large, mid and small capitalization companies across 17 European countries.[12] We do not conduct any new feature search, but simply apply the models designed for the S&P 500 on the STOXX 600. Due to availability of data, our analysis for the STOXX 600 starts three years later than for the S&P 500 with the out-of sample evaluation period covering the years from January 2005 to March 2021.[13] Table 8 and Figure 9 present the performance of the machine learning strategies for the constituents of the STOXX 600. In line with the S&P 500, we set the portfolio size to 50 stocks. The results for the STOXX 600 confirm our results for the S&P 500. We find that all machine learning models outperform an equally weighted benchmark portfolio with the ensemble model providing the highest raw and risk-adjusted returns. In line with the results for the S&P 500, for the STOXX Europe 600 regularized logistic regression models (Ridge, LASSO and the Elastic net) work slightly better than the more complex models Boosting, DNN and LSTM. Overall, the results for the STOXX 600 confirms our finding that machine learning models successfully select attractive stocks and that a stock picking strategy based on ML adds value over a passive index investment and over the 1/N benchmark. The findings for the STOXX 600 provide

[12] Austria, Belgium, Denmark, Finland, France, Germany, Ireland, Italy, Luxembourg, the Netherlands, Norway, Poland, Portugal, Spain, Sweden, Switzerland and the United Kingdom.

[13] Our training dataset for the STOXX 600 starts in January 2002. Due to the unavailability of trading volume data we had to exclude the variable PX_Volume for the Stoxx 600.

an important robustness check since the ML models were originally designed for the S&P 500 and were then applied on a completely new dataset without adjusting the model's architecture.[14]

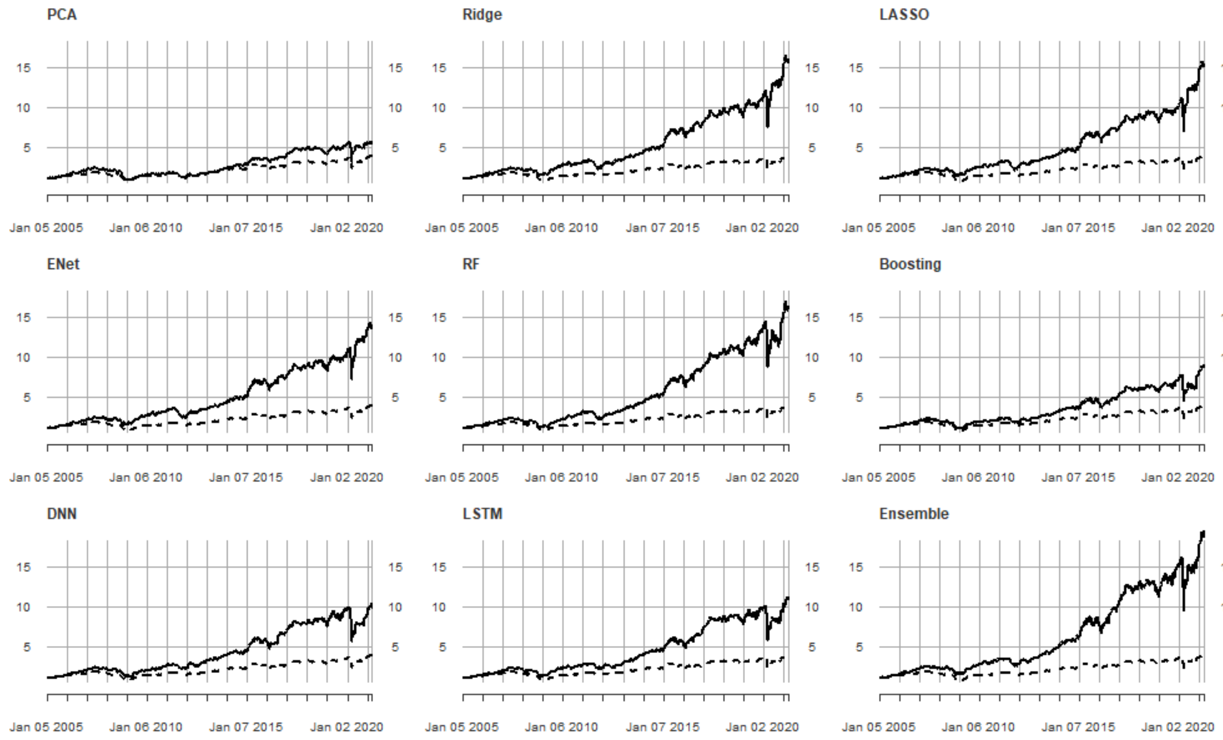
TABLE 8: PERFORMANCE OF MACHINE LEARNING MODELS FOR ALTERNATIVE ASSET UNIVERSE (STOXX 600)

