Journal of Asset Management (2021) 22:507–538
https://doi.org/10.1057/s41260-021-00237-x

ORIGINAL ARTICLE

Empirical asset pricing via machine learning: evidence from the European stock market

Wolfgang Drobetz[1] · Tizian Otto[1]

Revised: 21 June 2021 / Accepted: 8 August 2021 / Published online: 12 November 2021

Abstract
This paper evaluates the predictive performance of machine learning methods in forecasting European stock returns. Compared to a linear benchmark model, interactions and nonlinear effects help improve the predictive performance. But machine learning models must be adequately trained and tuned to overcome the high dimensionality problem and to avoid overfitting. Across all machine learning methods, the most important predictors are based on price trends and fundamental signals from valuation ratios. However, the models exhibit substantial variation in statistical predictive performance that translate into pronounced differences in economic profitability. The return and risk measures of long-only trading strategies indicate that machine learning models produce sizeable gains relative to our benchmark. Neural networks perform best, also after accounting for transaction costs. A classification-based portfolio formation, utilizing a support vector machine that avoids estimating stock-level expected returns, performs even better than the neural network architecture.

Keywords Stock return prediction · Machine learning · Active trading strategy

JEL Classification G11 · G12 · G14 · G17

Introduction

This paper studies two complementary topics in the empirical asset pricing literature: stock return prediction and machine learning. The traditional forecasting literature applies statistical models to estimate future stock returns along two strands: time series (TS) predictions and cross-sectional (CS) predictions. TS models typically conduct time series regressions of broad aggregate portfolio returns on a number of macroeconomic variables, including valuation ratios and interest rates, and on technical indicators (Cochrane 2011; Rapach and Zhou 2013). CS models, in contrast, usually explain differences in expected returns across stocks as a function of stock-level characteristics, such as size, value, and momentum (Fama and French 1993; Jegadeesh and Titman 1993; Lewellen 2015). This approach typically runs Fama and MacBeth (1973) regressions (FM regressions).

In most applications, in line with the traditional asset pricing literature, TS and CS models consider only linear relationships between financial variables and subsequent stock returns. For example, the Capital Asset Pricing Model (CAPM) introduced by Sharpe (1964), Lintner (1965), and Mossin (1966) posits that, in equilibrium, a stock's expected return is solely driven by its sensitivity to a systematic risk factor, i.e., the market risk. An assumption is that the underlying pricing kernel is *linear* in only a *single* factor, i.e., the market portfolio.[1] Various studies, however, report violations of this assumption (see, e.g., Hou et al. 2020, for a comprehensive list of asset pricing anomalies) and study alternative asset pricing models. Following Dittmar (2002), we classify them into two subcategories. The first subcategory utilizes other pricing factors in addition to the market portfolio. Most prominently, Fama and French (1993) propose

✉ Tizian Otto
tizian.otto@uni-hamburg.de

[1] Faculty of Business Administration, University of Hamburg, Moorweidenstr. 18, 20148 Hamburg, Germany

[1] In modern asset pricing representations, the pricing kernel refers to the intertemporal marginal rate of substitution or stochastic discount factor m_{t+1} in the Euler equation, i.e., $E[\left(1+R_{i,t+1}\right)\times m_{t+1}|\Omega_t]=1$, which expresses the first-order condition of an investor's intertemporal consumption-investment problem. $(1+R_{i,t+1})$ denotes the total return on asset i, and Ω_t is the information set available to investors at time t. The stochastic discount factor m_{t+1} represents the pricing kernel that prices all risky assets, assuming the law of one price holds. It is specified differently in different asset pricing models (see Cochrane 2005, for a textbook treatment).

a multifactor alternative to the CAPM and find that it is better at explaining cross-sectional variation in expected returns than the CAPM. Other examples include Ross' (1976) asset pricing theory (APT) or Merton's (1973) intertemporal CAPM (ICAPM). The second subcategory abandons the restriction that the pricing kernel must be linear in pricing factors. Bansal et al. (1993), Bansal and Viswanathan (1993), Chapman (1997), Dittmar (2002), and Asgharian and Karlsson (2008), among others, explore various nonlinear pricing kernels and show that such specifications outperform linear counterparts.

While the first subcategory of models motivates the use of multiple pricing factors, the second subcategory suggests that using interactions between these factors and incorporating nonlinear relationships between price-related variables and expected stock returns add incremental explanatory power. A problem arises particularly in a prediction context, because the convexity of the traditional least squares objective tends to emphasize heavy-tailed observations. Therefore, with an increasing number of predictors simple linear models begin to overfit noise rather than extract signals, thereby undermining the stability of the predictions. Since financial data are inherently noisy, predictors potentially multicollinear, and the relationships between predictor variables and expected returns likely interactive and nonlinear as well as time-varying and/or contextual, it seems virtually impossible for simple linear models to generate reliable forecasts of future stock returns.

Machine learning can help overcome the high dimensionality problem by combining many weak sources of information into a strong composite forecast. According to Gu et al. (2020; p. 2225), "[t]he definition of machine learning is inchoate and is often context specific. [It] […] describe[s] (i) a diverse collection of high-dimensional models for statistical prediction, combined with (ii) so-called 'regularization' methods for model selection and mitigation of overfit, and (iii) efficient algorithms for searching among a vast number of potential model specifications."

In our empirical analysis, we exploit a set of twenty-two predictors as per the linear FM regressions approach used in Drobetz et al. (2019). This is our benchmark model, and it is able to explain a substantial percentage of the cross-sectional variation in European stock returns. Against this established conservative benchmark, we compare the performance of different machine learning methods in forecasting stock returns, from both a statistical and an economic perspective. Our objective is to examine whether incorporating interactions between predictor variables and nonlinear effects lead to incremental predictive power.

Our empirical analysis follows that of Gu et al. (2020), with several key differences: First, we use European data. Second, we use an established linear benchmark model with a parsimonious set of twenty-two predictors, rather than a comprehensive set of predictor variables from the voluminous literature. Third, in addition to inspecting changes in a forecast model's degree of complexity over time, our tests show the importance of each predictor in a given model over the full sample period, and we also scrutinize the time variation of each predictor variable's importance. This approach can reveal whether (1) some predictors are uninformative during the entire sample period, (2) they lead to a permanent deterioration in a forecast's signal-to-noise ratio, and (3) they can be removed from the set of baseline variables. Fourth, we apply a conservative transaction cost model to investigate whether the value-added of active trading strategies remains promising under realistic trading assumptions.

Finally, we challenge the traditional formation process of expected return-sorted portfolios, which uses some statistical prediction model to estimate expected returns first, and selects stocks with the highest expected return forecasts into decile portfolios second. The underlying assumption is that stocks with high expected returns ex ante will deliver high realized returns ex post. Rather than taking the "detour" to estimate stock-level expected returns, we take an alternative approach here. In particular, we use a support vector machine to directly classify stocks into decile portfolios based on linear combinations of predictor variables.

In our empirical analysis, we enhance the set of twenty-two predictors from Drobetz et al. (2019) by two-way interactions as well as second- and third-order polynomials to capture nonlinearity. Overall, we find that interactions and nonlinear effects help improve the predictive performance. However, machine learning models must be adequately trained and tuned to avoid overfitting.

Overfitting can manifest along two different strands: model overfitting and backtest overfitting. Model overfitting refers to machine learning models with overly high in-sample fit but poor out-of-sample predictive performance. To avoid model overfitting, we control for the degree of model complexity by tuning relevant hyperparameters. These parameters cannot be preset, but must be determined adaptively from the sample data. Backtest overfitting refers to a researcher's arbitrariness in choosing firm coverage, sample period, predictors, and tuning parameters. If information from the out-of-sample period is used to fit the models, consciously or not (Schorfheide and Wolpin 2012), this might lead to overstated out-of-sample predictive performance (Bailey et al. 2014, 2017; Harvey and Liu 2014, 2015; Harvey et al. 2016). To avoid backtest overfitting, we include all firms that were or are publicly listed in one of the nineteen Eurozone countries as of December 2020. According to Gu et al. (2020), using large datasets mitigates sample selection or data snooping biases (Lo and MacKinlay 1990) and also can help avoid model overfitting by increasing the ratio of observation count to parameter count. In line with Drobetz et al. (2019), we use a set of twenty-two firm-level predictors that are commonly used in the asset pricing literature (Chordia et al. 2017; Harvey et al. 2016; Lewellen 2015; McLean and Pontiff 2016). We refrain from focusing on only a small subset of covariates that have been shown to perform best in similar prediction tasks. Finally, we opt for the time series cross-validation approach to fit the machine learning models. It maintains the temporal ordering of the data and splits the sample into three distinct subsamples: a training sample, a validation sample, and a test sample (which is used for neither model estimation nor parameter tuning, and thus is truly out-of-sample).[2]

We observe that each forecast model's degree of complexity varies substantially over time, but they all denote similar predictors as important. The most influential predictors are based on recent price trends, e.g., short-term reversal and stock momentum, and fundamental signals from valuation ratios, e.g., earnings-to-price and book-to-market ratios. Despite these commonalities, our forecast models exhibit differences in statistical predictive performance (as indicated by predictive slopes, predictive R^2 metrics, and Diebold and Mariano (1995) test statistics), which also translate into marked differences in economic profitability. The return and risk measures of long-only investment strategies, i.e., sorting stocks into decile portfolios based on expected return estimates and buying the top decile portfolio, indicate that machine learning methods can produce predictive gains. These gains are attributable to predictor interactions and to nonlinear effects that is overlooked by the linear benchmark model. We find that neural networks perform the best overall, also after accounting for transaction costs.

Finally, we compare the performance of traditional expected return-based portfolio formation with a classification-based approach. In particular, we contrast neural networks (which outperform all other traditional approaches in our empirical analysis) with support vector machines (SVMs). The idea behind SVMs is to search for hyperplanes that territorially divide a multidimensional vector space (our sample of firm observations, consisting of stock-level predictors and decile portfolio labels) into groups of vectors that belong to the same class. This allows for the classification of stocks into decile portfolios without taking the "detour" to predict stock-level expected returns. We find that the classification-based approach is superior to even the best-performing expected return-based portfolio formation because it avoids some of the noise in stock-level returns. Most importantly, it is able to maintain the broad return signal, which is essential for our trading strategy, i.e., the correct classification into a decile portfolio.

The remainder of this paper is organized as follows: The second section briefly reviews the literature on stock return prediction and machine learning. The third section describes our dataset. The fourth section summarizes the different machine learning methods and tuning parameters, while the fifth section compares their predictive performance. We contrast the traditional formation process of expected return-sorted portfolios with an alternative approach using classification-based portfolios in the sixth section. The seventh section concludes.

Literature review

Several prior studies are related to our paper. Haugen and Baker (1996), Lewellen (2015), and Drobetz et al. (2019) examine composite estimates of expected excess returns obtained from cross-sectional FM regressions. They present comprehensive evidence that the simultaneous incorporation of multiple firm characteristics as predictors provides strong out-of-sample predictive power for subsequent stock returns. They also challenge the efficient market hypothesis by concluding that stocks with higher expected (and realized) returns are not riskier than their low-return counterparts. In turn, they show that stock return predictions can be used to implement a profitable investment strategy.

Research on machine learning methods in the asset pricing literature is still scant, and different forecast approaches are usually examined separately. Most studies compare different specifications from the same model family, e.g., approaches that conduct variable selection/shrinkage or dimension reduction, tree-based models, and neural networks. For example, Kozak et al. (2020) use the joint explanatory power of a large set of cross-sectional stock return predictors to form a robust stochastic discount factor, contrasting the lasso and ridge penalizations.

Freyberger et al. (2020) and Rapach et al. (2013) also use the lasso framework. The former applies a nonparametric adaptive group lasso version to explore which characteristics add incremental predictive power for the cross-section of expected returns. The latter considers country-level stock returns to examine lead–lag relationships. Giglio and Xiu (2021) focus on a dimension reduction path. They apply a principal component analysis (PCA) method to estimate the risk premiums of factors that are potentially omitted in asset pricing models due to missing or limited observability. Kelly et al. (2019) introduce an instrumentalized version of the PCA approach to re-estimate common equity factors. Incorporating tree-based architectures, Coqueret and Guida (2018) use single regression trees to estimate the association of firm characteristics and subsequent stock returns. Both Leung et al. (2021) and Moritz and Zimmermann (2016) apply ensemble versions that avoid overfitting to study how common equity factors and lagged stock returns predict future stock returns.

[2] The hyperparameters are selected from a comprehensive set of parameter specifications (see Appendix 1, Table 9 for details), following common parameter choices in the literature. We cover a representative range of possible parameter specifications, which can help mitigate the risk of backtest overfitting.

Several studies apply neural networks in various specifications, most commonly feedforward multilayer architectures. For example, Levin (1995) and Messmer (2017) examine the predictive power of firm characteristics on subsequent stock returns, while Heaton et al. (2016) assess the applicability of deep learning approaches for multiple regression and classification problems in finance. Chen et al. (2019) and Gu et al. (2021) consider other architectures, such as recurrent long short-term memory networks, generative adversarial networks, and autoencoder networks. Rasekhschaffe and Jones (2019), Wolff and Neugebauer (2019), and Gu et al. (2020) provide a comparison of different forecast models.

Finally, Leung et al. (2000) focus on the sign rather than the level of stock returns. They apply classification-based methods (including linear discriminant analysis, logit, and probit) to predict the direction of stock market movements. Similarly, Huerta et al. (2013) train an SVM to classify stocks into future out- and underperformers.

Data

The market and fundamental data for European firms come from Thomson Reuters Datastream. Our sample is free of survivorship bias, and includes all firms that were or are publicly listed in one of the nineteen Eurozone countries as of December 2020.[3] We collect all data on a monthly basis, and, if currency-related, denominated in Euro. We only include firm observations that provide full information on excess returns and all twenty-two firm characteristics used in the empirical analysis (no missing values). In every month, we require at least fifty firms with a minimum market capitalization of €25 mill. that meet this full information criterion. This shrinks our sample period to January 1990–December 2020. To calculate excess returns, we use the three-month EURIBOR rate, scaled to the one-month horizon, as the risk-free rate.

We use twenty-two firm-level predictors that are commonly used in the asset pricing literature (Chordia et al. 2017; Harvey et al. 2016; Lewellen 2015; McLean and Pontiff 2016). These variables have been shown to explain most of the cross-sectional variation in expected returns. We also consider two-way interactions between firm characteristics as predictors as well as second- and third-order polynomials of firm characteristics to cover nonlinearity. We assume that market data become public immediately, while fundamental data are known four months after the fiscal year-end. Table 1 gives the definitions of the twenty-two predictors that serve as our starting point for the creation of an extended set of covariates.

We follow Drobetz et al. (2019) in constructing our twenty-two variables, who note two important caveats: First, most characteristics represent level variables that change slowly (like size) or flow variables that are measured over at least one year (like book equity). This translates into high persistence over time, which in turn suggests that any predictability in monthly excess returns is likely to extend to longer horizons (Campbell and Cochrane 1999; Cochrane 2008). Second, many of the characteristics are constructed similarly, i.e., issuance metrics, or incorporate similar firm information, i.e., profitability measures, which leads to high correlations. According to Lewellen (2015), the resulting multicollinearity is not of concern in our empirical analysis. This is because we are mostly interested in the overall predictive power of the respective machine learning models, rather than the marginal effects of each single predictor. Machine learning models are suitable for solving the multicollinearity problem by selecting/shrinking variables, etc., and could thus add incremental predictive power.

We adjust our initial sample in two steps: First, we winsorize all monthly firm characteristics at the 1% and 99% levels to correct for outliers. In contrast to Gu et al. (2020), we also winsorize excess returns.[4] Second, we rank all firm characteristics month-by-month cross-sectionally and then map the ranks into the $(-1, +1)$ interval, following Kelly et al. (2019) and Freyberger et al. (2020).

Table 1 also reports the time series averages of monthly cross-sectional means and standard deviations for excess returns and the twenty-two characteristics as well as the overall sample size. The average universe of stocks covers 832 firms with non-missing observations, ranging from 56 in January 1990 to 1,012 in December 2020. The average monthly excess return is 0.51%, with a standard deviation of 4.55%. The first and second moments of all predictor variables are similar to those found in earlier studies (Lewellen 2015; Drobetz et al. 2019).

[3] The countries in the Eurozone are: Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Portugal, Slovakia, Slovenia, and Spain.

[4] Gu et al. (2020) apply a Huber loss function to least squares-based forecast models to ensure stable forecasts in the presence of extreme observations. However, if the Huber loss is not applicable (such as in the pcr or pls models), this approach undermines the different models' comparability. Winsorizing excess returns avoids differences in predictive power that are driven by differences in the underlying objective function. However, when we do not winsorize excess returns, the results remain qualitatively the same throughout our empirical analysis.