	Return p.a.	**Vola p.a.**	**Sharpe p.a.**	**Max. Drawdown**	**CAPM Alpha p.a.**	**FF-6 Alpha p.a.**	**Turn-over p.a.**	**BTC vs. 1/N**
Market (Stoxx 600)	3.4%	17.7%	0.19	-59.1%	0.00	0.00		
Benchmark (1/N)	8.9%	20.0%	0.44	-59.1%	0.05***	0.04	0.83	0.00
PCA	11.4%	26.1%	0.44	-65.8%	0.08*	0.08	25.94	0.05%
Ridge	18.5%	24.5%	0.76	-45.0%	0.14***	0.13**	27.38	0.18%
LASSO	18.3%	24.7%	0.74	-45.1%	0.14***	0.13**	28.08	0.17%
ENet	17.6%	24.0%	0.73	-43.9%	0.13***	0.12**	28.57	0.16%
RF	18.7%	23.4%	0.80	-53.1%	0.14***	0.13**	25.84	0.20%
Boosting	14.4%	24.5%	0.59	-59.4%	0.11***	0.1*	33.85	0.08%
DNN	14.8%	24.1%	0.61	-49.7%	0.11***	0.11*	30.31	0.10%
LSTM	15.3%	23.9%	0.64	-48.6%	0.11***	0.1*	31.45	0.10%
Ensemble	19.6%	24.3%	0.81	-45.5%	0.15***	0.14**	31.01	0.18%

The table shows the performance of the different ML trading strategies for the alternative STOXX 600 universe. '***', '**', '*' indicate significance at the 0.1%, 1% and 5% significance level, respectively for the null hypothesis that the alpha of a trading strategy equals zero.

[14] As for the S&P 500 the ML models were trained once a year based on the previous three years of data and model parameters were tuned using the same grid search approach as for the S&P 500.

FIGURE 9: MODEL PERFORMANCE FOR STOXX 600 UNIVERSE VS. 1/N BENCHMARK (DASHED)



The figure displays the performance of the trading strategies compared to a benchmark portfolio that equally invests in all stocks (1/N benchmark, dashed line) for the alternative STOXX 600 universe.

5. CONCLUSION

In this study, we use machine learning models for stock selection and empirically analyze the performance of deep neural networks (DNN), long short-term neural networks (LSTM), random forest, boosting and regularized logistic regression. We train the models on stock characteristics including the typical equity factors as well as additional fundamental data and technical indicators to predict whether a specific stock outperforms the market over the subsequent week. Our asset universe builds on the historical constituents of the S&P 500 over the period from January 1999 to March 2021. We analyze the risk-adjusted performance of a trading strategy that picks stocks with the highest predictions to outperform. Our empirical results show a substantial and significant risk-adjusted outperformance of machine learning based stock selection models compared to a simple equally weighted benchmark.

The higher returns of machine learning-based stock selection models are not fully explained by the common risk factors and positive and statistically significant alphas remain after controlling for the six Fama and French (2018), and the betting-against-beta (Frazzini and Pederson, 2013) factors. The sub-period analysis indicates that the outperformance cannot be attributed to a single short time period but is present in all 4 sub-periods including periods of expansion and recession. Moreover, our analysis of rolling factor exposures shows, that machine learning strategies do not follow a traditional static factor strategy but exhibit dynamic and fluctuating factor exposures which can be interpreted as implicit factor timing strategies.

Our analysis of different portfolio sizes shows that on the one hand lower portfolio sizes generally yield higher returns since only stocks with the highest predicted outperformance potential are selected. On the other hand, lower portfolio sizes lead to higher idiosyncratic risk due to lower diversification and higher portfolio volatility. Therefore, the sharpe ratio optimal portfolio size is a trade-off of outperformance potential and diversification. Empirically, we find that for our trading strategy the optimal portfolio size is around 50 stocks. Our results are robust when applied on the STOXX Europe 600 as alternative asset universe.