Table 1 Descriptive statistics

#		Avg	Std	Definition
	Excess return [%]	0.51	4.55	Return in excess of the 3-month EURIBOR rate (scaled to the 1-month horizon)
1	size	13.22	0.48	Log market value of equity at the end of the prior month
2	bm	− 0.55	0.18	Log book value of equity minus log market value of equity at the end of the prior month
3	operatingprofitability	0.06	0.01	Operating profit divided by average total assets in the prior fiscal year
4	totalassetgrowth	0.07	0.04	Log total asset growth in the prior fiscal year
5	ret_2_12	0.09	0.20	Excess return from month − 12 to month − 2
6	issues_1_36	0.09	0.02	Log growth in split-adjusted shares outstanding from month − 36 to month − 1
7	accrualschange	− 0.04	0.01	Working capital accruals (based on Sloan 1996) in the prior fiscal year
8	roa	0.05	0.01	Income before extraordinary items divided by average total assets (return on assets) in the prior fiscal year
9	investment	1.02	0.08	Capital expenditures divided by net sales and revenues in the prior fiscal year
10	ret_1	0.01	0.05	Excess return of the prior month
11	ret_12_36	0.09	0.31	Excess return from month − 36 to month − 12
12	dy	0.03	0.01	Dividends per share over the prior 12 months divided by price at the end of the prior month
13	beta	0.83	0.10	Market beta, estimated from weekly returns from month − 36 to month − 1
14	vola	0.32	0.06	Standard deviation, estimated from weekly returns from month − 12 to month − 1
15	turnover	0.02	0.01	Average monthly turnover volume from month − 12 to month − 1
16	debttoprice	0.84	0.32	Short-term plus long-term debt in the prior fiscal year divided by market capitalization at the end of the prior month
17	salestoprice	2.16	0.60	Net sales and revenues in the prior fiscal year divided by market capitalization at the end of the prior month
18	cftoprice	0.14	0.04	Net income plus depreciation, depletion and amortization in the prior fiscal year divided by market capitalization at the end of the prior month
19	earningstoprice	0.04	0.02	Net income in the prior fiscal year divided by market capitalization at the end of the prior month
20	issues_1_12	0.02	0.01	Log growth in split-adjusted shares outstanding from month − 12 to month − 1
21	fcdispersion	− 2.05	0.18	Log standard deviation of I/B/E/S EPS forecasts minus log average absolute value of I/B/E/S EPS forecasts from month − 12 to month − 1
22	grossprofitability	0.30	0.03	Net sales and revenues minus costs of goods sold divided by average total assets in the prior fiscal year
	Sample size	832		Number of firms with non-missing observations for the excess return and each of the twenty-two characteristics

This table presents the definitions and reports the time series averages of monthly cross-sectional mean and standard deviation for the excess return and each of the twenty-two characteristics as well as the overall sample size. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end

Machine learning methods

For our empirical analysis, our benchmark is the modified linear FM regressions approach used in Drobetz et al. (2019). Our main objective is to examine whether incorporating interactions and nonlinearity adds incremental predictive power. Therefore, we compare the predictive performance of different machine learning methods from both a statistical and an economic perspective. As in Gu et al. (2020), we apply an additive prediction error model to describe a stock's excess return, i.e., $r_{i,t+1} = E_t\left(r_{i,t+1}\right) + e_{i,t+1}$, where $r_{i,t+1}$ is the excess return of stock $i = 1, \dots, N_t$ in month $t = 1, \dots, T$. The expected excess return is estimated as a function of predictor variables and described by the "true" model $g^*\left(z_{i,t}\right)$, where $z_{i,t}$ represents the P-dimensional set of predictors, i.e., $E_t\left(r_{i,t+1}\right) = g^*\left(z_{i,t}\right)$. Albeit our forecast models belong to different families, they are all designed to approximate the true forecast model by minimizing the out-of-sample mean squared forecast error (MSFE), i.e., $\text{MSFE}_{t+1} = \frac{1}{N_{t+1}} \sum_{i=1}^{N_{t+1}} \left(\hat{e}_{i,t+1}\right)^2$. $\hat{e}_{i,t+1}$ is the individual error for stock i stemming from a particular forecast model, and N_{t+1} is the number of stocks within the respective month $t + 1$. Approximations of conditional expectations $g^*\left(z_{i,t}\right)$ are flexible and family-specific. Approximation functions $g(.)$ can be linear or nonlinear, as well as parametric, $g\left(z_{i,t},\theta\right)$, where θ is the set of true parameters, or nonparametric $g\left(z_{i,t}\right)$.[5]

Before presenting the results of our empirical analysis, we now briefly introduce the sample-splitting scheme used to fit the forecast models. We then discuss each machine learning method and its key tuning parameters (see Appendix 1, Table 9 for details).[6]

Sample splitting

Machine learning methods can help overcome the high dimensionality problem, which arises when the number of predictors becomes very large relative to the number of observations. They can also help solve the problem of multicollinearity, which can occur when predictor variables are highly correlated. However, they are prone to overfitting, so we must control for the degree of model complexity by tuning the relevant hyperparameters. Examples for tuning parameters are the number and/or depth of trees in boosted regression trees or random forests. To avoid overfitting and maximize out-of-sample predictive power, hyperparameters cannot be preset, but must be determined adaptively from the sample data. In particular, they are selected from a comprehensive set of parameter specifications (see Appendix 1, Table 9 for details). The parameter tuning approach iteratively reduces in-sample fit by searching for the degree of model complexity that will produce reliable out-of-sample predictive performance. To this end, it uses the time series cross-validation approach, which maintains the temporal ordering of the data, and splits the sample into three distinct subsamples: a training sample, a validation sample, and a test sample.

The training sample is used to estimate the model for multiple parameter specifications, while the validation sample is used to tune the parameters. Based on the models estimated from the training sample, the MSFE within the validation sample is calculated for each parameter specification.[7] The model with the parameter specification that minimizes the MSFE is used for out-of-sample testing. Note that, because the tuning parameters are chosen from the validation sample, it is not truly out-of-sample. The test sample, however, is used for neither model estimation nor parameter tuning. This is why it is truly out-of-sample and appropriate for evaluating a model's out-of-sample predictive power.

In an asset management context, where new data emerge over time, a sample-splitting scheme that periodically includes more recent data must be applied (see, e.g., West 2006, for an overview). This is why the "rolling window" and "recursive window" methods gradually shift the training and validation samples forward in time. The former method holds the length of training and validation samples constant; the latter increases them progressively. Moreover, because the recursive window approach always incorporates the entire history of data, it is computationally more intensive than the rolling window approach.

Because of this, and because machine learning algorithms are generally computationally intensive, Gu et al. (2020) use a hybrid of the rolling and recursive approaches. They avoid recursively refitting models each month. Instead, they refit once every year, as most of the fundamental firm characteristics are updated annually. Each time, they increase the training sample by one year, while holding the length of the validation sample constant but rolling it forward one year. To maintain comparability with Drobetz et al. (2019), we choose the rolling window approach in our empirical analysis. Instead of annually increasing the training sample, we also hold the length of the training sample constant, and we roll it forward from year to year. We always select eight years for training, two years for validation, and one year for testing from the data.[8]

Forecast models

We use two nested sets of predictors. The parsimonious set covers the twenty-two baseline predictors used by Drobetz et al. (2019), enhanced by forty-four dummies that correspond to the ISO country codes and the first two digits of Standard Industrial Classification (SIC) codes. This set includes $22 + 44 = 66$ covariates. The full set adds two-way interactions between the twenty-two baseline predictors as well as second- and third-order polynomials. In total, it includes $22 + \frac{22\times21}{2} + 22 + 22 + 44 = 341$ covariates.

While all forecast models aim to minimize the MSFE, they differ in their overall approach and complexity. Therefore, we outline the major differences among the model families. We begin with simple linear regression models, continue with models that perform variable selection/shrinkage

[5] As in Gu et al. (2020), we make the following assumptions: $g(.)$ and $g^*(.)$ predict expected returns independently of any information prior to t or information from other stocks at t. $g(.)$ and $g^*(.)$ differ between the different forecast models, but they are identical across all stocks and over time within each model family. This leads to more stable estimates of expected excess returns.

[6] We present the forecast models from an intuitive perspective. For more details on the specific implementation choices of each machine learning algorithm, see Gu et al. (2020).

[7] While simple linear regression models (see the "Forecast models" section) do not require parameter tuning (based on the validation sample), we also estimate the models only from the training sample. This enhances the comparability with the machine learning-based forecast models. Note that the main findings of the empirical analysis are qualitatively similar if we use the training and validation samples together to estimate the models at each re-estimation date.

[8] The first models are estimated for training and validation from ten years of data (1990:01–1997:12 and 1998:01–1999:12, respectively). Therefore, twenty-one years of data (2000:01–2020:12) remain for testing.

or dimension reduction, and finish with other sophisticated machine learning methods. Furthermore, for each method, we visually examine how the expected return estimates change as we vary the values of a single predictor or a pair of predictors simultaneously within the $(-1, +1)$ interval. We hold all other predictors fixed at their uninformative median value of zero. The visualizations (unreported) confirm that the models that are designed to incorporate interactions and nonlinearity indeed capture these effects.

Ordinary least squares

Ordinary least squares (OLS) regression models are the least complex approach in our empirical analysis. Their aim is to minimize the standard "l_2" objective function, i.e., $l(\theta) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(r_{i,t+1} - g\left(z_{i,t};\theta\right)\right)^2$. We further distinguish between two similar but different model specifications. The first is denoted as "ols_pars" and regresses monthly excess returns on the *parsimonious* set of sixty-six predictors. Using a similar or identical set of covariates, Lewellen (2015) and Drobetz et al. (2019) document that this approach is promising, both from a statistical and an economic perspective. Although they use the FM regressions model that is re-estimated on a monthly basis, the ols_pars model provides nearly identical predictions in our sample-splitting and re-estimation setting.

In particular, we replicate the linear FM regression approach implemented in Drobetz et al. (2019) and compare the expected excess return series. We find a Pearson correlation coefficient close to 1, which can be explained by the rolling window sample splitting with annual re-estimation. In this case, running cross-sectional FM regressions on a monthly basis and averaging coefficients recursively (as in Lewellen 2015, or Drobetz et al. 2019) are very similar to running a pooled OLS regression based on a rolling training sample (as in our empirical analysis).

To ensure comparability with the machine learning models that cannot be re-estimated on a monthly basis (due to computational limitations), we use the ols_pars model as a proxy for the FM regressions approach and our main benchmark. It performs well and thus serves as a conservative threshold for indicating incremental predictive performance. However, the ols_pars model cannot capture interactions or nonlinearity that may add predictive power.

This is why the second model specification is denoted as "ols_full" and regresses monthly excess returns on the *full* set of 341 predictors. Because we now face a high-dimensionality problem, we expect the ols_full model to perform worse than the ols_pars model. It incorporates predictors that may be highly correlated and possess a very low signal-to-noise ratio. When the number of predictors approaches the number of observations, the unrestricted OLS approach begins overfitting noise rather than extracting signal. This problem becomes worse in a low-signal-to-noise environment. In this case, the ols_full model is expected to become inefficient or even inconsistent.

We use this model specification as our subordinate benchmark in order to emphasize how well sophisticated forecast models can identify and incorporate relevant predictors to improve the predictive performance and to determine which firm characteristics, interactions, and nonlinear effects matter. To overcome the overfitting problem of highly parameterized OLS models and reduce the number of predictors, machine learning algorithms impose variable selection/shrinkage, dimension reduction, and other adjustments.

As already explained, we restrict our sample to firms with a minimum market capitalization of €25 mill. Nevertheless, our results still tend to be dominated by the large number of smaller firms. This is in line with Fama and French (2008), who find that the smallest 20% of stocks comprise only three percent of aggregate market capitalization. These smallest firms are often economically inconsequential to large institutional investors, i.e., such market participants cannot invest enough money as passive shareholders, or face high transaction costs for large trades. To find a balance between the large number of small firms and the small number of very large firms, Grinold and Kahn (2000) propose a generalized least squares (GLS) regression setting. In particular, they suggest to weight each observation by the inverse of its estimated error variance, which they proxy by the square root of its market capitalization. Gu et al. (2020) also suggest this adjustment to improve the prediction efficiency. Therefore, in a robustness test (unreported), we apply a weighted least squares loss function for both the ols_pars and ols_full model specifications to achieve more robust predictions. We use weights that are proportional to the square root of stock i's market capitalization at time t. However, because we do not observe improvements in the predictive performance relative to the two OLS-based models, we use the simpler ols_pars and ols_full models as our benchmarks, emphasizing that they are more conservative representatives for this forecast model family.

Penalized least squares

The most common machine learning tool to achieve variable selection/shrinkage in a high-dimensional regression setting is the penalized least squares approach. In particular, it identifies which predictors are informative and omits those that are not. This approach modifies the least squares loss function by adding a penalty term, denoted as $\Phi(\theta)$, to prefer more parsimonious model specifications, i.e., $l(\theta) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(r_{i,t+1} - g\left(z_{i,t};\theta\right)\right)^2 + \Phi(\theta)$.

There are two common types of penalties: First, the lasso approach penalizes the sum of absolute coefficients, thereby setting regression coefficients of a subset of predictors to exactly zero (variable selection). Second, the ridge approach penalizes the sum of squared regression coefficients, thereby only pushing coefficients close to zero (variable shrinkage). We follow the elastic net (elanet) approach, which combines the lasso and ridge methodologies.[9] It computes the weighted sum of both penalties to increase flexibility, i.e., $\Phi(\theta) = \lambda(1-p)\sum_{j=1}^{P}\left|\theta_j\right| + \lambda p\sum_{j=1}^{P}\left(\theta_j\right)^2$. Tuning parameters in this forecast model are $\lambda \in (0,1)$ and $p \in (0,1)$. λ indicates the general strength of the penalty (in particular, how strongly the regression coefficients are forced to zero); p denotes the relative weights on the lasso and the ridge approach. $p = 0$ corresponds to lasso; and $p = 1$ corresponds to ridge.

Principal component regressions and partial least squares

Although the elanet approach is capable of reducing a highly parameterized model's complexity by forcing coefficients close to or exactly to zero, it might perform badly if predictors are highly correlated. In this case, where predictors face low signal-to-noise ratios, a superior approach is to create new, de-correlated predictors as linear combinations of highly correlated variables (e.g., by averaging their values) instead of just omitting some. This reduces noise and thus increases the signal-to-noise ratio.

The basic concept of dimension reduction is variable averaging, as opposed to variable selection/shrinkage. Two common methods are principal component regression (pcr) and partial least squares (pls). Both follow a two-step procedure: First, the models conduct dimension reduction by creating new, de-correlated predictors (so-called components). Second, they identify the optimal number of components and then use them in a standard predictive regression model to estimate expected excess returns. The pcr approach forms principal components by incorporating the covariance matrix of the predictors. However, during the dimension reduction step, it does not take into account how the predictors relate to subsequent excess returns.

To overcome this deficiency, the pls approach keeps the forecast objective in mind, even during the dimension reduction step. To form the first component, it runs a univariate OLS regression of realized excess returns on each predictor separately. We can consider the resulting coefficients as reflecting "partial" sensitivity of the realized excess returns to each predictor. It then computes the weighted average of all predictors, using weights proportional to the absolute value of each coefficient. Higher weights are assigned to stronger univariate predictors (with higher absolute coefficient value), and vice versa.

To form further components, during each repetition, the realized excess return and the predictors are orthogonalized with respect to previously formed component(s). The tuning parameter in both forecast models is the number of components $number_{comp}$ included in the predictive regression. Using only a certain number of components can be considered equivalent to setting the coefficients for the other components (with low signal-to-noise ratios) to exactly zero.

Random forests and gradient boosted regression trees

Regression trees incorporate multi-way interactions and nonlinearity inherently, without having to add these effects as new predictors. The idea behind regression trees is that they adaptively split the dataset into groups of observations that behave in a similar manner. They follow an iteration process that is inspired by the growing behavior of real trees in nature: First, the process begins with one initial node, the root, in order to find the optimal split variable and the optimal split value by minimizing the MSFE within each partition. This results in two nodes with minimized impurity. Second, to further disentangle the dataset, it determines optimal split variables and values on the subsamples left over from the preceding step(s) to iteratively grow the regression tree. This results in multiple final nodes with minimized impurity, the leaves. The predicted excess return for each leaf reflects the simple average of the realized excess returns of the firms sorted into this leaf. Regression trees are invariant to monotonic transformations of predictors, able to incorporate categorical and numerical data in the same forecast models, and designed to capture interactions and nonlinearity. However, they are prone to overfitting and must be strongly regularized. To accomplish this, the ensemble forecast approach aggregates forecasts from many different regression trees into a single one. According to Gu et al. (2020), there are two common methods: bagging and boosting.

Random forests (rf) modify Breiman's (2001) traditional bagging approach. The idea is to draw multiple bootstrap samples of the original dataset, fit deep trees independently, and then average their predictions into an ensemble forecast, creating a single strong learner. Because dominant predictors are always more likely to become split variables at low levels, which can lead to large correlations between bootstrap-replicated trees, random forests apply the so-called dropout method. At each potential branch, they randomly drop out predictors, leaving only a subsample of predictors to be selected as potential split variables. The tuning parameters in this forecast model are the depth of trees L, the number of predictors M randomly considered as potential split variables, and the number of trees B added to the ensemble prediction.

[9] We also test the lasso and ridge approaches separately. Because we find no improvement in predictive performance relative to the elastic net approach, we do not present the results for these penalty functions here.

In contrast, gradient boosted regression trees (gbrt) follow the boosting approach, which is based on the idea that combining multiple shallow trees creates a single strong learner, stronger even than a single deep tree. The iterative procedure is as follows: The gbrt approach computes a first shallow tree to fit the realized excess returns. This oversimplified tree exhibits a high forecast error. Next, it computes a second shallow tree, fitting the forecast residuals from the first tree. The forecasts from these two trees are then added together to form an ensemble prediction. The forecast component from the second tree is shrunk by a factor $\nu \in (0, 1)$ to avoid overfitting the forecast residuals. Each additional shallow tree is fitted to the forecast residual from the preceding ensemble prediction, and its shrunk forecast component is added according to the ensemble forecast. The tuning parameters in this forecast model are the depth of the trees L, the shrinkage weight ν, and the number of trees B added to the ensemble prediction.

Neural networks

Neural networks (nn) are the most complex method in our empirical analysis. They are highly parameterized, which makes them suitable for solving very complicated machine learning problems. But they are opaque and can be difficult to interpret. In general, they map inputs (predictors) to outputs (realized excess returns). Inspired by the functioning of the human brain, they are composed of many interconnected computational units, called "neurons". Each neuron on its own provides very little predictive power, but a network of multiple neurons functions cohesively and improves the predictive performance. We use feedforward neural networks, where each node is connected to all the nodes in the previous layer and the connections follow a one-way direction, from input to output layer. The input layer contains the predictor variables (e.g., lagged firm characteristics), while the output layer contains a prediction for the dependent variable (realized excess returns). The simplest neural network (without any hidden layer) equals the OLS regression model. Adding hidden layers leads from shallow to deep architectures and is able to capture interactions and nonlinear effects (see Appendix 2, Fig. 6, Panel A).

Neural networks predict the output y as the weighted average of inputs x. In the simplest model, the OLS regression coefficients are taken as weights. In more complex architectures, the weights must be computed iteratively by using the "backpropagation" algorithm. As an initialization, this algorithm assigns random weights to each connection. It also calculates the initial MSFE based on the predictions derived from the inputs of the (last) hidden layer. It then proceeds iteratively as follows: First, it recursively (from output to input layer) computes the gradient of the MSFE with regard to the weights. Second, it adjusts the weights slightly in the *opposite* direction of the computed gradients, because the objective is to *minimize* the MSFE. Third, based on the adjusted weights, it re-calculates the MSFE.

The iteration process, known as "gradient descent," stops when the MSFE is ultimately minimized. Thus far, it is assumed that each node in the hidden layer creates a signal (e.g., it is incorporated into the computation of the weighted average). In the human brain, however, neural networks work somewhat differently. To avoid noise, a specific node transforms each of the preceding signals it transmits (if at all). For example, it may amplify or condense the preceding signals, or only create a signal if the accumulated information is sufficient. At each node, the weighted average of the preceding signals (either from the input or the preceding layer) is used as input x in an activation function (see Appendix 2, Fig. 6, Panel B). Following Gu et al. (2020), we choose the rectified linear unit (ReLU) activation function and apply it to each node in the hidden layers. To encourage sparsity in the number of active neurons, it only provides a signal as an output if the information from the preceding layer accumulates beyond a threshold: $\mathrm{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{otherwise.} \end{cases}$

In our analysis, we consider neural networks with up to five hidden layers (HLs) and up to thirty-two neurons (N), which we choose according to the geometric pyramid rule (Masters 1993).[10] In addition to a ReLU activation and a lasso-based penalization of the weights, we simultaneously apply other types of regularization to ensure computational feasibility and avoid overfitting (Gu et al. 2020).

First, we use the *stochastic gradient descent* (SGD) approach to train the neural networks. During the iteration process, the algorithm cuts the training sample into small random subsamples, so-called batches, and uses one at each iteration. This leads to strong improvements in computational speed. The algorithm still sees the entire training sample (consecutively, not contemporaneously, and at least once but usually multiple times), which helps incorporate all available information and avoids impairing the predictive performance. Consequently, the number of iterations depends on the size of the batches and the number of epochs (i.e., the number of times the algorithm sees the entire training sample).

Second, we adopt the *batch normalization* algorithm introduced by Ioffe and Szegedy (2015). It mitigates the internal covariate shift that occurs as the distribution of each hidden layer's inputs changes during the training (as the parameters of the preceding layers change) and slows down the learning process. To this end, within each batch, it cross-sectionally normalizes the input to each hidden layer.

[10] The neural network architectures are: nn_1 ($HL = 1; N = \{32\}$), nn_2 ($HL = 2; N = \{32, 16\}$), nn_3 ($HL = 3; N = \{32, 16, 8\}$), nn_4 ($HL = 4; N = \{32, 16, 8, 4\}$), and nn_5 ($HL = 5; N = \{32, 16, 8, 4, 2\}$).

Third, we apply *learning rate shrinkage* (see Appendix 2, Fig. 6, Panel C). The learning rate determines the size of the incremental steps in the gradient, while iteratively minimizing the MSFE. It faces a trade-off between finding the global minimum instead of the local counterpart (smaller learning rate) and computational speed (larger learning rate). This regularization procedure begins with a larger learning rate to speed up computation. As the gradient approaches zero, it shrinks the learning rate towards zero to overcome a potential local minimum.

Fourth, we implement *early stopping* (see Appendix 2, Fig. 6, Panel D), as neural networks aim to minimize the MSFE in the training sample. This regularization terminates the SGD iteration process when the MSFE in the validation sample increases for a pre-specified number of iterations (so-called patience), which also speeds up computation.

Fifth, we adopt the *ensemble* approach proposed by Hansen and Salamon (1990) and Dietterich (2000). We compute ten neural networks from the same specification family at each re-estimation date, using independent seeds.[11] We then average over the predictions to increase the signal-to-noise ratio, since the stochastic nature of the SGD approach leads to different forecasts for different seeds.

Lastly, in addition to the regularization applied by Gu et al. (2020), we employ the *dropout* method (similar to that applied for regression trees). It randomly sets a fraction rate of input variables to exactly zero at each iteration, and thus is one of the most effective methods in the neural network framework to prevent overfitting.

Neural networks are computationally intensive and can be specified in an innumerable amount of different architectures. This is why we retreat from tuning parameters (e.g., the size of batches or the number of epochs) and instead pre-specify five different models. We assume that our nn_1 to nn_5 architectures serve a conservative lower bound for the predictive performance of neural network models in general.

Empirical results

In-sample tests

Model complexity

Since we re-estimate each forecast model on an annual basis, it is interesting to gauge whether model complexity changes over time or rather remains stable. We only consider models that can actually exhibit time-varying model complexity in our setting (elanet, pcr, pls, rf, and gbrt). These models pertain to different families, so there is no uniform model complexity measure. For the elanet approach, which performs variable selection/shrinkage, we use the number of nonzero regression coefficients. The pcr and pls approaches conduct dimension reduction, so we consider the optimal number of components included in the predictive regression. But, because the rf and gbrt approaches are nonparametric and tree-based, we take the optimal depth (for rf) and the number of unique split variables (for gbrt) of the trees to measure model complexity.[12]

Figure 1 presents the model complexity of each forecast model at each re-estimation date. In line with the results from Gu et al. (2020), we observe that each model's complexity varies substantially over time. The time-varying behavior of different approaches within the same forecast model family (namely pcr/pls and rf/gbrt) is similar with regard to low- and high-complexity periods.

Variable importance

Next, because the different forecast models behave similarly in terms of model complexity (at least within each model family), it is further instructive to explore whether the different approaches denote different predictors as most relevant for estimating subsequent excess returns. To this end, we calculate each model's variable importance matrix based on a two-step approach, separately for each re-estimation date: First, we compute the absolute variable importance as the reduction in R^2 from setting all values of a given predictor to zero within the training sample.[13] Second, we normalize the absolute variable importance measures to sum to 1, signaling the relative contribution of each variable to a model. Figure 2 depicts the time series average of relative variable importance measures for each forecast model separately. We find that, on average, the various forecast models denote similar variables as the most informative (e.g., *earningstoprice* or *ret_2_12*). However, some models focus on a highly concentrated set of predictors. For example, tree-based models (rf and gbrt) put most of their weights on only a few variables, whereas the dimension-reducing approaches (pcr and pls) consider a much larger range of predictors to be important.

[11] Seeds are numbers used to initialize random processes. This approach ensures different but reproducible predictions.

[12] Neural networks can exhibit time-varying model complexity. However, because we pre-specify the architectures and only present the results for five specifications (nn_1–nn_5), we do not allow for time variation in model complexity.

[13] In contrast to Gu et al. (2020), we not only set the value of the specific predictor itself to zero, but we also set each interaction or nonlinearity term that contains information from the respective variable to zero. We assume that each respective predictor is completely, not only partly, uninformative, which allows for more realistic variable importance metrics.

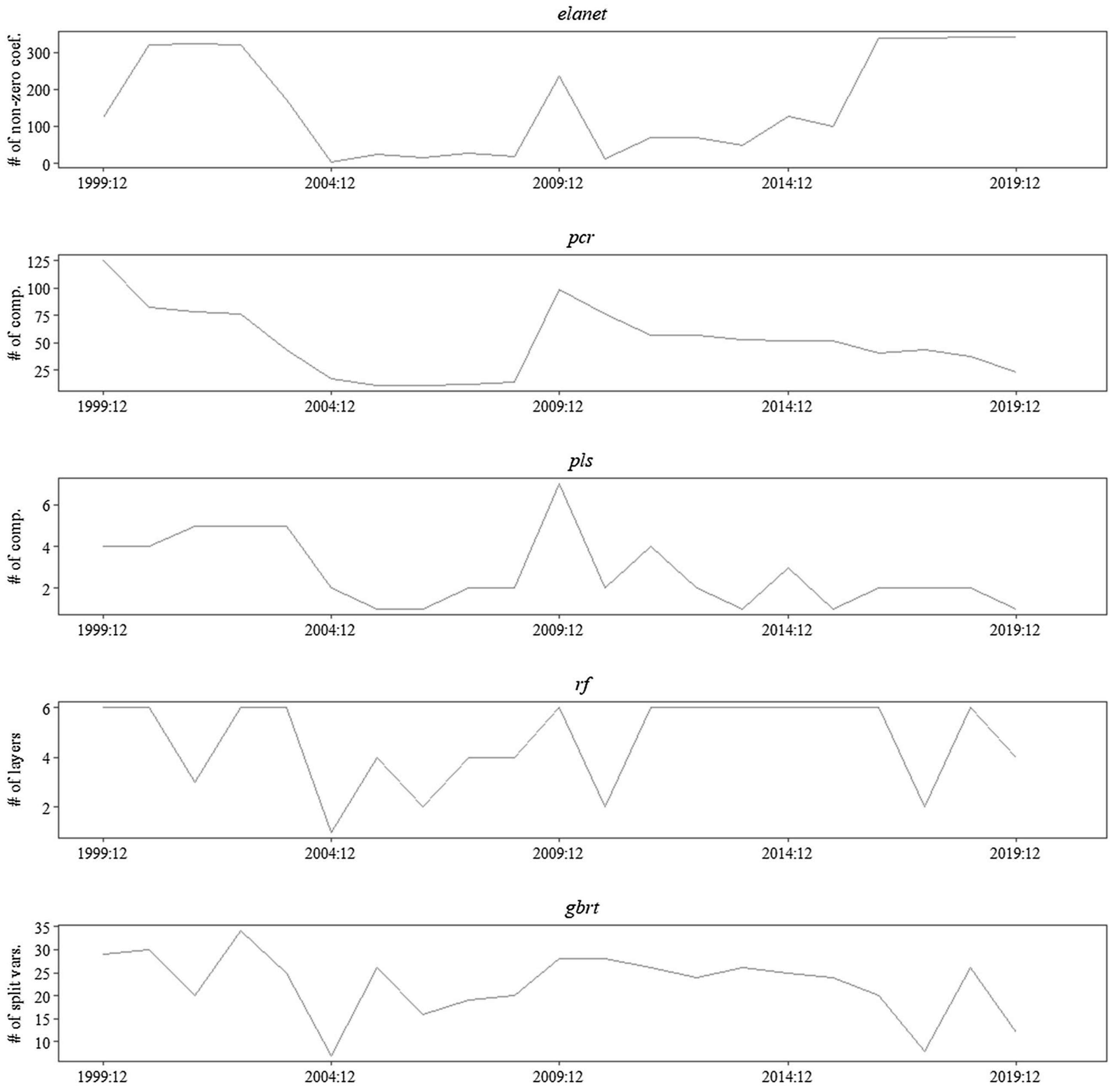


Fig. 1 Between-models comparison of model complexity over time. This figure presents the model complexity of selected machine learning models (elanet, pcr, pls, rf, and gbrt) introduced in the "Forecast models" section, at each re-estimation date. As the models pertain to different families, the model complexity measure varies: the number of nonzero regression coefficients (for elanet), the optimal number of components included in the predictive regression (for pcr and pls), and the optimal depth (for rf) or the number of unique split variables (for gbrt) of the regression trees. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end

To enhance between-models comparability, we rank the average relative contribution of each variable within a specific model and sum the ranks across all models to obtain an overall rank (higher variable importance = higher rank). Figure 3 presents a heat map of relative variable importance ranks. The rows are sorted in descending order based on overall rank. Therefore, the higher a variable is placed, the more important it is overall. Darker cell colors denote greater importance for the respective variable to a model. Again, we find commonalities across relative variable importance

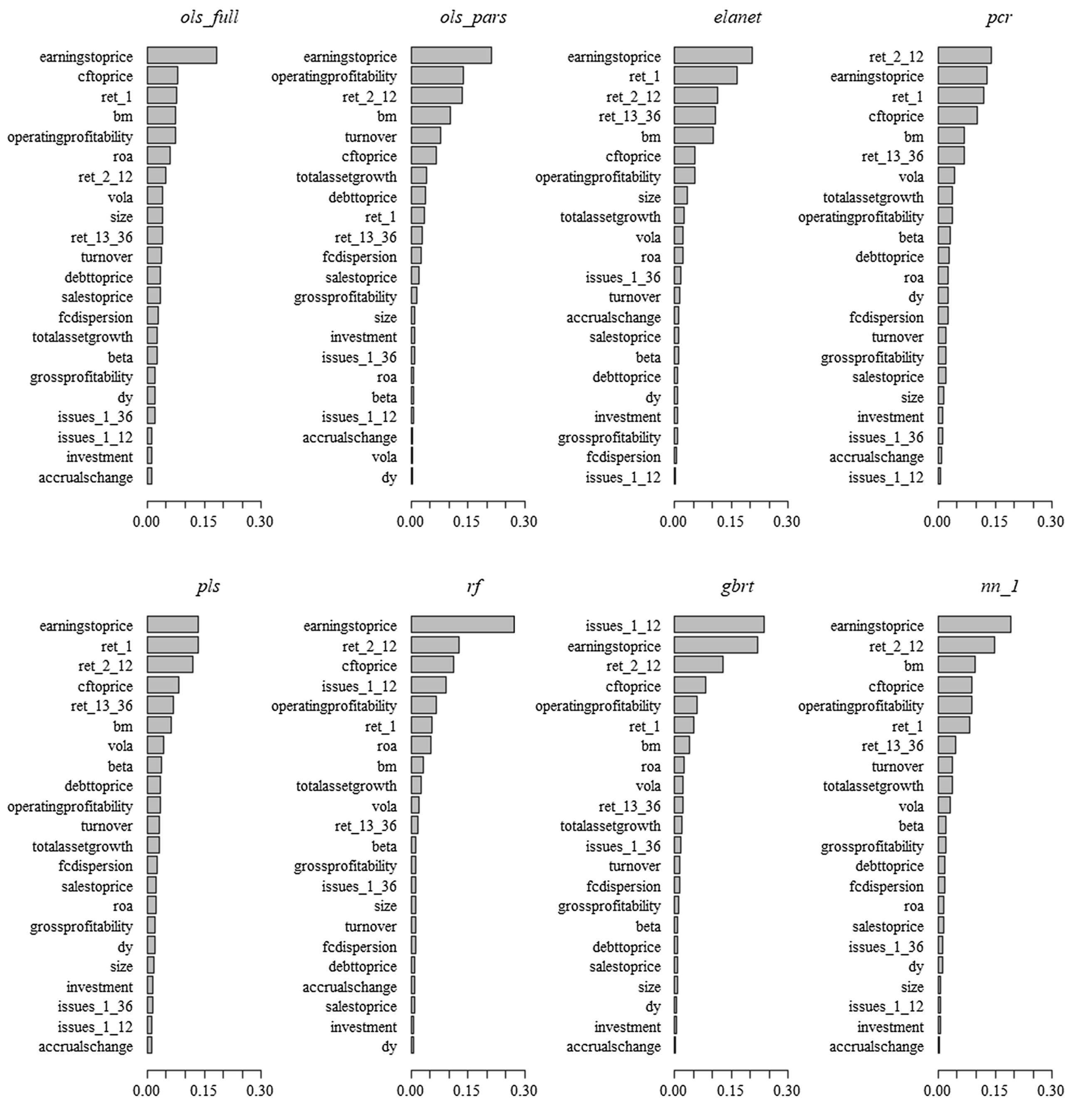


Fig. 2 Model-specific relative variable importance. This figure depicts the time series average of relative variable importance measures of each machine learning model introduced in the "Forecast models" section, which are calculated based on a two-step approach: First, the absolute variable importance is computed as the reduction in R^2 from setting all values of a given predictor to zero within the training sample. Second, the absolute variable importance measures are normalized to sum to 1, signaling the relative contribution of each variable to a model. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end

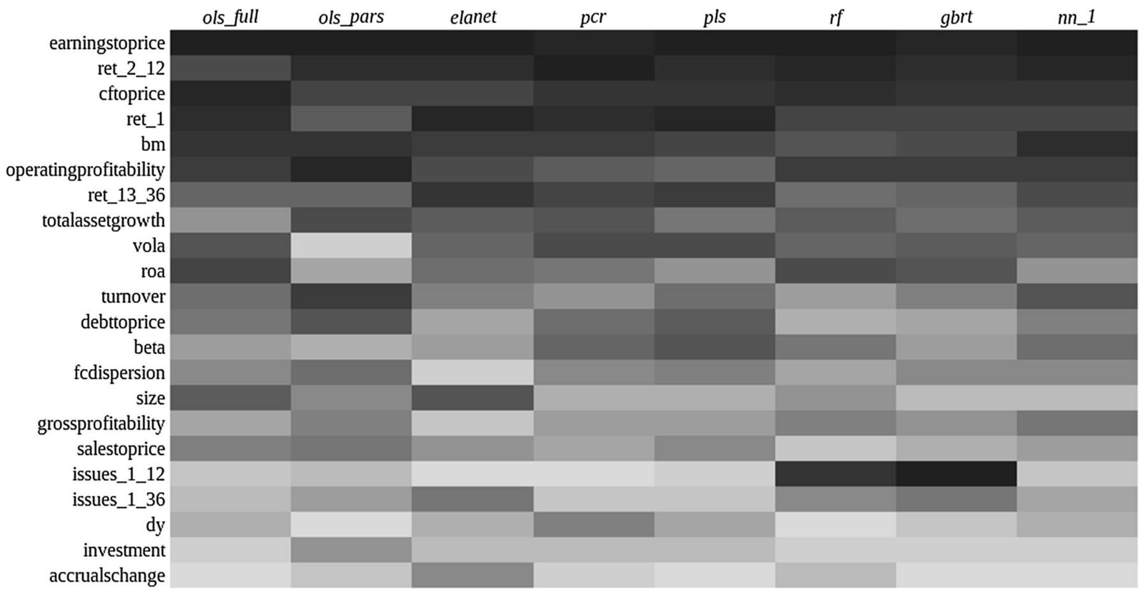


Fig. 3 Between-models comparison of relative variable importance. This figure presents a heat map of average relative variable importance ranks of each machine learning model introduced in the "Forecast models" section, which are obtained by ranking the relative contribution of each variable within a specific model, and summing the ranks across all models to obtain an overall rank (higher variable importance = higher rank). The rows are sorted in descending order based on overall rank. Darker cell colors denote greater importance for the respective variable to a model. The relative variable importance metrics are calculated based on a two-step approach: First, the absolute variable importance is computed as the reduction in R^2 from setting all values of a given predictor to zero within the training sample. Second, the absolute variable importance measures are normalized to sum to 1, signaling the relative contribution of each variable to a model. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end

metrics, which are even more pronounced within the same forecast model family. For example, both tree-based methods (rf and gbrt) find *issues_1_12* strongly informative, while all other forecast models assume this particular predictor is relatively unimportant.