References

Avramov, Doron, Cheng, S., Metzker, L., (2020) Machine learning versus economic restrictions: Evidence from stock return predictability, Working paper.

Bates, J. M., Granger C. W. J. (1969) The Combination of Forecasts, Operational Research Quarterly, 20 (4), 451-468

Bengio, Y., Simard, P., Frasconi, P. (1994) Learning Long-Term Dependencies with Gradient Descent Is Difficult," IEEE Transactions on Neural Networks 5, (2)

Breiman, L, Friedman, J. Stone, J.C, Olshen R.A. (1984) Classification and Regression Trees, The Wadsworth and Brooks-Cole statistics-probability series

Breiman, L. (1986) Bagging Predictors, Machine Learning, 24 (2), 123-140

Carhart, M. M. (1997) On Persistence in Mutual Fund Performance, The Journal of Finance, 52, 57–82.

Chen, L., Pelger, M, Zhu, J. (2019) Deep learning in asset pricing, Working paper.

Chinco, A., Clark-Joseph, A. D. Ye, M. (2019) Sparse signals in the cross-section of returns, Journal of Finance 74, 449-492.

Choi, D. Jiang, W., Zhang, C. (2019) Alpha Go Everywhere: Machine Learning and International Stock Returns, Working Paper

Clemen, R.T. (1989) Combining forecasts: a review and annotated bibliography, International Journal of Forecasting 5 (4), 559–583.

Cong, L.W., Tang, K., Wang, J., Zhang, Y. (2019) AlphaPortfolio and Interpretable AI for Finance, working Paper.

Coqueret, G., Guida, T. (2018) Stock Returns and the Cross-Section of Characteristics: A Tree-Based Approach, Working Paper, 2018

Cybenko, G. (1989) Approximations by superpositions of sigmoidal functions, Mathematics of Control, Signals, and Systems, 2 (4), 303-314

Fama, E.F., French, K.R. (2018) Choosing Factors, Journal of Financial Economics, 128, 2, 234-252.

Feng, G., Polson, N. G, Xu, J. (2018) Deep learning factor alpha, Working paper.

Fischer, T., Krauss, C. (2018) Deep learning with long short-term memory networks for financial market predictions, 270, 2, 654-669

Frazzini, A., Pederson, L.H. (2013) Betting against beta, Journal of Financial Economics, 111, 1, 1-252.

Freund Y., Schapire, R.E. 1997. A decision-theoretic generalization of online learning and an application to boosting. Journal of Computer and System Sciences 55(1): 119–139.

Freyberger, J., Neuhierl, A., Weber, M. (2018) Dissecting characteristics nonparametrically, Working paper.

Friedman, J., Hastie, T., Tibshirani, R. (2001) The Elements of Statistical Learning, New York, NY: Springer.

Gu, S, Kelly, B, Dacheng, X. (2020) Empirical Asset Pricing via Machine Learning, The Review of Financial Studies 33, 2223-2273.

Gu, S, Kelly, B, Dacheng, X. (2019) Autoencoder Asset Pricing Models, Working Paper.

Han, Yufeng, Ai He, David Rapach, and Guofu Zhou (2018) What firm characteristics drive US stock returns? Working paper.

Hendry, D.F., Clements, M.P. (2004) Pooling of forecasts, Econometrics Journal 7 (1), 1–31.

Hinton, G. (2012) Neural Networks for Machine Learning – Lecture slides, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Hochreiter, S., Schmidhuber, J. (1997) Long Short-Term Memory, Neural Computation, 9 (8), 1735-1780.

Hoerl, A.E., Kennard R.W. (1970) Ridge regression: biased estimation for nonorthogonal problems, Technometrics, 12, 1970, 55-67.

Jozefowicz, R. Zaremba, W. Sutskever, I. (2015) An Empirical Exploration of Recurrent Network Architectures Journal of machine Learning, (37)

Ludvigson, S.C., Ng, S. (2007) The empirical risk–return relation: A factor analysis approach, Journal of Financial Economics 83 (1), 171–222.