Finally, we determine whether the relative importance of different variables within a specific model changes over time or rather remains stable. Volatile metrics would indicate that all covariates in the predictor sets are essential; stable figures would suggest removing uninformative predictors permanently, as they can degrade a forecast's signal-to-noise ratio. To identify the predictors' time variability in relative importance measures, we rank the relative contribution of each variable within a specific model at each re-estimation date (higher variable importance = higher rank). The trends are similar for all forecast models. Therefore, for the sake of brevity, we only present and discuss the results for our benchmark model (ols_pars). Figure 4 shows the relative variable importance ranks of the ols_pars model at each re-estimation date. Although a few variables tend to stay close to the upper or lower bound, the graph indicates that the relative variable importance metrics change substantially over time. It does *not* recommend removing specific predictors.[14]

Out-of-sample tests

Statistical predictive performance

Since the forecast models behave similarly from a model complexity and relative variable importance perspective, we aim to determine whether they differ in statistical predictive performance. We therefore conduct three tests: First, in Panel A of Table 2, we follow Lewellen (2015) and Drobetz et al. (2019) and calculate out-of-sample predictive slopes. These allow us to assess the precision and accuracy of the excess return predictions (regardless of any naïve benchmark). Second, in Panel B of Table 2, we follow Gu et al.

[14] To be conservative, we compare the statistical and economic predictive performance of the original ols_pars model with versions that only consider the top five or ten predictors in terms of relative variable importance. Out-of-sample tests (unreported) are identical to the tests shown in the "Out-of-sample tests" section. We find that neither the ols_top5 nor the ols_top10 model exhibits substantial outperformance in any of the tests, so we retract the idea of removing uninformative variables from the predictor sets and instead consider each predictor as informative (to varying degrees). Additionally, we caution that the pre-estimation variable selection based on relative importance metrics derived from the entire sample period could lead to foresight bias, undermining the credibility of any out-of-sample tests.

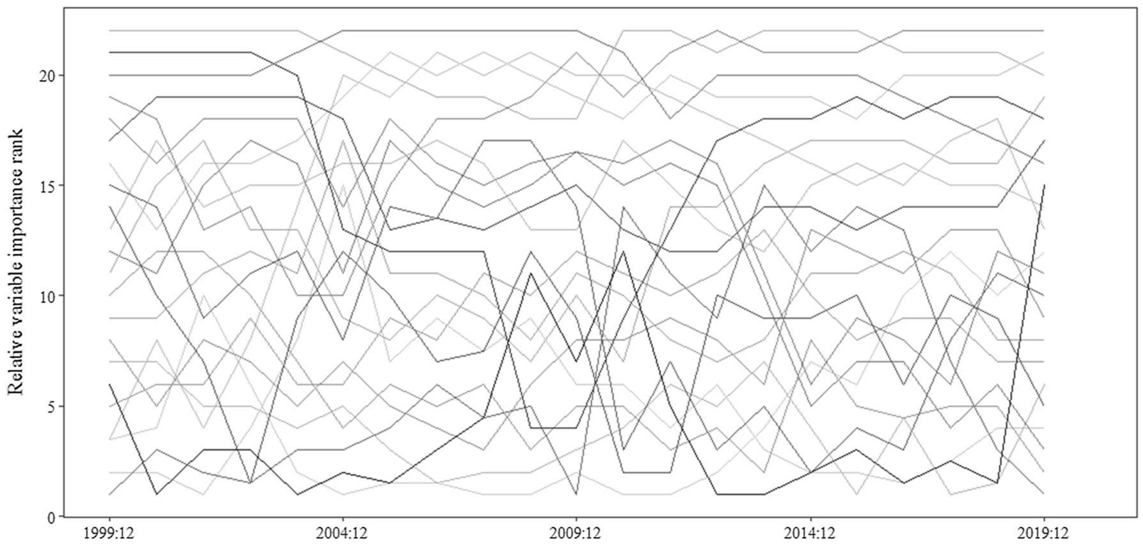


Fig. 4 Relative variable importance over time. This figure shows the relative variable importance ranks of an exemplary selected machine learning model (ols_pars) introduced in the "Forecast models" section, at each re-estimation date, which are obtained by ranking the relative contribution of each variable within a specific model (higher variable importance = higher rank). The relative variable importance metrics are calculated based on a two-step approach: First, the absolute variable importance is computed as the reduction in R^2 from setting all values of a given predictor to zero within the training sample. Second, the absolute variable importance measures are normalized to sum to 1, signaling the relative contribution of each variable to a model. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end

(2020) and calculate out-of-sample predictive R^2 metrics relative to a naïve excess return forecast of zero. We compute the metrics in Table 2 for each model and three subsamples: the full sample, and two subsamples that only contain large or small firms (based on the market capitalization). Third, in Table 3, we assess the relative predictive performance of each model in a pairwise comparison. We conduct a modified version of the Diebold and Mariano (1995) test (DM test) to gauge the differences in out-of-sample predictive power between two models.

Panel A of Table 2 presents the time series averages of out-of-sample predictive slopes (PS^2_{oos}) calculated from the test sample at each re-estimation date. These are derived from pooled regressions of realized excess returns on the corresponding estimates from a specific model, i.e., $PS_{oos} = \frac{Cov(R_{real}, R_{pred})}{Var(R_{pred})}$. The predictive slopes are close to 1 for the machine learning models, indicating that forecast dispersion primarily reflects cross-sectional variation in true expected excess returns.[15] They are much smaller for the OLS-based models and even more pronounced for the full predictor set, which is due to the overfitting that can arise because it contains too many predictors with a low signal-to-noise ratio.

Importantly, the predictive slopes for the neural network models deteriorate quickly in the number of hidden layers, which is in line with Gu et al.'s (2020) findings and might be due to overfitting. Deeper neural networks seem to be too complex for the relatively small dataset and parsimonious set of predictors, and/or the monthly excess return setting. Since this pattern holds for other measures of statistical and economic predictive performance, we only present and discuss the results for the simplest nn_1 architecture going forward.

Considering the full sample, the predictive slopes are closest to 1 for the tree-based models (1.03 and 1.07 for rf and gbrt) and the neural network architecture (0.98 for nn_1), followed by the dimension-reducing models (0.74 and 0.76 for pcr and pls). Note that selecting predictors manually (ols_pars with 0.65) is inferior to using a rule-based selecting/shrinking technique (elanet with 0.92), while incorporating the full set of 341 predictors performs the worst (0.45 for ols_full). Although the patterns are similar for the two market capitalization-based subsamples, the magnitudes of the predictive slopes are substantially higher for the "small firms" subsample. There are two likely reasons for this: First, the models are estimated from the full sample, so differences in cross-sectional variation in true

[15] PS_{oos} larger than 1 indicate overly narrow forecasts, while PS_{oos} smaller than 1 indicate overly wide predictions.

Table 2 Predictive slopes and predictive R^2s

	ols_full	ols_pars	elanet	pcr	pls	rf	gbrt	nn_1
Panel A: Predictive slopes								
Full sample	0.45	0.65	0.92	0.74	0.76	1.03	1.07	0.98
Large firms	0.30	0.46	0.67	0.56	0.58	0.77	0.82	0.71
Small firms	0.53	0.79	1.05	0.84	0.87	1.18	1.23	1.15
Panel B: Predictive R^2s [%]								
Full sample	− 0.52	0.57	1.21	1.24	1.08	1.18	1.14	1.23
Large firms	− 1.16	0.11	0.75	0.85	0.74	0.92	0.74	0.93
Small firms	− 0.10	0.90	1.55	1.54	1.37	1.40	1.46	1.47

This table presents predictive performance metrics of each machine learning model introduced in the "Forecast models" section. Panel A reports the time series averages of out-of-sample predictive slopes ($\mathrm{PS}^2_{\mathrm{oos}}$) calculated from the test sample at each re-estimation date. These are derived from pooled regressions of realized excess returns on the corresponding estimates from a specific model, i.e., $\mathrm{PS}_{\mathrm{oos}} = \frac{\mathrm{Cov}(R_{\mathrm{real}}, R_{\mathrm{pred}})}{\mathrm{Var}(R_{\mathrm{pred}})}$. Panel B reports the time series averages of predictive R^2 s (R^2_{oos}) calculated from the test sample at each re-estimation date. These are derived by comparing a model's R^2_{oos} with the numbers of the naïve excess return forecast of zero, i.e., $R^2_{\mathrm{oos}} = 1 - \frac{\mathrm{SSE}(R_{\mathrm{real}}, R_{\mathrm{pred}})}{\mathrm{SSE}(R_{\mathrm{real}}, 0)}$. The numbers are presented for three subsamples: the full sample, and two subsamples that only contain large or small firms (based on the market capitalization). The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end

Table 3 Between-model comparison of predictive performance

	ols_full	ols_pars	elanet	pcr	pls	rf	gbrt	nn_1
ols_full		**2.29**	**4.12***	**4.69***	**4.46***	**3.75***	**3.70***	**2.50***
ols_pars			**3.35***	**3.32***	**2.86***	**3.01***	**2.90***	1.55
elanet				1.03	0.06	−0.56	−0.82	−0.49
pcr					−1.23	−1.47	−1.41	−0.99
pls						−0.44	−0.55	−0.57
rf							−0.33	−0.27
gbrt								−0.19
nn_1								

This table reports Diebold and Mariano (1995) test (DM test) statistics for the pairwise comparison of the predictive performance of each machine learning model introduced in the "Forecast models" section. The modified version of the DM test compares the forecasts models' monthly MSFEs, i.e., $\mathrm{MSFE}_{t+1} = \frac{1}{N_{t+1}} \sum_{i=1}^{N_{t+1}} (\hat{e}_{i,t+1})^2$. The DM test statistic for comparing column model j and row model k is $\mathrm{DM}_{kj} = \frac{\bar{d}_{kj}}{\hat{\sigma}_{\bar{d}_{kj}}}$, where $d_{kj,t+1} = \mathrm{MSFE}^{(k)}_{t+1} - \mathrm{MSFE}^{(j)}_{t+1}$ is the difference in MSFEs, $\bar{d}_{kj} = \frac{1}{T} d_{kj,t+1}$ is the time series average of these differences, and $\hat{\sigma}_{\bar{d}_{kj}}$ is the Newey and West (1987) standard error of $d_{kj,t+1}$ (with four lags to account for possible heteroscedasticity and autocorrelation). The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end

expected excess returns between the subsamples could explain differences in predictive slopes. Second, machine learning algorithms could forecast the excess returns of small firms indeed better.

Panel B of Table 2 presents the time series averages of predictive R^2s (R^2_{oos}) calculated from the test sample at each re-estimation date. These are derived by comparing a model's R^2_{oos} with the numbers of the naïve excess return forecast of zero, i.e., $R^2_{\mathrm{oos}} = 1 - \frac{\mathrm{SSE}(R_{\mathrm{real}}, R_{\mathrm{pred}})}{\mathrm{SSE}(R_{\mathrm{real}}, 0)}$.[16] Considering the full

[16] As a naïve benchmark, the historical average of excess returns is widely used, especially because of its performance at a portfolio level. However, many studies, including Gu et al. (2020), suggest that

sample, our results are in line with Gu et al. (2020) and emphasize our prior findings. In particular, the ols_full model exhibits the worst predictive power, a negative R^2_{oos} metric (− 0.52%), while the ols_pars model avoids overfitting, resulting in a substantially larger number (0.57%). Each of the remaining machine learning models beats by far the ols_pars benchmark in the sense of higher R^2_{oos} metrics. Restricting the complexity of the ols_full model by variable selection/shrinkage (1.21% for elanet) or dimension reduction (1.24% and 1.08% for pcr and pls, respectively), or by adding interactions and nonlinearity to the ols_pars model with a tree-based approach (1.18% and 1.14% for rf and gbrt, respectively) or neural network architecture (1.23% for nn_1) boosts the predictive performance. Again, the patterns remain qualitatively the same for the two market capitalization-based subsamples, with substantially higher R^2_{oos} metrics for the "small firms" subsample.

Table 3 presents DM test statistics to enable a pairwise comparison of predictive performance. We again follow Gu et al. (2020) and apply a modified version of the DM test to compare the forecasts models' monthly MSFEs, i.e., $\text{MSFE}_{t+1} = \frac{1}{N_{t+1}} \sum_{i=1}^{N_{t+1}} (\hat{e}_{i,t+1})^2$.[17] The DM test statistic for comparing column model j and row model k is $\text{DM}_{kj} = \frac{\overline{d}_{kj}}{\hat{\sigma}_{\overline{d}_{kj}}}$, where $d_{kj,t+1} = \text{MSFE}^{(k)}_{t+1} - \text{MSFE}^{(j)}_{t+1}$ is the difference in MSFEs, $\overline{d}_{kj} = \frac{1}{T} d_{kj,t+1}$ is the time series average of these differences, and $\hat{\sigma}_{\overline{d}_{kj}}$ is the Newey and West (1987) standard error of $d_{kj,t+1}$ (with four lags to account for possible heteroscedasticity and autocorrelation). We follow the convention that positive signs of DM_{kj} indicate superior predictive performance of column model j, i.e., that it yields, on average, lower forecast errors than row model i. As DM test statistics are asymptotically $N(0, 1)$-distributed and test the null hypothesis that the divergence between two forecast models is zero, they map to p-values in the same way as regression t-statistics.

We interpret the DM test statistics derived from the pairwise comparison in two different ways: First, we consider them separately, without taking into account that other pairwise comparisons are conducted simultaneously. For a significance level of 5%, the threshold for the test statistics to indicate significance is 1.96. We denote single-comparison significance with boldface. Second, we address the multiple comparison issue by using the Bonferroni correction, which divides the 5% significance level by the number of simultaneously conducted comparisons. Comparing eight models, this adjustment increases the significance threshold to 2.50. We denote multiple-comparison significance with an asterisk.

We observe that our results underline the tendencies found so far. Each machine learning model beats the OLS framework, which holds for both single- and multiple-comparison cases. One idiosyncrasy is the nn_1 model, which slightly misses significance relative to the ols_pars benchmark (1.55). Moreover, since the remaining statistics are insignificant, it is impossible to draw any further conclusions regarding the relative outperformance of the other forecast models.

Expected return-sorted portfolios

The various models exhibit similar behavior with regard to model complexity and relative VI, but have substantially different out-of-sample predictive power. Note that the tests conducted in the "Statistical predictive performance" section are statistical in nature, while Leitch and Tanner (1991) suggest that there may be only a weak association between statistical measures and economic profitability. Therefore, it is helpful to understand whether differences in statistical predictive performance translate into differences in predictive power from the economic perspective of a realistic trading strategy. At the end of each month, we first sort stocks into decile portfolios based on the respective excess return estimates. For each model and decile portfolio, we then calculate the equal- and value-weighted mean of ex ante predicted and ex post realized excess returns. Based on the realized excess returns, we also compute the standard deviation, annualized Sharpe ratio, and the t-statistic testing the null hypothesis that realized excess returns are zero. Table 4 presents the time series averages of monthly figures.

In general, the patterns are similar for the equal- and value-weighted decile portfolios, although they are substantially more pronounced for the equal-weighting scheme (in terms of higher Sharpe ratios and t-statistics). The average realized excess returns line up almost monotonically with average predicted excess returns, resulting in positive H–L spreads that are statistically significant and economically large. The H–L returns are, on average, more than 75% higher for the equal-weighted decile portfolios, indicating that a substantial portion of the excess returns is driven by small firms. However, all H–L spreads, using both the equal-weighting and the value-weighting scheme, suggest economic profitability. The results highlight that

Footnote 16 (continued)
the historical average is less applicable at a stock level. Due to large noise in single excess returns, historical averages typically underperform the naïve forecast of zero by a large margin. To set the highest possible bar for identifying incremental predictive performance, we opt to compare the R^2 s against a naïve excess return forecast of zero.

[17] The DM test usually compares forecast errors from two models at a stock level, requiring weak error dependence for asymptotic normality. While the TS return dependence is sufficiently weak in our setting, it is questionable whether the requirement of weak error dependence is fulfilled due to a potentially strong CS return dependence.

Table 4 Excess returns for expected return-sorted portfolios

	Equal-weighted					Value-weighted				
	Pred [%]	Real [%]	Std [%]	Shp	*t*-stat	Pred [%]	Real [%]	Std [%]	Shp	*t*-stat
ols_full										
Low (L)	−2.70	−1.00	5.46	−0.64	−2.16	−2.39	−0.65	5.88	−0.39	−1.43
2	−1.07	−0.27	4.56	−0.20	−0.69	−1.06	−0.10	5.21	−0.07	−0.25
3	−0.44	0.09	4.45	0.07	0.25	−0.43	0.01	4.82	0.01	0.03
4	0.01	0.42	4.36	0.34	1.21	0.01	0.29	4.52	0.22	0.91
5	0.38	0.49	4.50	0.38	1.35	0.38	0.33	4.72	0.24	0.96
6	0.74	0.75	4.42	0.59	2.21	0.74	0.46	4.78	0.33	1.47
7	1.11	0.87	4.60	0.65	2.50	1.11	0.70	4.89	0.50	2.30
8	1.54	0.96	4.77	0.70	2.65	1.54	0.56	5.18	0.37	1.60
9	2.10	1.14	5.05	0.78	2.92	2.09	0.57	5.15	0.38	1.56
High (H)	3.36	1.72	5.73	1.04	4.01	3.21	0.69	6.05	0.40	1.64
H–L	6.06	2.72	2.61	3.62	13.34	5.60	1.35	4.03	1.16	4.67
ols_pars										
Low (L)	−1.71	−1.09	5.32	−0.71	−2.43	−1.58	−0.80	5.55	−0.50	−1.82
2	−0.70	−0.27	4.87	−0.19	−0.67	−0.69	−0.13	5.07	−0.09	−0.38
3	−0.22	0.00	4.57	0.00	−0.01	−0.21	−0.07	4.98	−0.05	−0.17
4	0.15	0.27	4.60	0.20	0.73	0.16	0.19	4.66	0.14	0.58
5	0.47	0.49	4.53	0.37	1.39	0.47	0.33	4.70	0.24	1.04
6	0.77	0.69	4.51	0.53	1.94	0.77	0.59	4.88	0.42	1.72
7	1.06	0.82	4.69	0.60	2.31	1.06	0.48	5.01	0.33	1.42
8	1.37	1.07	4.64	0.80	3.01	1.37	0.59	5.08	0.40	1.71
9	1.77	1.37	4.95	0.96	3.62	1.76	0.51	5.41	0.32	1.35
High (H)	2.56	1.84	5.26	1.21	4.52	2.53	0.84	5.02	0.58	2.68
H–L	4.27	2.93	2.60	3.91	15.05	4.11	1.64	3.71	1.53	6.19
elanet										
Low (L)	−1.38	−1.17	6.08	−0.67	−2.32	−1.16	−0.92	7.43	−0.43	−1.55
2	−0.41	−0.30	4.92	−0.21	−0.75	−0.39	−0.27	5.58	−0.17	−0.65
3	0.00	0.05	4.51	0.04	0.14	0.01	−0.05	4.90	−0.04	−0.15
4	0.27	0.41	4.37	0.33	1.14	0.28	0.36	4.80	0.26	1.11
5	0.50	0.43	4.33	0.35	1.32	0.50	0.15	4.63	0.11	0.45
6	0.70	0.61	4.36	0.48	1.77	0.70	0.48	4.68	0.36	1.52
7	0.90	0.85	4.36	0.68	2.58	0.90	0.49	4.74	0.36	1.59
8	1.14	1.05	4.60	0.79	3.01	1.14	0.57	4.83	0.41	1.61
9	1.45	1.35	5.01	0.93	3.65	1.44	0.78	5.31	0.51	2.42
High (H)	2.10	1.90	5.48	1.20	4.44	2.03	0.92	5.60	0.57	2.31
H–L	3.48	3.07	2.47	4.31	16.20	3.19	1.84	4.66	1.37	5.41
pcr										
Low (L)	−1.82	−1.21	5.93	−0.71	−2.37	−1.57	−0.86	7.29	−0.41	−1.53
2	−0.60	−0.26	4.81	−0.19	−0.66	−0.58	−0.29	5.70	−0.17	−0.70
3	−0.11	0.03	4.59	0.02	0.07	−0.10	0.02	5.25	0.01	0.06
4	0.22	0.26	4.38	0.21	0.77	0.22	0.16	4.72	0.12	0.51
5	0.49	0.46	4.22	0.38	1.40	0.49	0.30	4.42	0.24	1.00
6	0.73	0.71	4.28	0.58	2.19	0.73	0.36	4.38	0.29	1.27
7	0.98	0.92	4.44	0.72	2.68	0.97	0.49	4.50	0.38	1.63
8	1.25	1.00	4.75	0.73	2.74	1.25	0.54	5.02	0.37	1.48
9	1.62	1.34	4.99	0.93	3.51	1.61	0.79	5.27	0.52	2.12
High (H)	2.44	1.94	5.51	1.22	4.45	2.33	1.12	5.65	0.69	2.94
H–L	4.26	3.15	2.58	4.23	13.86	3.90	1.98	5.09	1.35	5.32
Pls										
Low (L)	−1.81	−1.30	5.75	−0.78	−2.57	−1.52	−1.11	7.20	−0.53	−1.94
2	−0.54	−0.24	4.92	−0.17	−0.58	−0.52	−0.23	5.63	−0.14	−0.55
3	−0.03	0.05	4.64	0.04	0.14	−0.02	0.05	5.27	0.04	0.15
4	0.30	0.35	4.54	0.26	0.96	0.30	0.27	4.74	0.20	0.78

Table 4 (continued)

	Equal-weighted					Value-weighted				
	Pred [%]	Real [%]	Std [%]	Shp	*t*-stat	Pred [%]	Real [%]	Std [%]	Shp	*t*-stat
5	0.56	0.46	4.31	0.37	1.36	0.56	0.15	4.65	0.11	0.47
6	0.80	0.68	4.43	0.54	2.02	0.80	0.57	4.51	0.43	1.93
7	1.04	0.89	4.53	0.68	2.65	1.04	0.54	4.65	0.40	1.77
8	1.31	1.07	4.54	0.81	3.07	1.31	0.62	4.94	0.43	1.85
9	1.67	1.29	4.97	0.90	3.44	1.66	0.60	5.13	0.40	1.63
High (H)	2.47	1.93	5.35	1.25	4.60	2.37	0.90	5.57	0.56	2.26
H–L	4.28	3.23	2.45	4.56	14.02	3.89	2.01	4.46	1.56	5.69
rf										
Low (L)	− 1.19	− 1.15	6.38	− 0.63	− 2.18	− 1.01	− 0.62	7.54	− 0.28	− 1.10
2	− 0.28	− 0.33	5.11	− 0.23	− 0.83	− 0.26	− 0.40	5.85	− 0.24	− 0.99
3	0.10	0.07	4.64	0.05	0.20	0.11	0.00	5.03	0.00	0.00
4	0.35	0.33	4.45	0.25	0.93	0.35	0.30	4.61	0.23	0.97
5	0.53	0.47	4.25	0.38	1.40	0.53	0.39	4.54	0.30	1.21
6	0.68	0.66	4.13	0.55	2.05	0.68	0.58	4.44	0.46	2.01
7	0.86	0.85	4.30	0.69	2.57	0.86	0.54	4.62	0.41	1.83
8	1.07	1.02	4.59	0.77	2.90	1.07	0.58	4.84	0.41	1.73
9	1.30	1.35	4.93	0.95	3.46	1.29	0.67	5.21	0.44	1.74
High (H)	1.75	1.91	5.39	1.23	4.42	1.69	0.89	6.41	0.48	1.86
H–L	2.94	3.06	2.75	3.86	14.17	2.70	1.51	4.79	1.09	4.50
gbrt										
Low (L)	− 1.28	− 1.14	6.21	− 0.63	− 2.23	− 1.05	− 0.77	6.92	− 0.39	− 1.59
2	− 0.31	− 0.35	4.91	− 0.25	− 0.89	− 0.30	− 0.42	5.41	− 0.27	− 1.08
3	0.06	0.11	4.43	0.09	0.32	0.06	0.17	5.00	0.12	0.48
4	0.30	0.27	4.37	0.22	0.79	0.30	0.23	4.32	0.19	0.73
5	0.51	0.42	4.35	0.33	1.24	0.51	0.32	4.65	0.24	0.99
6	0.69	0.62	4.35	0.50	1.81	0.69	0.34	4.53	0.26	1.10
7	0.87	0.84	4.30	0.68	2.52	0.87	0.39	4.76	0.29	1.14
8	1.08	1.10	4.61	0.83	3.02	1.08	0.56	5.02	0.39	1.72
9	1.35	1.36	5.09	0.93	3.41	1.34	0.94	5.30	0.61	2.69
High (H)	1.94	1.94	5.43	1.24	4.62	1.85	0.85	6.00	0.49	1.96
H–L	3.22	3.08	2.70	3.95	13.86	2.91	1.62	4.22	1.33	5.37
nn_1										
Low (L)	− 1.21	− 1.25	6.08	− 0.71	− 2.50	− 1.09	− 0.94	7.11	− 0.46	− 1.79
2	− 0.36	− 0.27	5.06	− 0.18	− 0.65	− 0.35	− 0.25	5.36	− 0.16	− 0.64
3	0.04	− 0.02	4.49	− 0.01	− 0.05	0.05	− 0.19	5.14	− 0.13	− 0.52
4	0.32	0.34	4.58	0.26	0.94	0.33	0.26	4.72	0.19	0.83
5	0.56	0.44	4.47	0.34	1.23	0.56	0.47	4.72	0.34	1.36
6	0.76	0.63	4.37	0.50	1.85	0.76	0.38	4.70	0.28	1.17
7	0.96	0.85	4.43	0.67	2.50	0.96	0.54	4.50	0.42	1.79
8	1.17	1.06	4.50	0.82	3.07	1.17	0.56	4.71	0.41	1.72
9	1.43	1.39	4.89	0.98	3.71	1.43	0.69	5.18	0.46	1.96
High (H)	1.92	2.01	5.14	1.35	4.89	1.87	0.99	5.34	0.64	2.77
H–L	3.13	3.26	2.91	3.89	13.96	2.96	1.94	4.76	1.41	5.89

This table presents the time series averages of predicted and realized excess returns of decile portfolios based on the expected excess returns obtained from each machine learning model introduced in the "Forecast models" section. In addition, based on the realized excess returns, the standard deviation and the annualized Sharpe ratio are reported. The *t*-statistics, testing the null hypothesis that realized excess returns are zero, are calculated using the monthly point estimates, incorporating a Newey and West (1987) correction with four lags to account for possible heteroscedasticity and autocorrelation in the realized excess returns. The numbers are presented for both equal- and value-weighted decile portfolios. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end all presented approaches capture cross-sectional variation in realized excess returns, but the potential for providing profitable trading signals differs substantially across them.

We compare the equal-weighted H–L returns, which tend to be driven more by the long side (decile 10) than the short side (decile 1). The values indicate that differences in statistical predictive performance translate at least partly into differences in economic profitability. Again, the ols_full model performs the worst, the ols_pars model performs slightly better, and the remaining forecast models outperform substantially. The realized excess returns of decile portfolios based on the nn_1 architecture are most promising, ranging from − 1.25% to 2.01% per month, resulting in a monthly H–L spread of 3.26%. This implies a return ratio, i.e., ratio between realized and predicted average H–L spread, of 1.04 (= 3.26% ÷ 3.13%), which is almost identical to the time series average of predictive slopes (see Table 2, Panel A). The annualized Sharpe ratio is 3.89, with a corresponding *t*-statistic of 13.96.

Given that higher average returns could be caused by higher systematic risk, it is necessary to examine risk-adjusted performance. We calculate alphas of the decile portfolios relative to the CAPM and the Fama and French (2015) five-factor model (FF5M), which we extend here by using Carhart's (1997) momentum factor. As the results are similar but less conservative for the unconditional version, where betas are assumed to be time-constant, we only present conditional alphas for the sake of brevity. The beginning-of-month aggregate dividend yields are used to capture time variability in betas (Chordia and Avramov 2006; Ferson and Schadt 1996; Shanken 1990). Table 5 presents the time series averages of monthly figures.

Overall, the risk-adjusted values are very close to their raw counterparts, suggesting that the positive average H–L returns are not driven mainly by higher systematic risk. Additionally, all the patterns observable in Table 4 emerge in Table 5 as well. The performance is better for the equal-weighting scheme, the alphas line up almost monotonically with the average predicted excess returns (regardless of the underlying systematic risk model), and the results hold for all forecast models.

Investment portfolio performance

Since the decile portfolio sorts suggest that some forecast models are more promising than others, we next explore whether these models provide superior value-added to investors under realistic trading assumptions. We visually summarize the cumulative performance for long- and short-only investment strategies based on each forecast model, taking the cumulative market excess return as a benchmark. We derive cumulative performance by applying a rule-based approach. At the end of each month, it selects the top decile stocks with the highest expected excess returns, or the bottom decile stocks with the lowest expected excess returns.[18] We present the results for both weighting schemes. Figure 5 depicts the cumulative performance of investments of €1 in each strategy at the beginning of January 2000.

The results again reflect our previous findings and are qualitatively comparable for both weighting schemes. All investment strategies are capable of dissecting the market universe into high- and low-performing stocks. However, that ability varies substantially among the forecast models and leads to considerable performance discrepancies. Again, we observe that the ols_full model performs the worst, the nn_1 model performs the best, and the remaining models lie in between.

As the portfolio performance tends to be driven by the long side rather than the short side (see Tables 4 and 5), and due to widespread short-selling restrictions in the industry, we now focus on a long-only version of our investment strategy.[19] We compute return and risk figures for the market portfolio and each forecast portfolio. We also take into account transaction costs when measuring performance. In line with Bollerslev et al. (2018) and Koijen et al. (2018), we compute the portfolio's transaction costs (PTC) by incorporating the portfolio's one-sided turnover (PTO) and half-spread (PHS) at the end of each month t.[20]

Table 6 depicts the return and risk metrics for each portfolio (both before and after transaction costs). The terminal value is presented at the end of December 2020, as well as annualized return and volatility, maximum drawdown, annualized Sharpe ratio, and annualized information ratio (relative to the market portfolio). It also shows actual and applied transaction costs.

Independent of the weighting scheme, all the forecast portfolios substantially outperform the market. However, our main objective is to examine whether interactions and nonlinear effects add incremental predictive power to a parsimonious forecast model that has already been proven to

[18] The results remain qualitatively the same for different restructuring periods and for different quantile definitions.

[19] The results remain qualitatively the same for a long-short version of our investment strategies.

[20] We calculate the PTO as the sum of absolute changes in the value weight of the $i = 1, \dots, n$ stocks included in the portfolio, divided by two to avoid double-counting buys and sells, i.e., $\mathrm{PTO}_t = \frac{1}{2} \times \sum_{i=1}^{n} \left| w_{i,t-1} \times \left(1 + r_{i,t}\right) - w_{i,t} \right|$, where $w_{i,t}$ is the value weight assigned to stock i at the end of month t, and $r_{i,t}$ is the stock's excess return over month t. For equal-weighted portfolios, we compute the PHS as the equal-weighted average of the stocks' quoted half-spreads, i.e., $\mathrm{PHS}_t = \frac{1}{n} \times \sum_{i=1}^{n} \frac{1}{2} \times \frac{(\mathrm{ASK}_{i,t} - \mathrm{BID}_{i,t})}{\frac{1}{2} * (\mathrm{ASK}_{i,t} + \mathrm{BID}_{i,t})}$. For value-weighted portfolios, we calculate the PHS as the value-weighted average. We derive the PTC as the product of the PTO and PHS figures, i.e., $\mathrm{PTC}_t = \mathrm{PTO}_\mathrm{t} \times \mathrm{PHS}_t$.

Table 5 Conditional alphas for expected return-sorted portfolios

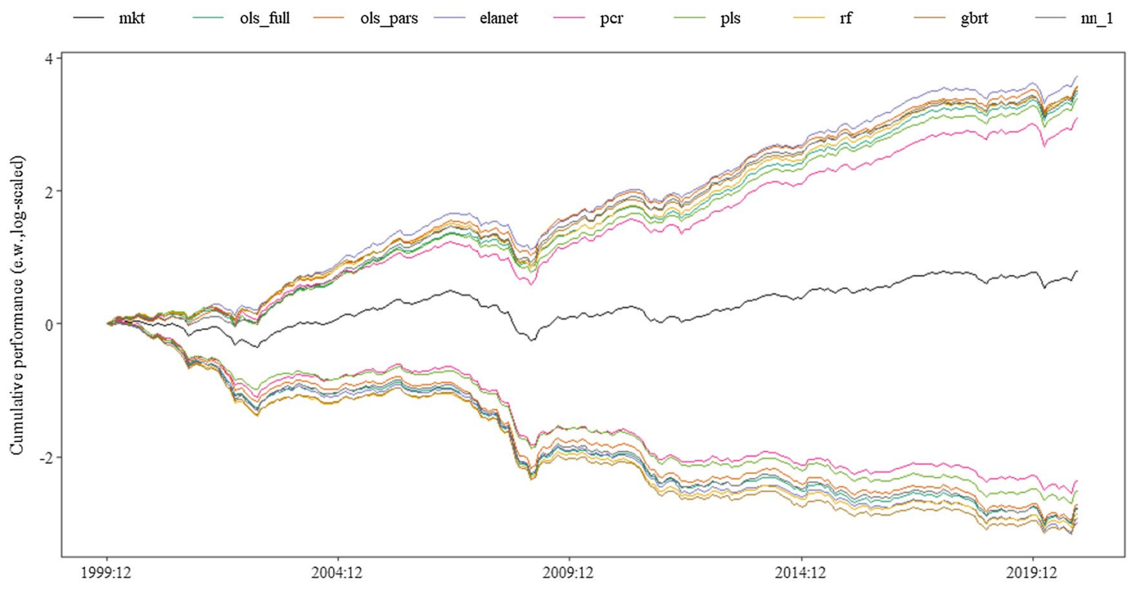
	Equal-weighted						Value-weighted					
	a_{CAPM} [%]	$t(a_{CAPM})$	R^2_{CAPM}	a_{FF5M} [%]	$t(a_{FF5M})$	R^2_{FF5M}	a_{CAPM} [%]	$t(a_{CAPM})$	R^2_{CAPM}	a_{FF5M} [%]	$t(a_{FF5M})$	R^2_{FF5M}
ols_full												
Low (L)	−1.35	−7.21	0.80	−1.43	−11.90	0.92	−1.02	−4.98	0.80	−0.65	−3.01	0.82
2	−0.56	−3.85	0.84	−0.73	−8.10	0.95	−0.41	−2.07	0.82	−0.29	−1.80	0.85
3	−0.21	−1.89	0.87	−0.41	−7.17	0.96	−0.28	−2.77	0.89	−0.31	−3.09	0.90
4	0.14	1.21	0.86	−0.07	−1.15	0.96	0.02	0.23	0.87	0.09	0.85	0.89
5	0.19	1.61	0.88	−0.08	−1.53	0.97	0.04	0.37	0.88	0.02	0.13	0.91
6	0.45	4.20	0.87	0.18	3.15	0.97	0.14	1.43	0.90	0.04	0.39	0.91
7	0.56	4.62	0.86	0.28	5.01	0.97	0.39	3.43	0.86	0.24	2.24	0.88
8	0.63	5.68	0.88	0.37	7.02	0.97	0.23	2.48	0.90	0.16	1.82	0.92
9	0.78	5.94	0.85	0.48	6.33	0.96	0.24	1.88	0.86	0.10	0.76	0.86
High (H)	1.32	7.79	0.84	1.07	8.92	0.94	0.26	1.63	0.84	0.33	1.93	0.86
H–L	2.67	12.15	0.04	2.51	12.10	0.09	1.27	4.35	0.07	0.99	2.95	0.14
ols_pars												
Low (L)	−1.45	−7.60	0.78	−1.46	−10.52	0.91	−1.13	−5.47	0.75	−0.67	−3.56	0.82
2	−0.60	−4.05	0.84	−0.75	−9.62	0.95	−0.44	−3.60	0.83	−0.30	−2.16	0.84
3	−0.30	−2.38	0.84	−0.47	−6.01	0.95	−0.37	−2.33	0.84	−0.27	−1.78	0.85
4	−0.04	−0.37	0.88	−0.25	−4.03	0.97	−0.10	−1.02	0.86	−0.08	−0.72	0.89
5	0.18	1.96	0.88	−0.06	−0.84	0.96	0.04	0.37	0.86	0.04	0.32	0.88
6	0.38	3.51	0.89	0.10	1.59	0.97	0.25	2.67	0.89	0.14	1.12	0.90
7	0.50	4.38	0.87	0.20	4.25	0.96	0.15	1.63	0.90	0.00	0.01	0.90
8	0.76	6.27	0.87	0.48	8.10	0.97	0.26	2.48	0.89	0.17	1.67	0.89
9	1.02	7.65	0.87	0.72	10.07	0.96	0.17	1.31	0.87	0.08	0.55	0.87
High (H)	1.49	9.77	0.83	1.16	11.89	0.93	0.53	4.37	0.83	0.26	2.02	0.86
H–L	2.94	14.31	0.00	2.62	12.99	0.11	1.66	6.13	0.00	0.93	3.71	0.24
elanet												
Low (L)	−1.59	−8.37	0.79	−1.64	−12.70	0.93	−1.40	−5.76	0.79	−0.86	−3.55	0.84
2	−0.64	−4.73	0.84	−0.72	−7.30	0.94	−0.65	−4.31	0.84	−0.49	−3.42	0.86
3	−0.26	−2.04	0.87	−0.46	−6.28	0.96	−0.38	−3.21	0.87	−0.32	−2.47	0.88
4	0.12	1.01	0.86	−0.11	−1.72	0.96	0.07	0.67	0.86	0.13	1.10	0.87
5	0.14	1.34	0.87	−0.12	−2.07	0.96	−0.14	−1.29	0.89	−0.09	−0.89	0.90
6	0.32	2.60	0.86	0.01	0.13	0.96	0.20	1.67	0.86	0.18	1.82	0.88
7	0.57	5.31	0.87	0.29	5.29	0.97	0.22	2.20	0.89	0.13	1.44	0.89
8	0.75	6.09	0.85	0.40	5.52	0.96	0.27	2.42	0.88	0.11	0.90	0.89
9	1.01	7.74	0.86	0.75	9.68	0.95	0.46	3.27	0.84	0.24	1.71	0.85
High (H)	1.52	9.77	0.84	1.27	11.82	0.95	0.54	3.35	0.83	0.42	2.63	0.84
H–L	3.11	16.76	0.02	2.91	15.03	0.14	1.95	6.13	0.10	1.28	4.19	0.22
pcr												
Low (L)	−1.60	−7.51	0.79	−1.62	−10.91	0.92	−1.30	−4.65	0.73	−0.83	−3.40	0.80
2	−0.59	−4.28	0.85	−0.67	−8.75	0.95	−0.62	−3.73	0.81	−0.18	−1.03	0.84
3	−0.29	−2.55	0.87	−0.46	−6.35	0.96	−0.30	−2.38	0.88	−0.07	−0.60	0.91
4	−0.04	−0.35	0.87	−0.29	−4.90	0.96	−0.14	−1.27	0.87	−0.19	−1.90	0.89
5	0.18	1.88	0.88	−0.05	−1.00	0.97	0.05	0.46	0.86	0.13	1.32	0.87
6	0.42	4.39	0.88	0.18	4.03	0.96	0.10	1.04	0.87	−0.04	−0.38	0.90
7	0.62	4.95	0.85	0.30	5.05	0.97	0.21	1.76	0.86	−0.07	−0.62	0.88
8	0.68	5.66	0.88	0.36	5.70	0.97	0.20	2.00	0.90	0.01	0.06	0.91
9	0.99	7.36	0.86	0.63	8.82	0.96	0.43	3.75	0.90	0.29	2.66	0.91
High (H)	1.56	9.23	0.84	1.28	11.76	0.94	0.72	5.02	0.84	0.54	3.59	0.86
H–L	3.16	13.90	0.01	2.91	13.63	0.14	2.02	5.42	0.05	1.37	4.38	0.27
Pls												
Low (L)	−1.68	−7.86	0.79	−1.71	−12.06	0.92	−1.59	−6.40	0.78	−1.21	−5.87	0.83
2	−0.57	−3.93	0.85	−0.70	−8.21	0.96	−0.60	−3.80	0.84	−0.42	−2.58	0.86
3	−0.26	−2.37	0.87	−0.44	−7.99	0.96	−0.27	−2.50	0.84	−0.01	−0.06	0.86