Lundberg, S. M., Lee, S.I. (2017) A unified approach to interpreting model predictions, Advances in Neural Information Processing Systems, 30, 4765-4774.

Markowitz, H. (1952) Portfolio selection, The Journal of Finance 7 (1), 77–91.

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.

Sak, H., Senior, A., Beaufays, F., (2014) Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. arXiv preprint arXiv:1402.1128.

Shapley, L. S. (1953). A value for N-person games. Contributions to the Theory of Games, 2, 307-317.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever,I., Salakhutdinov R. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research, 15 (56), 1929−1958.

Stock, J.H., Watson, M.W. (2004) Combination forecasts of output growth in a seven-country data set, Journal of Forecasting 23 (6), 405–430.

Tibshirani, R. (1996) Regression shrinkage and selection via the Lasso, Journal of the Royal Statistical Society. Series B (Methodological) 58 (1), 267–288.

Timmermann, A. (2006) Forecast Combinations, Handbook of Economic Forecasting, 1, 135-196.

Wolff, D. Neugebauer, U. (2019) Tree-based machine learning approaches for equity market predictions, Journal of Asset Management, 20, 273–288.

Zou H., Hastie T. (2005) Regularization and variable selection via the Elastic Net, Journal of the Royal Statistical Society, 67, 2005, 301-320.

Appendix

Table 1: Model hyperparameter

Model	Hyperparameter
PCA	PCA Factors n ∈ [1, ...N] where N is the number of PCA factors explaining 90% of feature variance.
Ridge	Penalty parameter λ: 500 values on a logarithmic scale ranging from 0.0001 to 10,000.
LASSO	Penalty parameter λ: 500 values on a logarithmic scale ranging from 0.0001 to 10,000.
ENet	Combination parameter for combining absolute and squared penalties (α) ∈ [0, 0.05...1]
	Penalty parameter λ: 500 values on a logarithmic scale ranging from 0.0001 to 10,000.
RF	Number of trees ∈ [100, 250, 500, **1000**]
	Maximum tree depth ∈ [3, 5, **7**, 10, 15, 20]
	Minimal node size ∈ [1, 3**, 5**, 10]
	Number of randomly allowed predictors in each node: Round off sqrt of the number of predictors
Boosting	Number of iterations: 1000
	Maximum tree depth ∈ [**3**, 5, 7, 10, 15, 20]
	Minimal node size (child weight) ∈ [1, 3, **5**, 10]
	Step size shrinkage parameter (eta) ∈ [**0.01**, 0.05, 0.1, 0.3]
	Subsample (Subsample ratio of the training instances used in each iteration): 0.5 (default)
	Colsample_bytree (subsample of predictors used in each tree) ∈ [**0.5**, 0.7, 0.8, 0.9, 1]
	Minimum loss reduction for additional node of the tree (gamma) ∈ [0, **0.001**, 0.01, 0.1] (default=0)
DNN	Number of hidden layers: 3, Neurons per hidden layer: (20, 10, 5). [Tested alternatives: (10,5,5), (10,10,5), (15,10,5), (20,10,10)]
	Activation function: Relu, Output Layer: Softmax
	Learning Parameter: Loss Function= Binary Crossentropy, Learning Rate: 0.001, Decay = 0, Optimizer: RMSprop, 100 epochs, early stopping=10
	Regularization Parameter: L1 Regularization ∈ [**0.0001**, 0.001, 0.01, 0.1], batch normalization, Dropout rate ∈[**0**, 0.1, 0.2, 0.3, 0.4, 0.5]
LSTM	Number of hidden LSTM layers: 1, Neurons in layer [10,20,25,**30**,35]
	Learning Parameter: Loss Function= Binary Corssentropy, Learning Rate: 0.001, Decay = 0, Optimizer: RMSprop, 100 epochs, early stopping=10
	Regularization Parameter: Dropout rate ∈[0, 0.1, 0.2, 0.3, 0.4, **0.5**]

This table reports the tuning parameters required to train each machine learning model. For PCA, Ridge, LASSO, ENet the parameters are estimated via grid search for each training (and validation) set using cross validation. For RF, Boosting, DNN and LSTM we determine the tuning parameters based on the first training (and validation) set via grid search. Bold parameters highlight the final parameters chosen in the grid search.