Table 5 (continued)

	Equal-weighted						Value-weighted					
	a_{CAPM} [%]	$t(a_{CAPM})$	R^2_{CAPM}	a_{FF5M} [%]	$t(a_{FF5M})$	R^2_{FF5M}	a_{CAPM} [%]	$t(a_{CAPM})$	R^2_{CAPM}	a_{FF5M} [%]	$t(a_{FF5M})$	R^2_{FF5M}
4	0.04	0.41	0.87	− 0.19	− 2.78	0.96	− 0.04	− 0.39	0.90	0.00	0.05	0.91
5	0.17	1.65	0.87	− 0.13	− 1.96	0.96	− 0.13	− 1.47	0.89	− 0.24	− 2.71	0.91
6	0.39	3.36	0.85	0.08	1.59	0.97	0.28	2.70	0.86	0.15	1.32	0.89
7	0.60	5.42	0.87	0.32	5.54	0.96	0.26	3.05	0.89	0.18	2.10	0.89
8	0.75	6.58	0.88	0.46	6.93	0.97	0.31	2.67	0.87	0.07	0.65	0.88
9	0.95	7.47	0.87	0.66	7.71	0.96	0.26	1.84	0.86	0.14	1.08	0.87
High (H)	1.55	9.13	0.83	1.30	9.61	0.93	0.50	3.34	0.82	0.35	2.20	0.85
H–L	3.24	14.10	0.00	3.01	13.03	0.10	2.08	6.48	0.10	1.56	5.76	0.31
Rf												
Low (L)	− 1.58	− 7.88	0.80	− 1.59	− 11.67	0.94	− 1.11	− 4.69	0.76	− 0.91	− 4.11	0.83
2	− 0.67	− 5.48	0.85	− 0.74	− 8.21	0.95	− 0.77	− 4.56	0.80	− 0.52	− 3.08	0.86
3	− 0.23	− 2.00	0.86	− 0.48	− 5.36	0.95	− 0.30	− 2.51	0.87	− 0.22	− 1.79	0.88
4	0.03	0.30	0.87	− 0.14	− 1.97	0.95	0.00	0.03	0.88	− 0.03	− 0.34	0.88
5	0.20	1.60	0.86	− 0.04	− 0.76	0.96	0.13	1.37	0.86	0.09	0.87	0.87
6	0.40	3.65	0.86	0.13	2.21	0.96	0.34	2.96	0.84	0.39	3.11	0.85
7	0.57	4.44	0.84	0.25	3.37	0.95	0.25	2.08	0.85	0.04	0.34	0.87
8	0.70	5.27	0.86	0.41	5.70	0.95	0.27	2.13	0.85	0.17	1.65	0.86
9	0.99	7.66	0.86	0.67	9.76	0.96	0.30	2.22	0.85	0.06	0.44	0.86
High (H)	1.53	8.75	0.82	1.19	9.78	0.93	0.42	1.90	0.81	0.24	1.16	0.84
H–L	3.11	14.87	0.10	2.79	13.35	0.29	1.54	4.89	0.04	1.15	3.88	0.26
gbrt												
Low (L)	− 1.56	− 7.70	0.79	− 1.58	− 11.63	0.93	− 1.24	− 6.24	0.80	− 0.95	− 5.10	0.87
2	− 0.69	− 5.50	0.86	− 0.75	− 9.55	0.95	− 0.76	− 5.29	0.79	− 0.43	− 2.30	0.81
3	− 0.18	− 1.64	0.86	− 0.38	− 5.59	0.96	− 0.11	− 0.90	0.86	− 0.04	− 0.29	0.86
4	− 0.02	− 0.13	0.86	− 0.28	− 4.08	0.96	− 0.03	− 0.26	0.84	− 0.18	− 1.38	0.86
5	0.12	1.25	0.89	− 0.11	− 2.26	0.97	0.01	0.14	0.87	− 0.04	− 0.32	0.88
6	0.34	2.88	0.85	0.08	1.34	0.96	0.07	0.68	0.86	0.01	0.10	0.86
7	0.56	4.91	0.86	0.29	4.82	0.96	0.10	1.08	0.89	0.12	1.25	0.90
8	0.79	6.54	0.86	0.51	8.49	0.96	0.25	2.44	0.89	0.09	0.85	0.90
9	1.01	7.51	0.86	0.62	8.97	0.96	0.58	4.57	0.88	0.37	3.37	0.90
High (H)	1.56	9.24	0.84	1.27	10.16	0.93	0.44	2.35	0.83	0.19	1.02	0.85
H–L	3.12	14.35	0.03	2.85	12.83	0.18	1.68	5.66	0.02	1.13	4.06	0.30
nn_1												
Low (L)	− 1.66	− 8.67	0.79	− 1.59	− 12.28	0.93	− 1.42	− 6.92	0.78	− 0.79	− 4.11	0.88
2	− 0.61	− 4.50	0.85	− 0.65	− 6.52	0.95	− 0.60	− 4.23	0.82	− 0.34	− 2.39	0.86
3	− 0.32	− 2.99	0.87	− 0.50	− 7.14	0.96	− 0.51	− 4.26	0.88	− 0.33	− 2.60	0.88
4	0.03	0.27	0.88	− 0.25	− 3.33	0.96	− 0.01	− 0.05	0.85	− 0.02	− 0.12	0.86
5	0.14	1.24	0.86	− 0.12	− 1.90	0.96	0.18	1.39	0.85	0.23	1.52	0.85
6	0.33	3.12	0.87	0.08	1.32	0.96	0.07	0.62	0.86	0.01	0.08	0.87
7	0.56	5.21	0.87	0.27	4.46	0.96	0.25	2.41	0.88	0.12	1.21	0.89
8	0.75	5.56	0.86	0.45	6.23	0.96	0.24	1.98	0.85	0.16	1.46	0.86
9	1.05	7.15	0.85	0.73	8.57	0.96	0.37	2.89	0.85	0.09	0.67	0.87
High (H)	1.67	9.68	0.82	1.26	11.58	0.94	0.67	4.50	0.82	0.24	1.80	0.86
H–L	3.33	15.05	0.07	2.85	14.02	0.38	2.09	7.27	0.11	1.03	4.60	0.52

This table presents conditional alphas [relative to the CAPM and the Fama and French (2015) five-factor model, extended by Carhart's (1997) momentum factor (FF5M)] of decile portfolios based on the expected excess returns that are obtained from each machine learning model introduced in the "Forecast models" section. Conditional alphas incorporate the beginning-of-month aggregate dividend yields to capture time variability in betas. The *t*-statistics (testing the null hypothesis that conditional alphas are zero) are calculated using the monthly point estimates, incorporating a Newey and West (1987) correction with four lags to account for possible heteroscedasticity and autocorrelation in the alphas. In addition, the regression R^2 metrics are reported. The numbers are presented for both equal- and value-weighted decile portfolios. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end

Panel A: Equal-weighted cumulative performance



Panel B: Value-weighted cumulative performance

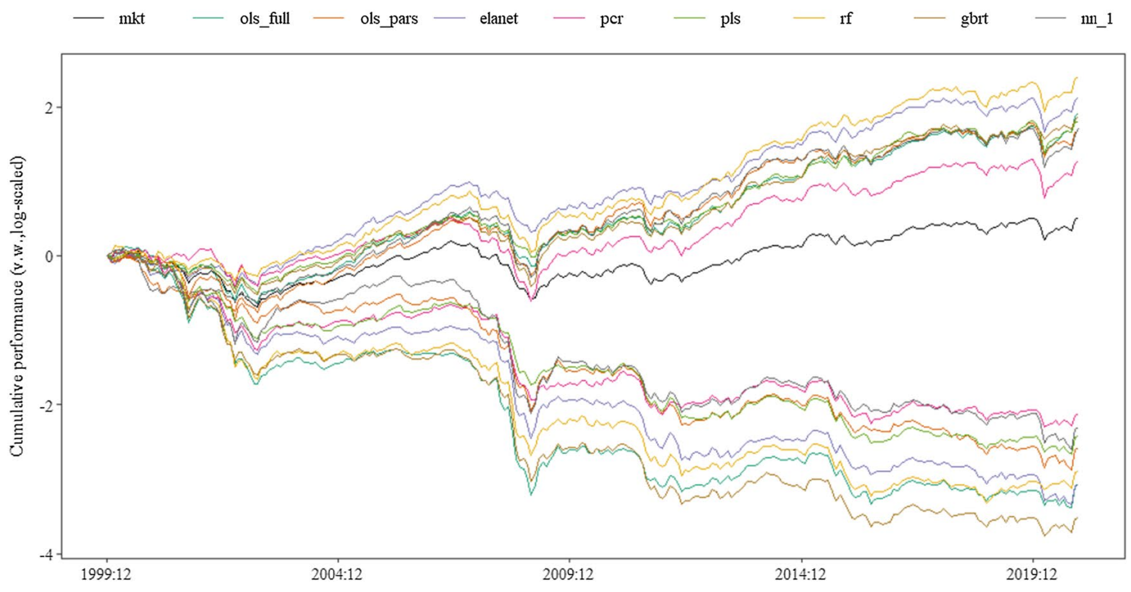


Fig. 5 Cumulative performance of expected return-sorted portfolios with monthly restructuring. This figure depicts the performance (logarithmic scale) of the market portfolio and each machine learning portfolio introduced in the "Forecast models" section. Portfolio values are scaled to €1 at the beginning of January 2000 and presented for long- and short-only portfolios, for both the equal- and value-weighting scheme, and gross of transaction costs. The calculation of transaction costs and the choice of applied transaction costs are explained in the "Investment portfolio performance" section. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end.

perform well. This is why we use the ols_pars model as our conservative benchmark. In addition, as we are primary interested in the forecast models' ability to select stocks ex ante that tend to perform well ex post, we opt for the naïve equal-weighting scheme in presenting and discussing investment portfolio performance.

Table 6 Cumulative performance of long-only forecast portfolios with monthly restructuring

	mkt	ols_full	ols_pars	elanet	pcr	pls	rf	gbrt	nn_1
Panel A: Equal-weighted									
Before transaction costs									
Terminal value [€]	2.77	48.65	70.29	78.33	86.29	86.22	81.96	88.05	107.18
Excess return annualized [%]	4.96	20.32	22.45	23.08	23.65	23.64	23.35	23.77	24.93
Std annualized [%]	16.21	19.86	18.20	18.97	19.07	18.54	18.66	18.81	17.82
Maximum drawdown [%]	60.32	55.34	51.24	48.63	56.27	52.12	49.33	48.48	50.12
Sharpe ratio	0.31	1.03	1.24	1.22	1.25	1.28	1.26	1.27	1.41
Information ratio		2.48	3.57	3.60	3.58	3.42	3.32	3.40	4.27
Actual transaction costs [%]									
Avg PTO		39.54	27.17	36.24	34.80	33.33	32.82	33.95	31.18
Avg PHS		0.63	0.57	0.64	0.63	0.65	0.61	0.60	0.59
Avg PTC		0.24	0.16	0.23	0.23	0.22	0.20	0.21	0.18
Applied transaction costs [%]									
Avg PTC		0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57
After transaction costs									
Terminal value [€]	2.77	11.75	17.02	18.97	20.91	20.90	19.86	21.34	26.01
Excess return annualized [%]	4.96	12.45	14.45	15.04	15.58	15.57	15.29	15.69	16.79
Std annualized [%]	16.21	19.86	18.20	18.97	19.07	18.54	18.66	18.81	17.82
Maximum drawdown [%]	60.32	60.59	56.94	54.63	61.41	57.73	55.25	54.49	55.73
Sharpe ratio	0.31	0.63	0.80	0.80	0.82	0.84	0.82	0.84	0.95
Information ratio		1.21	1.94	2.01	2.04	1.95	1.87	1.94	2.53
Panel B: Value-weighted									
Before transaction costs									
Terminal value [€]	1.67	3.58	6.00	6.80	11.08	6.47	5.47	5.32	8.40
Excess return annualized [%]	2.47	6.26	8.90	9.56	12.13	9.30	8.43	8.29	10.67
Std annualized [%]	16.12	20.94	17.39	19.40	19.56	19.29	22.22	20.79	18.50
Maximum drawdown [%]	54.62	67.26	44.25	52.77	55.77	60.60	61.15	56.99	48.87
Sharpe ratio	0.15	0.30	0.51	0.49	0.62	0.48	0.38	0.40	0.58
Information ratio		0.39	0.91	0.86	1.13	0.75	0.53	0.62	1.06
Actual transaction costs [%]									
Avg PTO		45.98	29.59	44.58	41.48	39.41	41.29	36.66	37.29
Avg PHS		0.17	0.14	0.16	0.18	0.18	0.22	0.18	0.15
Avg PTC		0.09	0.05	0.08	0.08	0.07	0.09	0.07	0.06
Applied transaction costs [%]									
Avg PTC		0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57
After transaction costs									
Terminal value [€]	1.67	0.85	1.43	1.62	2.65	1.55	1.30	1.27	2.01
Excess return annualized [%]	2.47	– 0.76	1.72	2.34	4.76	2.10	1.27	1.14	3.38
Std annualized [%]	16.12	20.94	17.39	19.40	19.56	19.29	22.22	20.79	18.50
Maximum drawdown [%]	54.62	71.23	52.00	61.29	59.84	66.50	64.76	65.17	53.53
Sharpe ratio	0.15	– 0.04	0.10	0.12	0.24	0.11	0.06	0.06	0.18
Information ratio		– 0.33	– 0.10	– 0.02	0.27	– 0.04	– 0.11	– 0.14	0.12

This table presents return and risk characteristics of the market portfolio and each machine learning portfolio introduced in the "Forecast models" section. Portfolio values are scaled to €1 at the beginning of January 2000. The numbers are presented for long-only portfolios, for both the equal- and value-weighting scheme, and both gross and net of transaction costs. The calculation of transaction costs and the choice of applied transaction costs are explained in the "Investment portfolio performance" section. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end.

Ignoring transaction costs, the ols_pars portfolio yields a terminal value of €70.29. The excess return is substantially higher than that for the market portfolio (22.45% vs. 4.96%), while volatility is only slightly elevated (18.20% vs. 16.21%). This increases the Sharpe ratio (1.24 vs. 0.31). Moreover, the OLS-based forecast portfolio exhibits a lower maximum drawdown (51.24% vs. 60.32%) as well as a high information ratio (3.57). Because the ols_pars model outperforms, it is suitable to serve as our conservative benchmark. However, adding interactions and nonlinear effects to the ols_pars model without any restriction, leading to the ols_full model, weakens the return and risk figures.

Implementing regularization (i.e., variable selection/shrinkage or dimension reduction) to restrict the complexity of the ols_full model, or adding interactions and nonlinearity with a tree-based approach, adds substantial predictive power. The volatilities are slightly higher than those of the ols_pars model (ranging from 18.54% for pls and 19.07% for pcr), and the excess returns range from 23.08% (elanet) to 23.77% (gbrt). The machine learning portfolios sharply increase the terminal value relative to the ols_pars portfolio, with similar figures for maximum drawdown, Sharpe ratio, and information ratio. The nn_1 model performs the best: It raises the terminal value relative to the ols_pars portfolio by €36.89 (or 52.48%) to €107.18 and increases the excess return to 24.93%, with similar figures for standard deviation (17.82%) and maximum drawdown (50.12%). This translates into the highest Sharpe ratio (1.41) and information ratio (4.27).

Next, we account for transaction costs. The average PTO ranges from 27.17% (ols_pars) to 39.54% (ols_full) and is slightly higher for the value-weighting scheme. In contrast, the average PHS is much higher for the equal-weighted scheme (60 bps vs. 20 bps). By definition, it puts larger weights on smaller stocks that are less liquid and therefore more expensive (in terms of higher bid-ask spreads). As a result, the average PTCs are nearly three times larger for the equal-weighted portfolios.

Although the transaction cost models of Bollerslev et al. (2018) and Koijen et al. (2018) are quite conservative, we follow an even more cautious approach and deduct 57 bps from each portfolio's monthly excess returns.[21] The actual transaction costs are many times lower than this deduction, and therefore the applied transaction costs offer a conservative lower bound for the performance of a realistic trading strategy. Overall, each equal-weighted portfolio continues to outperform the market. The performance of value-weighted portfolios (except for pcr and nn) falls slightly under the benchmark. We attribute this to the overly conservative transaction cost discount.

Classification-based portfolio formation

Our results indicate that interactions and nonlinear effects are important and could be exploited by adequately trained and tuned machine learning models. The forecast models handle the high dimensionality issue with varying degrees of success, resulting in different levels of predictive performance. To evaluate the economic predictive power, we compare the cumulative performance of investment portfolios. They are formed through a two-step procedure that uses forecast models to derive expected return estimates first, and selects stocks with the highest expected return forecasts into long-only portfolios second.

Considering this "detour" through the estimation of stock-level expected returns, we next study whether a single-step portfolio formation process can further improve the predictive performance. We follow the popular learning-to-rank (LTR) classification approach. Instead of incorporating realized excess returns as labels to train a regression model, the stocks from the training and validation sample are cross-sectionally sorted into decile portfolios based on realized excess returns. These sortings are then used as aggregate labels in fitting a rank-based classification model. Top-decile stocks are preferable to bottom-decile stocks, so decile portfolio labels serve as preference ranks.

Such pre-estimation sortings, rather than classifying stocks based on expected return estimates, can help avoid the noise in stock-level excess returns, while maintaining the signal that is most important from a trading perspective, i.e., the correct classification into a decile portfolio. The LTR approach computes probability estimates for the affiliation of a stock to each decile portfolio separately and then sums these probabilities into an aggregate probability score, which it uses to classify the stock into a specific decile portfolio.

Machine learning method: Support vector machines

As discussed earlier, support vector machines (SVMs) are a popular classification method. The idea behind SVMs is to search for hyperplanes that territorially divide a multidimensional vector space (our sample of firm observations, consisting of stock-level predictors and decile portfolio labels) into groups of vectors that belong to the same class. Each potential hyperplane is located in an area where vectors of two different classes are close together. To increase computational speed, SVMs do not always use all vectors from the vector space. Rather, it focuses on those in the immediate neighborhood of the potential hyperplane, or so-called

[21] Novy-Marx and Velikov (2016) suggest a lump-sum discount on the portfolio's return, which ranges from 20bps to 57bps for mid-turnover strategies, with an average PTO between 14 and 35%. Since all forecast

Footnote 21 (continued)

portfolios exhibit PTOs within this interval or only slightly above, they are likely to be mid- rather than low-turnover strategies (average PTO < 10%), or high-turnover strategies (average PTO > 90%).

support vectors. The algorithm then specifies the optimal hyperplane by aiming to (1) maximize the distance of correctly classified support vectors from the hyperplane, and (2) minimize the number of misclassified support vectors.

In a multi-class scenario, SVMs fit optimal hyperplanes by means of a pairwise class comparison. In theory, by allowing for an unrestricted nonlinear transformation of the vector space and with indefinite computational power and time, the algorithm can avoid any misclassifications. But since this likely leads to overfitting, regardless of obvious computational limitations, SVMs must be strongly regularized.

To this end, we use a radial basis function (RBF) kernel for a proper nonlinear transformation of the vector space. We simultaneously apply three types of regularization. First, we constrain the influence of any single vector, i.e., we restrict the space within which it can serve as a support vector. A smaller vector influence γ avoids enabling vectors to serve as supports for overly distant hyperplanes. Second, we set the permitted number of misclassified vectors to a positive value, i.e., we allow for a certain number of misclassifications. Smaller misclassification costs c allow us to ignore more of the misclassified support vectors, while continuing to fit optimal hyperplanes. To tune both parameters, we select a loss function that meets our objective to classify stocks into *ranked* decile portfolios, based on the confusion matrix that contrasts predicted and realized classes. For binary (two-class) classifications, potential outcomes are true positives (TPs) if the predicted and realized class equal 1, true negatives (TNs) if predicted and realized class equal 0, false positives (FPs) if predicted class equals 1 but realized class equals 0, and false negatives (FNs) if predicted class equals 0 but realized class equals 1.

In a two-class scenario, classification performance is usually evaluated using confusion matrices, together with a broad set of classification measures, in particular *accuracy*, i.e., $\frac{\#(\mathrm{TP+TN})}{\#(\mathrm{TP+FP+FN+TN})}$, *sensitivity*, i.e., $\frac{\#\mathrm{TP}}{\#(\mathrm{TP+FN})}$, and *specificity*, i.e., $\frac{\#\mathrm{TN}}{\#(\mathrm{TN+FP})}$. Multi-class classification metrics are derived as follows: Separately for each class, the binary one-against-all approach is used to compute hypothetic two-class numbers. It considers the class under investigation as 1, and all remaining classes as 0. Aggregate metrics are then calculated as the weighted average of each class' individual metric, taking the number of realized cases within each class as weights. Note that the aggregate accuracy can equivalently be computed as the number of true classifications (cases in the cells on the confusion matrix's left diagonal) divided by the number of total classifications (cases in any cell of the confusion matrix). To give an example, Table 10 of Appendix 2 provides a visualization of a three-level classification with inherent rank order ($3 \succ 2 \succ 1$). The multi-class accuracy measure in this illustration is $\frac{10+10+10}{100} = 30\%$ for both scenarios.

Traditional classifications would aim to maximize this multi-class accuracy. However, this loss function is not applicable for an LTR classification because it does not consider the inherent rank order, i.e., the severity of the misclassifications. Although exhibiting the same 30% accuracy, the predictions in scenario A differ from their realizations more strongly than in scenario B. To put this in a trading context, selecting a stock that is expected to be in the best decile portfolio, but is actually in the second-best decile portfolio, is much less problematic than accidently selecting a stock from the worst decile portfolio. This is why the goal of our SVM approach is to minimize an objective function that penalizes the difference between predicted and realized class ranks, i.e., $\Phi(\theta) = \sum_{i=1}^{N} \left|\mathrm{pred}_i - \mathrm{real}_i\right|$, where $\mathrm{pred}_i \in \{1, \dots, 10\}$ and $\mathrm{real}_i \in \{1, \dots, 10\}$ are the predicted and realized decile portfolio ranks of stock $i \in \{1, \dots, N\}$, respectively.

Multi-class SVMs are computational intensive Therefore, adapt the ensemble approach from the neural network architecture (see the "Neural networks" section). At each re-estimation date, we randomly split the training sample into ten distinct subsamples and train ten independent SVMs from these. This increases the computational speed, but still enables the algorithm to see the entire training sample. We average the different predictions into a single ensemble prediction, which additionally increases the signal-to-noise ratio.

Our SVM approach uses a scoring model to classify stocks into decile portfolios. Based on the disaggregated vector space, it estimates the likelihood that a specific stock will be part of a specific decile portfolio in the following month $t + 1$. It aggregates those decile-specific probabilities into a probability score S_i. It is higher for stocks that are ex ante expected to be in the decile portfolio with high ex post realized returns, i.e., $S_{i,t} = \sum_{j=1}^{10} w_j \times p_{ij,t}$, where $w_j \in \{1, \dots, 10\}$ is the rank of the decile portfolio, and $p_{ij,t} \in (0, 1)$ is the probability that stock i will be part of decile portfolio j, while incorporating all the information available at the end of month t.

Out-of-sample tests

Statistical predictive performance

Our primary interest is potential performance differences between traditional expected return-based portfolio formation and a classification-based approach. Therefore, we next compare measures of predictive performance. To set the highest possible bar for identifying incremental predictive performance, we use the best-performing machine learning method, i.e., the nn_1 model, as our relevant benchmark. Table 7 shows the confusion matrices, together with a broad set of classification measures.

The entries in the diagonal of the confusion matrix indicate that, at an aggregate level, both models correctly

Table 7 Classification performance of forecast portfolios with monthly restructuring

		Decile-specific statistics [%]										Aggregate statistics [%]	
Panel A: Support vector machine (svm)													
		Realized											
		Low (L)	2	3	4	5	6	7	8	9	High (H)		
Predicted	Low (L)	**6,086**	3,240	2,378	2,057	1,919	1,780	1,720	1,875	2,077	3,422		
	2	3,660	**3,014**	2,841	2,616	2,531	2,394	2,227	2,159	2,346	2,734		
	3	2,939	2,914	**2,841**	2,760	2,816	2,621	2,538	2,321	2,394	2,352		
	4	2,440	2,717	2,852	**2,917**	2,867	2,797	2,669	2,581	2,403	2,233		
	5	2,152	2,576	2,792	2,851	**2,885**	2,842	2,766	2,722	2,544	2,324		
	6	1,932	2,579	2,730	2,860	2,836	**2,920**	2,962	2,793	2,712	2,109		
	7	1,879	2,423	2,620	2,853	2,824	2,899	**2,991**	2,925	2,749	2,242		
	8	1,723	2,386	2,620	2,718	2,725	2,874	2,958	**3,035**	2,965	2,372		
	9	1,869	2,380	2,527	2,549	2,621	2,789	2,859	3,035	**2,939**	2,784		
	High (H)	1,874	2,293	2,295	2,295	2,430	2,517	2,715	2,930	3,223	3,752		
Accuracy (balanced)		57.16	50.74	50.39	50.56	50.50	50.58	50.74	50.85	50.66	52.39	Accuracy	12.63
												Accuracy$_{NIR}$	10.04
												p-statistic (Acc > Acc$_{NIR}$)	0.00
Sensitivity		22.92	11.36	10.72	11.02	10.91	11.05	11.33	11.51	11.15	14.25	Sensitivity	12.62
Specificity		91.39	90.12	90.06	90.10	90.09	90.12	90.16	90.19	90.16	90.52	Specificity	90.29
Panel B: Neural networks (nn_1)													
		Realized											
		Low (L)	2	3	4	5	6	7	8	9	High (H)		
Predicted	Low (L)	**5,235**	3,092	2,623	2,489	2,282	2,145	1,958	1,921	2,007	2,802		
	2	3,384	**2,973**	2,760	2,739	2,743	2,598	2,417	2,202	2,244	2,462		
	3	2,834	2,824	**2,877**	2,836	2,813	2,640	2,620	2,426	2,385	2,241		
	4	2,551	2,658	2,781	**2,805**	2,733	2,783	2,680	2,619	2,523	2,343		
	5	2,310	2,675	2,704	2,719	**2,814**	2,817	2,809	2,766	2,607	2,233		
	6	2,121	2,570	2,727	2,762	2,799	**2,818**	2,869	2,794	2,705	2,268		
	7	2,029	2,541	2,604	2,696	2,726	2,774	**2,824**	2,930	2,854	2,427		
	8	1,982	2,492	2,594	2,640	2,649	2,774	2,802	**2,903**	2,919	2,621		
	9	2,029	2,357	2,545	2,501	2,573	2,637	2,753	2,929	**2,923**	3,105		
	High (H)	2,079	2,340	2,281	2,289	2,322	2,447	2,673	2,886	3,185	3,822		
Accuracy (balanced)		55.38	50.65	50.46	50.32	50.35	50.37	50.39	50.57	50.62	52.53	Accuracy	12.10
												Accuracy$_{NIR}$	10.04
												p-statistic (Acc > Acc$_{NIR}$)	0.00
Sensitivity		19.71	11.21	10.86	10.59	10.64	10.66	10.69	11.01	11.09	14.52	Sensitivity	12.10
Specificity		91.04	90.10	90.07	90.05	90.06	90.08	90.09	90.14	90.16	90.55	Specificity	90.23

Table 7 (continued)

This table shows the confusion matrices, together with a broad set of classification measures, of selected machine learning models (nn_1 and svm) introduced in the "Forecast models" and the "Machine learning methods: Support vector machines" sections, respectively. In a two-class scenario, *accuracy* is $\frac{\#(TP+TN)}{\#(TP+FP+FN+TN)}$, *sensitivity* is $\frac{\#TP}{\#(TP+FN)}$, and *specificity* is $\frac{\#TN}{\#(TN+FP)}$. Multi-class classification metrics are derived as follows: Separately for each class, the binary one-against-all approach is used to compute hypothetic two-class numbers. It considers the class under investigation as 1, and all remaining classes as 0. Aggregate metrics are then calculated as the weighted average of each class' individual metric, taking the number of realized cases within each class as weights. Accuracy$_{NIR}$ is the accuracy of a random classifier (no-information rate) and *p*-statistic ($Acc > Acc_{NIR}$) tests the null hypothesis that the actual accuracy is equal to the no-information rate. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end

classify stocks into decile portfolios. However, our SVM approach is slightly more accurate than the nn_1 architecture (12.63% vs. 12.10%), as well as marginally more sensitive (12.62% vs. 12.10%) and specific (90.29% and 90.23%). Compared with a 10.04% accuracy of a random classifier, the accuracy of both models is significantly higher. They work best for the top and bottom decile portfolios, and the patterns are slightly more pronounced for the short than the long side. Moreover, the SVM-based predictions differ less strongly from their realizations than those obtained from neural networks. This points towards a superior misclassification distribution, which might translate into an outperformance from an economic perspective. This is because SVMs are able to incorporate the inherent rank order of decile portfolios explicitly.

Investment portfolio performance

In a final step, recognizing Leitch and Tanner's (1991) argument that the association between statistical predictive performance (classification measures) and economic profitability may be weak, we compare the cumulative performance of long-only investment portfolios. Table 8 shows the return and risk figures from Table 6 for the ols_pars, nn_1, and svm models. The cumulative performance metrics extend our earlier findings. Both models substantially outperform the benchmark model (ols_pars), while exhibiting similar return-risk behavior. Most importantly, the numbers of the SVM-based model are slightly superior to that of the neural network for both weighting schemes. We interpret these results to mean that a classification-based approach can be even better than the best-performing expected return-based portfolio formation.

Conclusion

Using market and fundamental data for European firms, we study the predictive performance of machine learning methods in forecasting stock returns, including approaches for variable selection/shrinkage or dimension reduction, tree-based models, and neural networks. We conduct a comparative analysis in terms of statistical and economic performance metrics. Following Gu et al. (2020), we enhance the set of twenty-two predictors in Drobetz et al. (2019) by two-way interactions as well as second- and third-order polynomials to capture nonlinearity. We confirm that interactions and nonlinear effects are important and add incremental predictive power.

Machine learning methods must be adequately trained and tuned to avoid overfitting. Each model's degree of complexity varies substantially over time, and different models find similar predictors to be important. Despite these

Table 8 Cumulative performance of long-only forecast portfolios with monthly restructuring

	Equal-weighted			Value-weighted		
	ols_pars	nn_1	svm	ols_pars	nn_1	svm
Before transaction costs						
Terminal value [€]	70.29	107.18	113.57	6.00	8.40	9.07
Excess return annualized [%]	22.45	24.93	25.28	8.90	10.67	11.07
Std annualized [%]	18.20	17.82	18.36	17.39	18.50	18.85
Maximum drawdown [%]	51.24	50.12	51.80	44.25	48.87	45.87
Sharpe ratio	1.24	1.41	1.38	0.51	0.58	0.59
Information ratio	3.57	4.27	4.33	0.91	1.06	1.11
Actual transaction costs [%]						
Avg PTO	27.17	31.18	34.23	29.59	37.29	38.61
Avg PHS	0.57	0.59	0.54	0.14	0.15	0.14
Avg PTC	0.16	0.18	0.19	0.05	0.06	0.06
Applied transaction costs [%]						
Avg PTC	0.57	0.57	0.57	0.57	0.57	0.57
After transaction costs						
Terminal value [€]	17.02	26.01	27.57	1.43	2.01	2.17
Excess return annualized [%]	14.45	16.79	17.11	1.72	3.38	3.76
Std annualized [%]	18.20	17.82	18.36	17.39	18.50	18.85
Maximum drawdown [%]	56.94	55.73	57.44	52.00	53.53	50.82
Sharpe ratio	0.80	0.95	0.94	0.10	0.18	0.20
Information ratio	1.94	2.53	2.59	−0.10	0.12	0.17

This table presents return and risk characteristics of selected machine learning portfolios (ols_pars, nn_1, and svm) introduced in the "Forecast models" and the "Machine learning method: Support vector machines" sections, respectively. Portfolio values are scaled to €1 at the beginning of January 2000. The numbers are presented for long-only portfolios, for both the equal- and value-weighting scheme, and both gross and net of transaction costs. The calculation of transaction costs and the choice of applied transaction costs are explained in the "Investment portfolio performance" section. The sample includes all firms that were publicly listed in one of the nineteen Eurozone countries in any given month during the January 1990–December 2020 sample period, while the first estimates of expected excess returns are obtained in December 1999. The data coming from Thomson Reuters Datastream are collected on a monthly basis and, if currency-related, denominated in Euro. Market data are assumed to become public immediately, while fundamental data are assumed to be known four months after the fiscal year-end

commonalities, the forecast models we study exhibit different statistical predictive performance (predictive slopes closer to 1, higher predictive R^2 metrics, and positive DM test statistics). This further translates into markedly different economic profitability.

The return and risk figures of long-only investment strategies suggest that all forecast portfolios beat our linear benchmark model. The neural network architecture performs the best, also after accounting for transaction costs. Because it follows the "traditional" expected return-based portfolio formation, i.e., estimating stock-level expected returns first and aggregating stocks again into decile portfolios second, we compare its performance with a simpler, classification-based approach. We find that a support vector machine can classify stocks into decile portfolios that are even better than that obtained from neural networks. This leads to a more straightforward portfolio formation process that avoids some of the noise in stock-level excess returns, while maintaining the signal that is most important for a trading strategy, i.e., the correct classification into a decile portfolio.

Appendix 1

See Table 9.

Table 9 Details on forecast models

	Hyperparameter	Specification	Definition
ols_full	None		
ols_pars	None		
elanet	λ	(0, 1)	General strength of the penalization
	p	{0, 0.5, 1}	Weight on the lasso and ridge penalization
pcr	$number_{comp}$	{1, …, P}	Number of components included in the predictive regression
pls	$number_{comp}$	{1, …, P}	Number of components included in the predictive regression
rf	L	{1, 2, 3, 4, 5, 6}	Depth of the single regression trees
	M	{22, 33, 44}	Number of predictors randomly considered as potential split variables
	B	(10, 1,000)	Number of trees added to the ensemble prediction
gbrt	L	{1, 2, 3}	Depth of the single regression trees
	v	{0.01, 0.05, 0.1}	Weight for the learning rate shrinkage
	B	(10, 1,000)	Number of trees added to the ensemble prediction
nn_1–nn_5	$size_{batch}$	1,000	Batch size
	$number_{epochs}$	100	Number of epochs
	patience	25	Number of iterations during which the MSFE is allowed to increase in the validation sample
	dropout rate	0.25	Fractional rate of input variables that are randomly set to zero at each iteration
	ensemble	10	Number of independent seeds used for each specification family
svm	c	{0.01, 0.05, 0.1, 0.5, 1}	Vector influence
	γ	{0.01, 0.05, 0.1, 0.5, 1}	Misclassification costs

This table presents the definitions and specifications of hyperparameters of each machine learning model introduced in the "Forecast models" section

Appendix 2

See Fig 6 and Table 10.

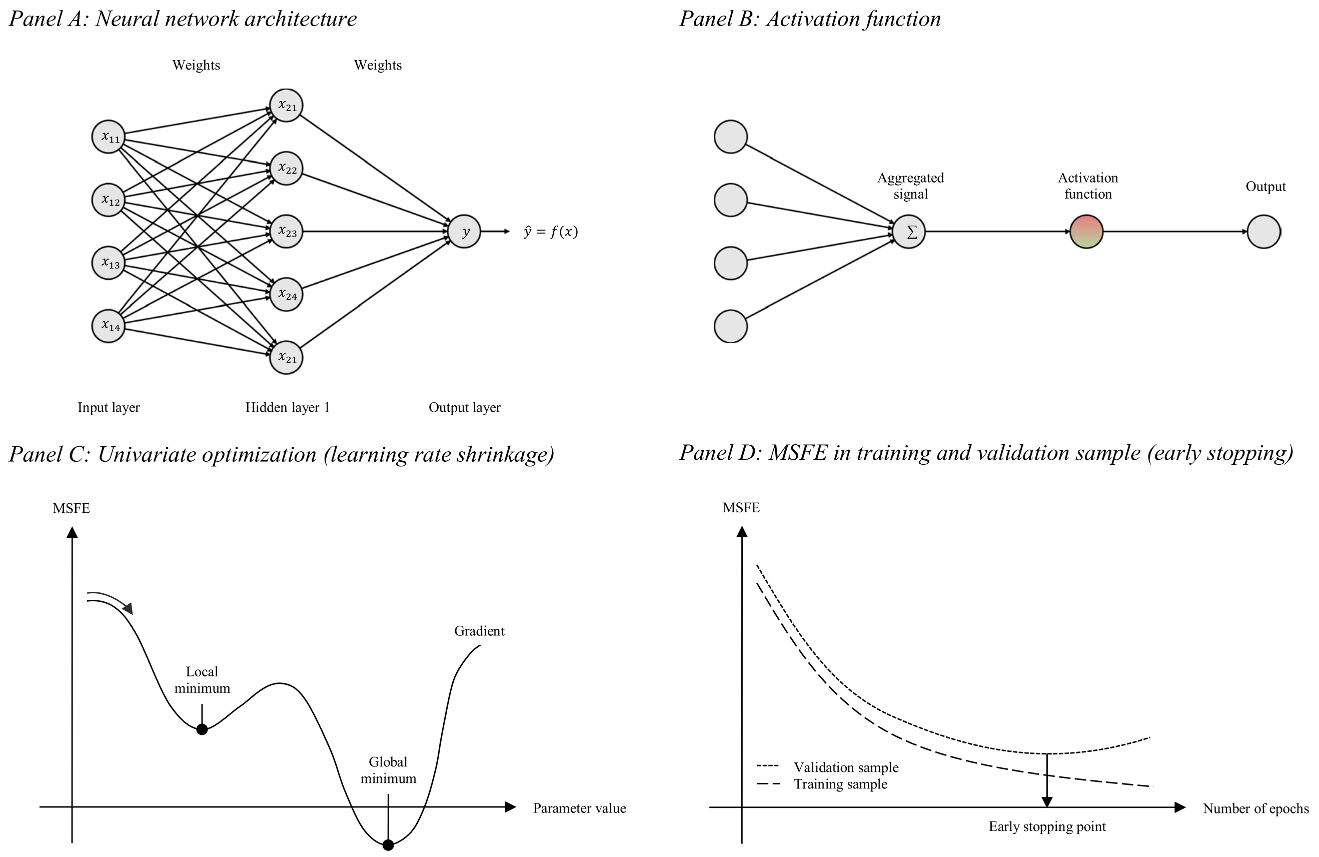


Fig. 6 Stylized visualizations | neural networks. This figure depicts four stylized visualizations that help explain the structure, functioning, and regularization of neural networks

Table 10 Stylized visualizations | classification results

		Scenario A					Scenario B		
		Realized					Realized		
		1	2	3			1	2	3
Predicted	1	10	0	30	Predicted	1	10	30	0
	2	5	10	5		2	5	10	5
	3	30	0	10		3	0	30	10

This table depicts a stylized visualization that helps explain the procedure to measure the performance of multi-class classifications. Both scenarios exhibit the same 30% accuracy, but the predictions in scenario A differ from their realizations more strongly than in scenario B (see the "Machine learning method: Support vector machines" section)

Acknowledgements We thank Dan diBartolomeo (the editor), Wolfgang Bessler, Hubert Dichtl, Andreas Neuhierl, Tatjana Puhan, and Michael Weber for helpful comments.

References

Asgharian, H., and S. Karlsson. 2008. Evaluating a Non-Linear Asset Pricing Model on International Data. *International Review of Financial Analysis* 17 (3): 604–621.

Bailey, D.H., J.M. Borwein, M.L. de Prado, and Q.J. Zhu. 2014. Pseudo-Mathematics and Financial Charlatanism: The Effects of Backtest Overfitting on Out-of-Sample Performance. *Notices of the American Mathematical Society* 61 (5): 458–471.

Bailey, D.H., J.M. Borwein, M.L. de Prado, and Q.J. Zhu. 2017. The Probability of Backtest Overfitting. *Journal of Computational Finance* 20 (4): 1460–1559.

Bansal, R., D.A. Hsieh, and S. Viswanathan. 1993. A New Approach to International Arbitrage Pricing. *The Journal of Finance* 48 (5): 1719–1747.

Bansal, R., and S. Viswanathan. 1993. No Arbitrage and Arbitrage Pricing: A New Approach. *The Journal of Finance* 48 (4): 1231–1262.

Bollerslev, T., B. Hood, J. Huss, and L.H. Pedersen. 2018. Risk Everywhere: Modeling and Managing Volatility. *Review of Financial Studies* 31 (7): 2729–2773.

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

Campbell, J.Y., and J.H. Cochrane. 1999. By Force of Habit: A Consumption-Based Explanation of Aggregate Stock Market Behavior. *Journal of Political Economy* 107 (2): 205–251.

Carhart, M.M. 1997. On Persistence in Mutual Fund Performance. *The Journal of Finance* 52 (1): 57–82.

Chapman, D.A. 1997. Approximating the Asset Pricing Kernel. *The Journal of Finance* 52 (4): 1383–1410.

Chen, L., M. Pelger, and J. Zhu. 2019. Deep Learning in Asset Pricing. *Working Paper*.

Chordia, T., and D. Avramov. 2006. Asset Pricing Models and Financial Market Anomalies. *The Review of Financial Studies* 19 (3): 1001–1040.

Chordia, T., A. Goyal, and A. Saretto. 2017. p-Hacking: Evidence from Two Million Trading Strategies. *Swiss Finance Institute Research Paper No.* 17–37: 1–53.

Cochrane, J.H. 2005. *Asset Pricing*. Revised. Princeton (NJ), US: Princeton University Press.

Cochrane, J.H. 2008. The Dog That Did Not Bark: A Defense of Return Predictability. *The Review of Financial Studies* 21 (4): 1533–1575.

Cochrane, J.H. 2011. Presidential Address: Discount Rates. *The Journal of Finance* 66 (4): 1047–1108.

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

Diebold, F., and R. Mariano. 1995. Comparing Predictive Accuracy. *Journal of Business & Economic Statistics* 13 (3): 253–263.

Dietterich, T. 2000. *Ensemble Methods in Machine Learning. Lecture Notes in Computer Science: Multiple Classifier Systems*, 1–15. Berlin, Germany: Springer.

Dittmar, R.F. 2002. Nonlinear Pricing Kernels, Kurtosis Preference, and Evidence From the Cross Section of Equity Returns. *The Journal of Finance* 57 (1): 369–403.

Drobetz, W., R. Haller, C. Jasperneite, and T. Otto. 2019. Predictability and the Cross Section of Expected Returns: Evidence from the European Stock Market. *Journal of Asset Management* 20 (7): 508–533.

Fama, E.F., and K.R. French. 1993. Common Risk Factors in the Returns on Stocks and Bonds. *Journal of Financial Economics* 33 (1): 3–56.

Fama, E.F., and K.R. French. 2008. Dissecting Anomalies. *Journal of Finance* 63 (4): 1653–1678.

Fama, E.F., and K.R. French. 2015. A Five-Factor Asset Pricing Model. *Journal of Financial Economics* 116 (1): 1–22.

Fama, E.F., and J.D. MacBeth. 1973. Risk, Return, and Equilibrium: Empirical Tests. *Journal of Political Economy* 81 (3): 607–636.

Ferson, W.E., and R.W. Schadt. 1996. Measuring Fund Strategy and Performance in Changing Economic Conditions. *The Journal of Finance* 51 (2): 425–461.

Freyberger, J., A. Neuhierl, and M. Weber. 2020. Dissecting Characteristics Nonparametrically. *The Review of Financial Studies* 33 (5): 2326–2377.

Giglio, S., and D. Xiu. 2021. Asset Pricing with Omitted Factors. *Journal of Political Economy* 129 (7): 1947–1990.

Grinold, R.C., and R.N. Kahn. 2000. *Active Portfolio Management: Quantitative Theory and Applications*. Chicago (IL), US: Probus.

Gu, S., B. Kelly, and D. Xiu. 2020. Empirical Asset Pricing Via Machine Learning. *The Review of Financial Studies* 33 (5): 2223–2273.

Gu, S., B. Kelly, and D. Xiu. 2021. Autoencoder Asset Pricing Models. *Journal of Econometrics* 222 (1): 429–450.

Hansen, L., and P. Salamon. 1990. Neural Network Ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (10): 993–1001.

Harvey, C.R., and Y. Liu. 2014. Evaluating Trading Strategies. *The Journal of Portfolio Management* 40 (5): 108–118.

Harvey, C.R., and Y. Liu. 2015. Backtesting. *Journal of Portfolio Management* 42 (1): 13–28.

Harvey, C.R., Y. Liu, and H. Zhu. 2016. … and the Cross-Section of Expected Returns. *The Review of Financial Studies* 29 (1): 5–68.

Haugen, R.A., and N.L. Baker. 1996. Commonality in the Determinants of Expected Stock Returns. *Journal of Financial Economics* 41 (3): 401–439.

Heaton, J., N. Polson, and J.H. Witte. 2016. Deep Learning for Finance. *Working Paper*.

Hou, K., C. Xue, and L. Zhang. 2020. Replicating Anomalies. *The Review of Financial Studies* 33 (5): 2019–2133.

Huerta, R., F. Corbacho, and C. Elkan. 2013. Nonlinear Support Vector Machines Can Systematically Identify Stocks with High and Low Future Returns. *Algorithmic Finance* 2 (1): 45–58.

Ioffe, S., and C. Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 448–456.

Jegadeesh, N., and S. Titman. 1993. Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency. *The Journal of Finance* 48 (1): 65–91.

Kelly, B., S. Pruitt, and Y. Su. 2019. Characteristics are Covariances: A Unified Model of Risk and Return. *Journal of Financial Economics* 134 (3): 501–524.

Koijen, R., T.J. Moskowitz, L.H. Pedersen, and E.B. Vrugt. 2018. Carry. *Journal of Financial Economics* 127 (2): 197–225.

Kozak, S., S. Nagel, and S. Santosh. 2020. Shrinking the Cross-Section. *Journal of Financial Economics* 135 (2): 271–292.

Leitch, G., and J.E. Tanner. 1991. Economic Forecast Evaluation: Profits Versus the Conventional Error Measures. *The American Economic Review* 81 (3): 580–590.

Leung, E., H. Lohre, D. Mischlich, Y. Shea, and M. Stroh. 2021. The Promises and Pitfalls of Machine Learning for Predicting Cross-Sectional Stock Returns. *The Journal of Financial Data Science* 3 (2): 21–50.

Leung, M., H. Daouk, and A.-S. Chen. 2000. Forecasting Stock Indices: A Comparison of Classification and Level Estimation Models. *International Journal of Forecasting* 16 (2): 173–190.

Levin, A. 1995. Stock Selection via Nonlinear Multi-Factor Models. In *NIPS: Proceedings of the 8th International Conference on Neural Information Processing Systems*, 966–972.

Lewellen, J. 2015. The Cross-Section of Expected Stock Returns. *Critical Finance Review* 4 (1): 1–44.

Lintner, J. 1965. Security Prices, Risk, and Maximal Gains From Diversification. *The Journal of Finance* 20 (4): 587–615.

Lo, A.W., and A.C. MacKinlay. 1990. Data-Snooping Biases in Tests of Financial Asset Pricing Models. *The Review of Financial Studies* 3 (3): 431–467.

Masters, T. 1993. *Practical Neural Network Recipes in C++*. Burlington (MA), US: Morgan Kaufmann Publishers.

McLean, R.D., and J. Pontiff. 2016. Does Academic Research Destroy Stock Return Predictability?. *The Journal of Finance* 71 (1): 5–32.

Merton, R.C. 1973. An Intertemporal Capital Asset Pricing Model. *Econometrica* 41 (5): 867–887.

Messmer, M. 2017. Deep Learning and the Cross-Section of Expected Returns. *Working Paper*.

Moritz, B., and T. Zimmermann. 2016. Tree-Based Conditional Portfolio Sorts: The Relation between Past and Future Stock Returns. *Working Paper.*

Mossin, J. 1966. Equilibrium in a Capital Asset Market. *Econometrica* 34 (2): 768–783.

Newey, W.K., and K.D. West. 1987. Hypothesis Testing With Efficient Method of Moments Estimation. *International Economic Review* 28 (3): 777–787.

Novy-Marx, R., and M. Velikov. 2016. A Taxonomy of Anomalies and Their Trading Costs. *The Review of Financial Studies* 29 (1): 104–147.

Rapach, D., J. Strauss, and G. Zhou. 2013. International Stock Return Predictability: What Is the Role of the United States?. *The Journal of Finance* 68 (4): 1633–1662.

Rapach, D., and G. Zhou. 2013. Forecasting Stock Returns. *Handbook of Economic Forecasting* 2 (A): 328–383.

Rasekhschaffe, K., and R. Jones. 2019. Machine Learning for Stock Selection. *Financial Analysts Journal* 75 (3): 70–88.

Ross, S.A. 1976. The Arbitrage Theory of Capital Asset Pricing. *Journal of Economic Theory* 13 (3): 341–360.

Schorfheide, F., and K.I. Wolpin. 2012. On the Use of Holdout Samples for Model Selection. *American Economic Review* 102 (3): 477–481.

Shanken, J. 1990. Intertemporal Asset Pricing: An Empirical Investigation. *Journal of Econometrics* 45 (1–2): 99–120.

Sharpe, W.F. 1964. Capital Asset Prices: A Theory of Market Equilibrium Unter Conditions of Risk. *The Journal of Finance* 19 (3): 425–442.

Sloan, R.G. 1996. Do Stock Prices Fully Reflect Information in Accruals and Cash Flows About Future Earnings?. *The Accounting Review* 71 (3): 289–315.

West, K. 2006. Forecast Evaluation. *In Handbook of Economic Forecasting*, ed. G. Elliott, C. Granger, and A. Timmermann. Amsterdam, Netherlands: Elsevier.

Wolff, D., and U. Neugebauer. 2019. Tree-Based Machine Learning Approaches for Equity Market Predictions. *Journal of Asset Management* 20 (4): 273–288.